UNIVERSITY OF SOUTHAMPTON

FACULTY OF PHYSICAL AND APPLIED SCIENCES

Electronics and Computer Science

Norm Based Service Selection

by

Andrew K. Douglas

Thesis for the degree of Doctor of Philosophy

March 2017

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF PHYSICAL AND APPLIED SCIENCES
Electronics and Computer Science

Doctor of Philosophy

NORM BASED SERVICE SELECTION

by Andrew K. Douglas

Distributed computing paradigms are increasingly moving towards collections of interoperating Web services. To facilitate this interoperation, dynamic discovery and selection of services is required. Existing distributed solutions for the dynamic discovery of services primarily focus on the deployment of directory, broker and matchmaking intermediaries, requiring third party participation and additional infrastructure costs.

The selection of Web services by autonomous actors has become a well-developed area of research. Service-oriented architectures can now provide for complex interactions described by semantically rich process models, thereby enabling consumption by autonomous agents. With distributed agent-based architectures becoming common, academics are increasingly looking towards norm-based approaches to offer flexible control of interacting agents.

Current semantically aware service selection methods rely on matching inputs and outputs provided by services profile models. This approach typically fails to allow actors to differentiate between services where the profile models may match, but the process models differ.

In this research, the question is asked: How can an actor with a set of known normative beliefs use these beliefs to aid service selection where IOPE matching typically falls short?

The following have been created: a model, a language and a module for the norm-based scoring of process definitions. In doing so it is shown that social norms can be used by actors to reason over the potential cost of any interaction and that this metric can provide useful context when selecting partners where the services' basic inputs and outputs may match, but process model specifications may not.

Contents

List of Figures

List of Tables

Nomenclature

ALGOL 60 Algorithmic Language (60)

API Application Programming Interface

BDI Belief-Desire-Intention

BOID Belief-Obligation-Intention-Desire

CORBA Object Management's Group's Common Object Request Broker Architecture

DAML DARPA Agent Markup Language

DARPA Defence Advanced Research Projects Agency

DCE Open Software Foundation's Distributed Computing Environment

DCOM Microsoft's Distributed Component Object Model

DII Dynamic Invocation Interface

DL Description Logic

DNS Domain Name Service

DRS Declarations in RDF Made Simple

EU European Union

FIPA Foundation for Intelligent Physical Agents

GIOP General Inter-ORB Protocol

HTML Hypertext Transfer Markup Language

HTTP Hypertext Transfer Protocol

I/O Input/Output

ICRC International Committee of the Red Cross

IDL Interface Definition Language

IGO International Governmental Organisation

IIOP Internet Inter-ORB Protocol

IMF International Monetary Fund

INGO International Non-Governmental Organisation

IOC International Olympic Committee

IOPE Inputs, Outputs, Preconditions and Effects

IOs International Organisations

IP Internet Protocol

IR Interface Repository

IRS-II Internet Reasoning Service (Version II)

IRS-III Internet Reasoning Service (Version III)

ISCT Integrated Social Contracts Theory

ISO International Organisation for Standards

JADE Java Agent Development Framework

KIF Knowledge Interchange Format

LLD Language for Legal Discourse

NAICS North American Industry Classification System

NPE Norm-Based Process Expectations

NPE-L Language for Norm-Based Process Expectations

NPE-M Norm-Based Process Expectations Module

NPE-R Norm-Based Process Expectations Results

OASIS Organisation for the Advancement of Structured Information Standards

OCL Object Constraint Language

OCML Operational Conceptual Modelling Language

OED Concise Oxford English Dictionary

OMG Object Modelling Group

OPOSSum Online Portal for Semantic Services

ORB Object Request Broker

OWL Web Ontology Language

OWL-S Web Ontology Language for Services

PDDL Planning Domain Definition Language

QoS Quality of Service

QVT Query/View/Transformation

RDF Resource Description Framework

REST Representational State Transfer

SAWSDL Semantic Annotations for WSDL

SDL Standard Deontic Language

SLA Service Level Agreement

SLF4J Simple Logging Facade for Java

SOA Service Oriented Architecture

SOAP Simple Object Access Protocol

SOC Service Oriented Computing

SSL Secure Sockets Layer

SWRL Semantic Web Rule Language

SWS Semantic Web Service

TCP Transmission Control Protocol

TTF Time To Finish

UDDI Universal Description, Discovery and Integration

UML Unified Modelling Language

UN United Nations

URL Uniform Resource Locator

W3C World Wide Web Consortium

WS Web Service

WS-BPEL Web services for Business Process Execution Language

WSD-S Web Service Description Language with Semantics

WSDL Web Service Description Language

WSLA Web Service Level Agreement

WSMF Web Service Modelling Framework

WSMO Web Service Modelling Ontology

WSMX Web Service Modelling Execution Framework

WTO World Trade Organisation

XML Extensible Markup Language

Declaration of Authorship

I, Andrew K. Douglas , declare that the thesis entitled *Norm Based Service Selection* and the work presented in the thesis are both my own, and have been generated by me as the result of my own original research. I confirm that:

- this work was done wholly or mainly while in candidature for a research degree at this University;
- where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated;
- where I have consulted the published work of others, this is always clearly attributed;
- where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work;
- I have acknowledged all main sources of help;
- where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself;
- parts of this work have been published as: (Douglas et al., 2010)

Signed:.....

Date:.....

Acknowledgements

I would like to pay special thankfulness, warmth and appreciation to the persons below who made my research possible and assisted me at every point to believe in my goal:

My Supervisor, Dr. Gary Wills for his vital support and assistance. His advice and wisdom guided me and enabled me to turn a disparate set of ideas into the following academic undertaking.

My Second Supervisor, Dr. Robert Walters, whose help, sympathetic attitude and advice surrounding the written content of this thesis has been invaluable.

Eric Cooke, for his help in navigating some of the trickier formal procedures laid out by the University of Southampton.

Dr. Terry Payne for supervising my first year and helping me better understand my research themes.

To my army of proof readers, including; Rupert Thomas, Matthew Green, Hannah Carnegy, Simon Rains, Radleigh Tant and Robert Gullan.

My Mother and Father, without whom I was nothing; they not only assisted me financially but also extended their support morally and emotionally.

Finally, to my lovely wife Adele. For always being there to support and believe in me. This work would not have been possible without you.

Chapter 1

Introduction

This thesis represents a narrative of findings from research into service selection on the Semantic Web. The work has been inspired by previous work in the fields of distributed computing, the Semantic Web and agents. The following is a review of pertinent research on which this thesis builds.

1.1 Distributed Computing

Traditional computing paradigms have relied on single machines providing resources to users on a local basis. Increasingly, however, resources and applications are being made available remotely over distributed systems such as the internet. This move towards information and resource sharing has been driven jointly by both research and business communities; together striving to reduce costs by sharing expensive resources, streamlining existing business processes and opening up new avenues of trade (NCUB, 2013). Various hardware and software architectures have been proposed for distributed computing, most notably client-server (Sinha, 1992), peer-to-peer (Bolosky et al., 2000), grid (Foster, 2004) and cloud (Armbrust et al., 2010) infrastructures. The evolution of distributed computing platforms has in recent years led to notions that distributed nodes should be open, composable, reusable, interoperable and accessible using clearly defined interfaces. These concepts have lead to the consideration of distributed nodes as services (Huhns and Singh, 2005). In distributed computing services expose capabilities or resources for use over a computer network by a third party. This model follows the client-server infrastructure; a server provides a service and any authorised client can consume the service. This basic view of a distributed endpoint as a service scales well for both large systems and individual companies offering a single service. Many companies and businesses have already adopted service based architectures in order to reduce costs and increase efficiency; these include governments and councils (Oracle-Consulting, 2008), learning establishments (Al-Ajlan and Zedan, 2008) and healthcare

professionals (Hoskins, 2008). One of the main driving factors behind the industrial take-up of the service metaphor has been that potentially any resource in a business process (or workflow) can be exposed as a service. For any resource exposed through a service interface to be useful and aid in the accurate modeling of complex business processes, actors must be able to link services. This linking is referred to as service composition (Milanovic and Malek, 2004). In service composition actors are able to reuse existing services to perform new tasks which might otherwise have been impossible. The previously introduced infrastructure utilising services enables this model for composition since individual entities are exposed using reusable, interoperable and clearly defined interfaces. The composition of multiple services can be independent of underlying platforms or implementations and is dependant only on a common communication language. This interoperability allows companies to interact easily with other companies, rapidly assimilating external services into their own workflows. To enable the rapid production of compositions many workflow and process specification languages have been proposed (van der Aalst, 2003; Wohed et al., 2003). These languages allow business rules to be added to service compositions, in addition to speeding up the development of new systems. One of the key issues involved in service composition is service discovery. As with any potentially decentralised distributed system, service-based approaches suffer from the "connection problem" (Davis and Smith, 1983) which can render them useless if a requesting service fails to find an appropriate provider. Many solutions exist to solve this problem, including directories, matchmaking architectures and blackboards. Service discovery is covered in more detail in Section 2.1.2.

In describing and working with potentialy complex service architectures it is important not to conflate the notion of the service and any underlying workflow or business process which is exposed by that service. The research described by this theis looks at using process models to differenciate services. Process models are not processes to be executed, but specifications of the ways in which a client can interact with a service. They, by there very nature, may be seen as workflows. However, a service need not expose an entire workflow, and a process model should not be seen as a complete workflow, nor should a service be seen as a workflow. The solution proposed in this thesis to the research questions stated in Section 1.4 could arguably be used effectively to rate workflows of any type reagrdless of their being exposed by a service. However, this would not answer our governing question, and further dicussion workflows in a wider context falls outside of the scope of this thesis.

There have been a large number of proposals for service oriented distributed systems. In the 1980's several projects emerged including: DCOM, DCE and CORBA (Brodie, 2003). Of all of these projects, CORBA was the most successful, with a first concrete specification released in 1991 and a number of successfully deployed applications. However, CORBA failed to gain support from key vendors and thus development was slow and not widely adopted. It took nearly 10 years for robust large scale CORBA solutions to become

possible. One of the main proponents of the CORBA infrastructure has been the telecoms industry including most notably: Nokia, Lucent and Sprint. However, many of these implementations rely on bespoke or vendor specific solutions which limit the appeal of CORBA for a wider audience. A further look the lessons which can be learned from CORBA can be found in Section 2.1.4.

1.1.1 Service Oriented Computing

So far, this section has outlined the beginnings of and advantages to a service-based paradigm for distributed computing. Service-based approaches have been implemented in recent years using Service Oriented Computing models (SOC). SOC embraces the notion of systems built around loosely coupled sets of services which may or may not be distributed either locally or remotely. Services within a system derived from notions of SOC can be said to be operating within a Service Oriented Architecture. At the core of SOC and SOA is the idea that systems and services can exchange messages in order to complete tasks.

The use of services and SOC within computing is not a new idea. In the past mail servers had forwarding services and automated listing servers had sign-up services enabling clients to email in requests. What has changed (and is constantly evolving) is the industry's perception of the power of and uses to which services can be put. The current trends in computing are towards the realisation of SOC though the utilisation of Web services (WS). Current standards and ideas surrounding WS focus on research into and the creation of well-defined protocols to enhance the SOC model. WS are services described in a standard manner which enables straightforward discovery and invocation. By exposing potentially complex underlying functionality through standardised interfaces to clients with sufficient privileges, the WS paradigm aims to simplify access to data and create a more open and accessible Web.

It is important to note at this early stage, that modern services are not analogous to traditional computing objects. It is suggested that services are more than objects. Services expose potentially complex functionality and can be complex in both their construction and behaviour. Modern WS can be easy to describe, discover and invoke across multiple platforms, languages, locations and domains. Services in a modern SOC environment can be accessed, but not modified, or instantiated by a remote user.

1.2 Software Agents

Just as the service model enables companies to enact business processes and workflows, software agents allow actors to be modelled in systems (Wooldridge, 2002). Software agents are not a new idea. In 1992 agent-based computing was described as "the next

big break though" (Sargent, 1992). However, despite plentiful research into the area, there is still no single agreed definition of an agent. To quote two of the most influential agent researchers, Prof. M. Wooldridge and Prof. N. R. Jennings, "software agents are essentially software components acting on behalf of a third party, with or without intervention, through social interactions and reactions to external stimuli" (Wooldridge and Jennings, 1995).

Both Web Services and agent-based computing are high-level models of distributed computing (Moreau, 2002). There are a great deal of similarities between agents and Service Oriented Computing, in terms of objectives and issues. Agents could be said to be a broader field, which subsumes Service oriented approaches, except of course for the boundaries set up through standardisation of languages and ontologies. Service based approaches could be differenciated as concerning themselves with the external characterization of behavior and interactions, and somewhat less on internal architecture and characteristics of individual agents. Both of course have matchmaking, solving the search problem and composition as primary concerns. This research takes the view of services as simpler things than agents, but they can be used by agents (as clients).

1.2.1 Operational Environments

Agents and services can operate in many different distributed environments. At the transport level, infrastructures such as peer-to-peer and client-server can be used for message passing. Functioning independently from any network standards and a level of abstraction up from the transport layer, there are two general types of environments in which agents can exist: open or closed. Closed environments are characterised by a central authority of control, stipulating rules and standards with which all participants must comply. Electronic institutions (Rodríguez-Aguilar, 2001; Sierra, 2004) are classic examples of systems where agents participate in a closed environment for ease of management and to enable better modelling of real world company structures. Open environments have few such restrictions. Agents and services are free to communicate in any way with any other entity. Locating partners for business in an open environment can be carried out using a directory, broker, or middleman (Sycara et al., 1999). Systems in open environments can be quick to set up and potentially limitless. The largest example of an open computing environment is the internet. Potential data storage space is unlimited, as is computing power and the number of connected nodes (an argument could be made that a limit is imposed by address lengths, however, this is false and overlooks technologies such as routers).

However, Systems in open environments fall down if provided with insufficient information about other services or available agents. Actors involved in open environments can fail to find resources or can be forced into sub-optimal engagements with unsatisfactory providers (Davis and Smith, 1983). This coupled with a lack of central control or oversight

are the greatest problems facing developer in open environments. For service-based distributed computing to perform efficiently in open environments developers will need to overcome issues involving service discovery and oversight.

1.3 General Model for Service Selection

The scope of this work is limited to open environments. Closed environments such as Electronic Institutions can be assumed in most cases to be trustworthy with well-defined rules governing permitted and forbidden behaviours (Bou et al., 2008). In such systems, any issues concerning norms in tasks will be resolved before runtime. The following general model can apply to both open and closed systems, however, it is best suited to uncertain open environments. It is possible to understand service selection to be a subtask of service composition. Laukkanen and Helin (2006) identify four distinct stages involved in service composition, of which three of these stages are relevant to service composition: goal formulation, semantic matchmaking of services and the creation of a workflow (the fourth is the execution of the workflow and falls outside of the interest of this work). These three stages can be reduced to: planning, locating and preparing remote services. To carry out this process in an open environment, three actors are required: a provider to implement the required process, a consumer to utilise the service and a directory to allow the consumer to locate the provider.

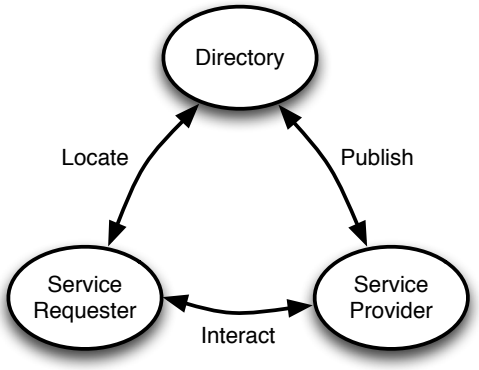

Figure 1.1: General Model for Service Discovery (this version adapted from Huhns and Singh 2005)

The model outlined in Figure 1.1 is often referred to as the general model for service discovery. A provider registers their service with the directory, the consumer then looks up the service in the directory and obtains any end-point information he/she might need to utilise this and the consumer then contacts the providers service (using this information). This structure forms the basis of all of the service discovery models covered in this work. Service selection is the evaluation of several services discovered to decide which is the most suitable for the current task. At this most basic level, service selection

may be carried out by any actor. This work will go on to investigate how service selection might be improved for the consumer. It is assumed that no selection is carried out by the provider and only basic semantic matching is carried out by the directory.

1.3.1 Limits of current service selection techniques

The general model for service discovery and selection provides a scaffold on which we can start to build the complex systems needed to overcome the "connection problem" (Davis and Smith, 1983) with regards to discovering and selecting partner services. Current issues surround the areas of privacy, trust, politics and the technical underpinnings of some of the applied technologies. Many solutions have been proposed, however, even more issues remain unsolved or with answers unproven. Section 2.3.5 will go on to further expand the issues outlined here and suggest where current research efforts fit in regards to fixing these and where current approaches may be lacking.

1.3.2 Normative Agents and Social Pressures

During this thesis, the concept of normalcy is introduced. Defined in Section 2.4 as "...potentially fluid concept[s], often context dependent, by which participants can be regulated". Norms as they relate to service selection can, by the wide variety of uses for services, take many forms and have many uses. For example in the use of governing which actors are permitted to carry out the edit process whilst utilising a publising service (de Moor and Jeusfeld, 2001). Norms are proposed as a novel mechanism to impose control on systems of autonomous actors. Distinct from rule or policies, norms are both pervasive in their nature and flexible in the degree of adherence required by members of a society. This thesis will explore the extent to which norms provided by a central authority differ from those that emerge from patterns of behaviour and will look at the topics such as internalisation of normative facts and reconciling of normative collisions. The current body of research available surrounding normative beliefs and normative agents was reviewed and will demonstrate that this area has received intense interest in recent years with the rapid expansion in the design and use of multi-agent systems. Providing one of the potential control mechanism to be used in the administration of open distributed environments, norms are rapidly being seen as core to many new agent based approaches as well as providing interesting parallels with "real world" dynamics during simulations or external interactions. This research also presents the perceivable disconnect between notions of norm-aware agents and traditional service based computing architectures.

1.4 Research Objectives and Contributions

This report aims to identify any gaps in the current state practices in service selection for open architectures. Furthermore, it aims to address any discovered gaps by proposing a client-based architecture for service selection. Research has been guided using techniques founded in the fields of the Semantic Web, modal logics and social commitments. The final aim of this work is to design a system capable of enabling a client to decide the best service to use from a set of semantically similar services. The work in this thesis has been based around a model of service composition where services are unknown at design time and selected shortly before they are executed while the client is running. Distinguishing semantically similar services can be difficult. Rules and an implementing logic are required for an actor to be able to order potential services. As such metric has been created for scoring services based on extrapolated data. There are a wealth of challenges in this field and numerous research gaps for exploration.

1.4.1 Research Objectives

The central research theme to this thesis comes from the two areas outlined in Section 1.3: Service selection and normative agents. Generally the research objective of this thesis can be identified as the following governing question:

> *"How can an actor with a set of known normative beliefs use these beliefs to aid service selection where IOPE matching typically falls short?"*

In carrying out this research, it is envisaged that the gap identified between normative agents and those generally operating with semantic web services can be narrowed. An agent capable of answering this question will need to have several key qualities: the ability to transfer (import) and represent norms, the ability to reason over norms with regards to expected behaviours of others and the ability to make distinctions between proposed partners based on the outcome of any norm based logic. This leads to the following additional investigations which will need to be resolved in order to fulfil the original research question.

- How can norms be represented so as to best relate to the proposed process models (workflows) of semantically aware actors?
- What is an appropriate mechanism for the reasoning of norms with regards to expected processes which best suits the selection of a partner service?
- What costs or bounds might a normative approach place on actors undertaking service selection?

This research is interested in the identification of tools and techniques created by the normative agent community which may be used to aid service selection in circumstances when conventional methods fall short. What follows will demonstrate that such circumstances typically occur when actors are selecting between services which are externally similar. By taking into account notions of normalcy and looking at expected processes at a more granular level differences between potential partners can be amplified.

This research will identify a disconnect between the Intelligent Agent and Semantic Web communities surrounding the use or norms. It is expected that by bridging this divide service selection for both can be improved. By bridging the divide, it is also anticipated that service selectors are aided by increased accuracy in the calculation of the possible cost of using a service and in general, service quality will increase as norms in use by actors indirectly affect individual services' ability to become selected and utilised.

It is likely that this research will intersect the fields of service selection and workflow selection as the process descriptions exposed by services can be seen as subsets of larger workflows. Section 1.1 touches on the interplay between process models used in service descriptions and workflows used in a wider context. It can been viewed that a solution to the research questions posed in this section might be the start of a wider discource as to the use of norms in workflow selection. However, broadening the context of this research to include the wider topic of workflows does not aid in answering the govering question and is out of scope of this thesis.

1.4.2 Solution Requirements

During this research a number of exisiting service discovery languages and techiques will be discussed. This research will show that they are currenly not sufficent to answer the research questions posed in Section 1.4. This lack of a suitable approach to define and reason over normative behaviors in proposed process models, is the driving factor behind the creation of a new model, ontology and module during this thesis. For this reason any final solution must provide the following:

- Provide a machine readable mechanism for the representation of norms to be used in reasoning over process expectations.
- Have ties to existing Semantic Web technologies so as to fit within the existing Web service stack (Section 2.1).
- Provide the ability to reason over both complex and simple sets of service based process model inputs.
- Consume over both complex and simple sets of normative inputs.

- Operate so as to enable the collection of data to answer the governing research questions posed in Section 1.4.
- Provide a set of input norms for the testing of any proposed solution.
- Identify or create a set of Web service descriptions to act as test cases for a service selection procedure. These descriptions should describe services which are similar enough that exisiting IOPE solutions would fail to differentiate between them, and yet where some expose processes which are not too be expected for any given set of normative beliefs.
- Identify appropriate methodologies to test any solution against the research questions posed in Section 1.4.

1.4.3 Research Contributions

The research contributions of this thesis can be divided into three main parts: a theoretical model for normative process expectations, a practical implementation of a service selection module and lessons learned while validating and testing. Focusing initially on the identification and creation of a model for the representation of and reasoning over norms with regards to processes relating to semantically aware services. The design will rely heavily on existing normative agent research so as to gain the strongest possible base on which to build. Conversely, the final selection system will rely heavily on research from the Semantic Web community; in particular notions of service composition and location twinned with well-formed description languages. The aim is to produce a system which can demonstrate how norms can be used to help improve existing service selection patterns where common solutions fail due to a lack of social context and granularity in approach.

Alongside this, it is expected that the testing and validation of an area of computing, which as been overlooked such as norms, will require a unique set of inputs. Any such inputs will need to be created. The process of creating any inputs will present some novel approaches to the generation of test cases for both services and norms.

In terms of solving the research objectives laid out in this section, the following contributions have been made:

- A model for reasoning over norm based process expectations and an associated ontology for the representation and storage (NPE-L) has been produced.
- An ontology for the representation and storage (NPE-L) has been produced. This has close ties to existing Semantic Web languages, as well as taking from the field of normative agent research.

- A module for the reasoning of norms with regards to expected behaviours as derived from semantic web services (NPE-M). This stand-alone module has been built to allow an implementing actor to reason over expected processes and produce results which can be used to influence services based on existing normative beliefs.
- A set of input services and a set of input norms produced in an manner intended to minimise bias.
- A framework for the testing and verification of normative service selection techniques.

1.4.4 Thesis outline

This thesis describes the research conducted towards and methodologies used to produce the research contributions outlined in Section 1.4.3. This work is structured so as to introduce the reader to new topics before utilising them to solve parts of the stated research questions. As such this thesis is presented using the following structure:

Chapter 2 provides an introduction to the topics surrounding this research and outlines some of the key areas on which this thesis will build. This chapter will also look to identify where the research questions originate and starts to build a response.

Chapter 3 uses ideas gained during the extensive literature review conducted in Chapter 2 to form an abstract model for normative facts with the flexibility to allow for reasoning over of expected processes.

Chapter 4 builds on this model of norms to form an ontology for the representation and storage of norms using techniques from the domain of the Semantic Web. This (along with the Chapter 3) satisfies the first of the research sub questions.

Chapter 5 presents a complete model enabling the reasoning over provided process models using sets of normative facts. This satisfies the second of the research sub questions and sets the basis for the validation of the primary research question.

Chapter 6 presents the testing methodology which has been used to take the services selection module created in Chapter 5 and use it to answer the governing research question.

Chapter 7 details the creation of a set of service test cases for use in validation. This includes a novel method for the production of degraded services based on an original service.

Chapter 8 augments the service test cases created in Chapter 7 and uses them to generate a set of norms for use in validating the research questions.

Chapter 9 presents the results of initial testing undertaken on the “service selection module”.

Chapter 10 outlines the techniques used to validate the system and answer both the final research sub question and the governing research question. Results here will demonstrate that the approach detailed in this thesis can aid service selection, but that in doing so some penalties have to be absorbed by implementing actors posing certain restrictions on use.

Chapter 11 provides a summary of the research conducted during this thesis and discusses future avenues for research.

Chapter 2

Literature Review

This chapter outlines some of the prominent work in the field of service based computing and service selection. It describes the key concepts involved in service selection for service based computing gives an account of why service selection is important and shows how the ideas and conclusions in Chapter 3 have been reached.

Firstly the overarching topics of distributed and service oriented computing are investigated. The intention is to form a solid based on which to build a more complex framework and to demonstrate that selection and composition of partner services is a key topic within the field.

In Section 2.2.2 notions of agency and basic agent structures consisting of intelligent socially aware actors are outlined. If would be entirely possible to create a model of selection which is devoid of any intelligence and purely functional. However, this research utilises the agent paradigm to give a platform on which the normative beliefs and techniques central to a norm based reasoning and section can be best articulated. Section 2.2.2 introduces agents and outlines many of the areas important in building a system of socially aware actors.

The remainder of this chapter focuses on notions of norms and the current state of the art in norm and normative agent research. This review will demonstrate that norms are essential for the regulation of behaviours in any social system, be it human or computational. This research will differentiate between the various sets of normative concepts and introduce definitions for all the key terms used in this thesis. Finally, a brief review of current research efforts in normative agents will be conducted to demonstrate who the key researchers are, what contributions they have made and what can be learned from their progress.

2.1 Service Oriented Computing

The SOC paradigm, built on the idea of SOA has attracted a lot of interest from both research and industrial communities. A simple search revealed there to be somewhere in the region of 29,800[1] papers investigating SOC, so this is obviously an area which is receiving a lot of attention. SOAs are based around the notion that remote computers can expose resources or capabilities to third parties using well-defined service interfaces (Huhns and Singh, 2005). The general model for SOC is that of a client-server architecture (consumer and provider). This model, along with the general model for service location/utilisation, has been outlined in Section 1.3 and Figure 1.1. As has been described above, there has been a great deal of interest into SOC by both industrial (with large companies such as Microsoft creating service-based products; Microsoft 2007) and research communities (Dustdar and Papazoglou, 2008). Adoption of these techniques has been widespread.

2.1.1 Web Services

Currently, the most promising standard for SOC is that of Web services. The support for this approach has been boosted by the ubiquity of TCP/IP computing and the growth of global computing networks such as such as the Internet. Remote capabilities can be made available to third parties over the Internet by services adhering to a few basic prescribed standards. An example of this behaviour is the Amazon.com Web services which allow developers to add book search facilities to almost any programme or website (Amazon.com, 2009b).

Figure 2.1 is the latest Web service stack to come from the W3C (Jacobs, 2008). There is currently no single agreed Web service stack. However, this relatively simple interpretation demonstrates nicely how groups of standards fit on top on each other in the Web service model. Most of the current standards have been developed either by the W3C or by OASIS (Márquez, 2008), the two main SOA standards groups. Most of their standards are written in XML (Bray et al., 2006). XML is a machine-readable markup format, intended to also be human editable. XML has enjoyed widespread support from industry and is now an integral part of most of the major programming platforms (Maharry et al., 2003; McLaughlin and Edelson, 2006). The main XML-based WS languages are SOAP (Gudgin et al., 2007) for communicating with services and WSDL (Christensen et al., 2001) for describing WS endpoints and message structures. Both SOAP and WSDL have received a large amount of praise from industry and have been successfully implemented by many organisations (Bloomberg, 2004; Microsoft, 2006, 2008). This standards-based approach has been critical in maintaining the success of the Web service paradigm (Kreger, 2003).

[1] Based on a search of the ACM digital library for papers and proceedings only (October 2015)

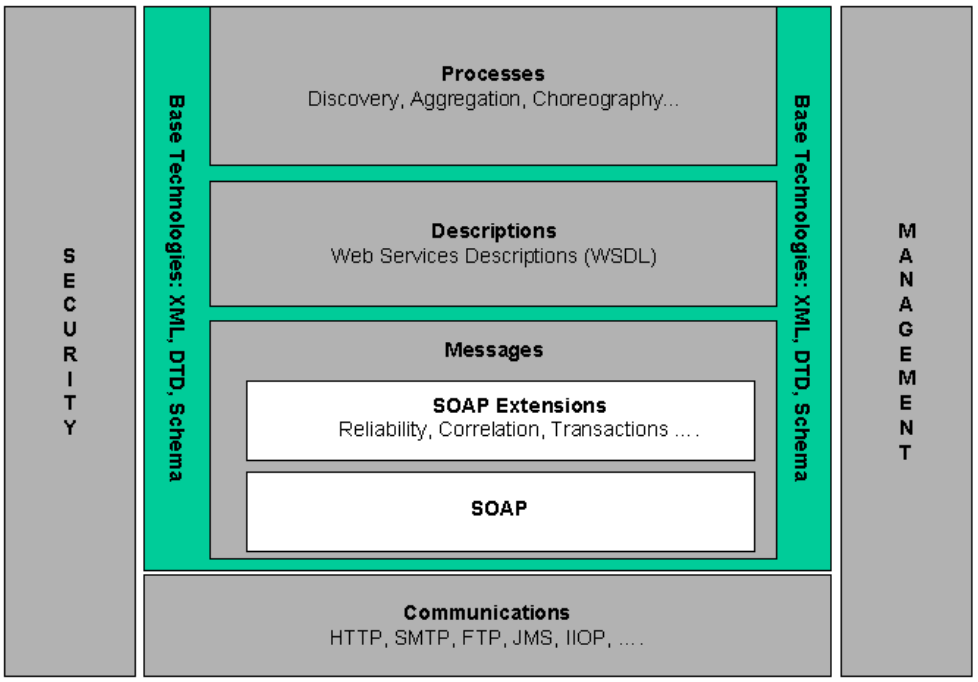

Figure 2.1: Web services Architecture Stack (Booth et al., 2004)

2.1.2 Service Discovery

As the demand for Web services has increased (Almeida, 2002), so developers have had to look more into detail at service discovery/selection. Even in relatively simple environments service discovery may be difficult. With the advent of ever more dynamic environments with participants which may enter and leave at any moment and resources which may be limited, the challenges surrounding service discovery and selection become much harder. Existing service architectures such as CORBA have devoted significant efforts into making service discovery available and comprehensive. If Web services are to succeed and become ubiquitous, efforts will have to be made to build on the successes of the past.

2.1.2.1 App Stores and API Marketplaces

In recent years, there has been a surge in the use of self-contained marketplaces for both software (in the form of bundled apps) and services. Whereas the growth of app stores has been driven by mobile technology and consumer need, growing consumption of remote services has been mainly to the benefit of enterprise (commercial) users. Many companies are purchasing software and services from remote end-points to fulfil business needs and solve operational problems (Money and Cohen, 2015). Examples of such forums include the Amazon AWS Marketplace (Amazon.com, 2009a), Microsoft Azure Marketplace(Microsoft, 2015) and the Mashape Marketplace (Mashape, 2015). All of

which allow users to both publish and consume services (often for a fee). Services may expose: software, compute resources or more complex platform and infrastructure solutions. Most of the discovery and selection process on these marketplaces is based on human interaction. However, work exists aiming at automating this process (Gonidis et al., 2015) and with further academic efforts being focused on utilising this field to aid research it could be that marketplaces are the beginnings of automated service discovery on a large scale (Boettiger, 2015).

2.1.3 Service Composition

One of the key features of SOC and the Web service architecture is that services should be reusable and composable. It should be possible to combine multiple simple services to form complex systems. Interoperable platform independent standards for communication make this (composition) possible. Services may be hosted on a variety of platforms, so long as they all use the same communication and workflow (or process) languages. In the field of distributed computing, there are two distinct views as to how distribution should be managed; choreography and orchestration. Both architectures allow remote services to be composed into more complex systems. The differences between the two lays in how the systems are controlled and which actors carry out the invocations.

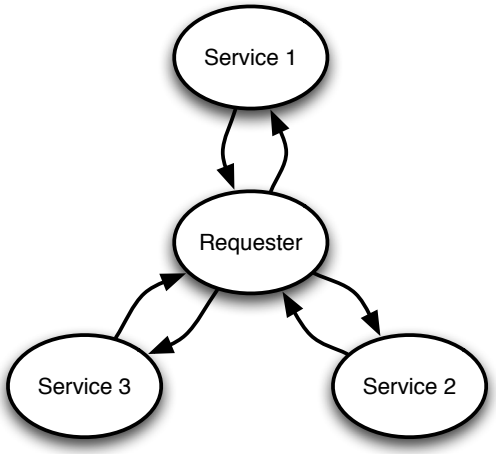

Figure 2.2: Web Service Orchestration

Orchestrated distributed systems feature one central point of control; for example a workflow engine as is used in the well supported Web service orchestration language WS-BPEL[2] (formerly BPEL4WS (Andrews et al., 2003)). This central workflow engine is used to run a workflow described in a given language at design time. All communication goes

[2]Developed by IBM, BEA Systems, Microsoft, SAP and Siebel Systems. Standardised by OASIS in 2003

though the orchestrator which can create a bottleneck, making Web service management, location and utilisation simpler but degenerating overall performance.

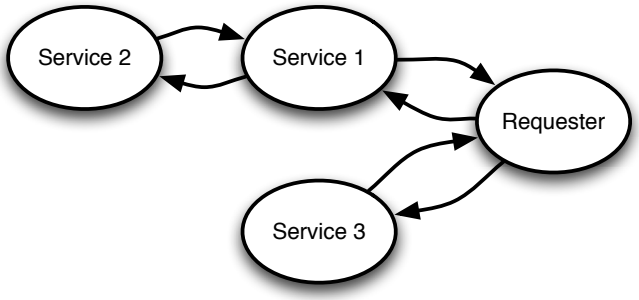

Figure 2.3: Web Service Choreography

Choreographies have no single central control system. Rather, workflows can be constructed at design time and agreed on by all parties, or created at runtime and implemented by one party. Choreographies are more complex than orchestrations and require a greater magnitude of trust between actors, but can reduce bottlenecks. This work will not concern itself with specific types of service composition. Realistically this work is aimed at helping to solve problems involved in service selection for both runtime choreographies and orchestrated sequences.

2.1.4 Learning From The Past

Section 1.1 introduced a number of distributed computing paradigms. Most notable among these was CORBA. In creating the Web service architecture developers should learn from its predecessors. There is little benefit to be gained by "reinventing the wheel", instead developers should take the successes of the past and build on them.

It could be said that there is a great deal of animosity between CORBA and Web service proponents. With the many supporters of the Web service paradigm labelling CORBA as failed (Gisolfi, 2001) and followers of existing CORBA systems regarding Web services as hype(de Jong, 2002). In truth CORBA and Web service can and should co-exist. CORBA can be seen as a lower level system less suited to the Web, but already established in telecoms and industry. CORBA provides greater support for state-full transactions and real-time systems required in many critical systems. Web services can be seen as "middleware for middleware" (Gokhale et al., 2002) better suited to the Web, easily deployable (as messaging can be routed through port 80) and with better mobility. The Web service community should learn from the experiences of CORBA. CORBA and Web services should be able to work together, not only through intermediary gateways (converting SOAP to IIOP) but also as a layered approach, working together towards a common goal.

In technical terms, the CORBA specifications can be seen as being more prescriptive than the collective Web service specifications. CORBA requires clients to have an ORB available through which clients can interact with other objects. During invocation an application will initialize the ORB and access its Object Adaptor. The Object Adaptor provides clients with generated class code stubs created by compiling IDL specifications into an OS and language-specific class for use by the client. At runtime, this code is then used to invoke a remote object transparently via the ORB. Table 2.1 outlines the differences between CORBA and Web services. It should be taken from this summary that although there are some significant differences between the two approaches, both attempt to create platform independent distributed systems capable of operating in numerous environments running any number of programming languages.

Aspect	CORBA	Web services
Protocols	IIOP, GIOP	SOAP, HTTP
Data Model	Object Model	SOAP Message Exchange
Client-Server Coupling	Tight	Loose
Serialisation	Built into ORB	User specified
State	Stateful	Stateless
Language Support	Any with an IDL binding	Any
Security	CORBA security service	HTTP/SSL and XML Signatures
Service Discovery	Interface Repository and Implementation Repository	UDDI
Firewall Traversal	Limited	HTTP (port 80)
Current Version	CORBA 3.0	SOAP 1.2

Table 2.1: Comparison Between CORBA and Web services (adapted from (Gokhale et al., 2002) and (de Jong, 2002))

2.1.4.1 Service Discovery and Selection in CORBA

In CORBA, applications invoke objects via opaque references. The processes involved are the same regardless of whether the object itself is remote or local. In Web services, services are referred to by URLs with any potential location information stored in DNS.

Service discovery in CORBA is based around the IR, on which applications can discover objects and their associated capabilities. Dynamic discovery and invocation of services is handled by first looking up the required object in the IR and then using the DII to

obtain IDL information and construct a request. This request can then be used with the appropriate ORB.

Service selection in CORBA is the responsibility of the client application. No attempt is made to conduct any form of service selection on the IR or ORB. The client selects the most suitable object from the IR.

2.2 The Semantic Web

First proposed by the inventor of the World Wide Web Sir Tim Berners-Lee, the Semantic Web has been touted by many as the next major evolution in computing. Sir Tim envisioned the Web with primarily machine-readable content, where machines reason over data to fulfil tasks for human consumers. "[The] Semantic Web will enable machines to comprehend semantic documents and data..." (Berners-Lee et al., 2001). Machines connected to the Semantic Web can share resources in an automated fashion enabling large interconnected knowledge bases to emerge. Complex queries over this shared data should be possible using federated queries or queries over agreed storage formats. Using these concepts, it is envisaged that the Web could evolve to a point where machines can carry out the tasks of finding, aggregating, sharing and even creating data.

2.2.1 Case Studies

Case studies for semantics Web and service based architectures use have been hard to find. However, this is rapidly changing as this technology becomes mature and used increasingly in industry.

2.2.1.1 Evaluating services

The ability to evaluate and compare services from multiple providers has been critical ever since humans started to trade. Consumers need to be able to find the most suitable provider; both to maximise any potential benefits and to protect against exploitation by "rogue" actors. During the mid nineteen-nineties, it was observed by researchers and industrial sources that computers could be used to search through millions providers rapidly and pick out a few suitable alternatives. The use of intelligent agent based systems in shopping was first used to any notable degree in 1995 by Andersen Consulting with their "BarginFnder" and later "LifeStyleFinder" products (Gonzalo, 2001). More recently websites such as Compare the Market[3] and Money Supermarket[4] have become successful, with the former turning over £248.1 million in 2014 (MoneySupermarket.com,

[3]http://www.comparethemarket.com
[4]http://www.moneysupermarket.com

2008). All of these comparison sites, whether they be old or new, operate in similar ways. They are either given information about available products by providers, or obtain information from publicly available sources (by "scraping" a providers' website (Kohavi, 2001), then an agent then ranks all the available options based on criteria given by the requester and returns a list of possible matches for the requester to evaluate. Most of these services are free to consumers and operate by charging providers for the provision of better ratings or higher visibility. With their increasing popularity and success, price comparison sites and shopping agents have been attacked recently for claimed bias and for lacking transparency in their business practices (Meyer, 2007).

With regards to the research questions proposed in Section 1.4, this thesis is only interested in the selection of technical services, or services exposed as processes by Web services. The previous examples from Gonzalo (2001), MoneySupermarket.com (2008) and Kohavi (2001) are still appropriate can be easily transfered to situations where selection might occur over a collection of providers with technical services.

2.2.1.2 Unassisted Selection

In the above study, a middle agent (the price comparison site) selects and rates a number of products or services for the requester. The final decision on which service to use is the responsibility of the requester. Ideally, however, if consumers wish to fully embrace the notion of an automated shopping agent, consumers would want the agent to make this final decision for them; a process this work refers to as "unassisted choice". The most well-known example of unassisted choice currently in use today is Google's[5] "I'm Feeling Lucky" button. When a user enters a search term into Google and presses search, Google searches its databases and returns a ranked list of pages (along with some ads and sponsored content). Pressing the "I'm Feeling Lucky" button performs a search and then returns the top ranked page (Google.com, 2009), bypassing Google's ranked list of results and adverts. This is a simple but powerful example of unassisted choice, if you believe that Google's search and ranking algorithms are reasonable (i.e. will return the most relevant page top), then the consumer should also believe that the first page is the best for them. Despite Google's reputation, it is clear that people do not yet trust unassisted choice in this situation. Historically, less than one percent of searchers click on the "I'm Feeling Lucky" button and for many years there have been researchers who argue that the button only remains to preserve the look of the Google front page (Newnam, 2007). However, recent developments in browsers and other software packages have seen other ways to use the "I Feel Lucky" emerging. Most notably in the Firefox and Chrome browser address bars, which enables users to enter search terms rather than URLs and if the term returns a strong enough result, be taken straight to the ranked page. This works especially well with company names and recognisable trademarks and

[5] http://www.google.com

has been popular. Simple as it may be, this case shows that just as page ranking was the next logical step on from basic search, so unassisted choice is the next logical step on from ranking.

Figure 2.4: Google's Main Interface (http://www.google.com)

2.2.1.3 Simple Web Services

There are a vast number of simple Web services being used by providers in industry to expose capabilities to consumers. Most notable among these is; the Amazon.com Associates Web service (Amazon.com, 2009a) for searching though Amazon's online store and catalogue Google Custom Search Web service (Google, 2009) for searching the Web and Ebay.com's Shopping Web service (Ebay.com, 2009). All of these services are offered free of charge and provide developers with an alternative method of accessing content from the provider. Many more services exist exposing data for use in third-party products.

2.2.1.4 SOA-Based Systems

Finding studies and information about entire SOA-based systems in use in industry is somewhat difficult. There are a wealth of examples of "theoretical" examples of SOA systems, the most well-known being the "travel agent" system, but SOA systems in use today tend to be proprietary. For example, it can be discovered that there are numerous local borough councils in the UK employing SOA techniques to reduce costs. These include Luton borough council, which with the help of Oracle has moved a significant amount of its information and services online (Oracle-Consulting, 2008). There are many financial institutions also benefiting from SOA techniques; many through automation of data entry and processing (Oracle-Consulting, 2009). Simple case studies showing the results of implementing SOA in industry are easy to find on-line. Revealing the techniques used is a harder task.

One of the most prominent and most thoroughly documented examples of SOA use in industry comes courtesy of Microsoft and QWest Communications International (Microsoft, 2008) (referred to in this thesis as QWest). QWest provides communications services to more than twenty-five million customers and has some fifty thousand employees nearly five thousand of which are developers. To aid communication and reuse of code among their developers, QWest decided to move forward with a Web service-based SOA framework. The aim of this solution was to allow QWest developers to improve testing and reuse code created by others. To facilitate this a number of UDDI registries (see Section 2.2.5) were created. When a team within QWest writes a new Web service (or wraps an existing functionality in a Web service interface), they publish it to a (UDDI) registry. When another team needs a specific new functionality, they can first search the registry to see if a Service providing that functionality already exists. If a service is found, the developer can add code in his/her project to invoke the service. This basic functionality is outlined in Figure 2.5.

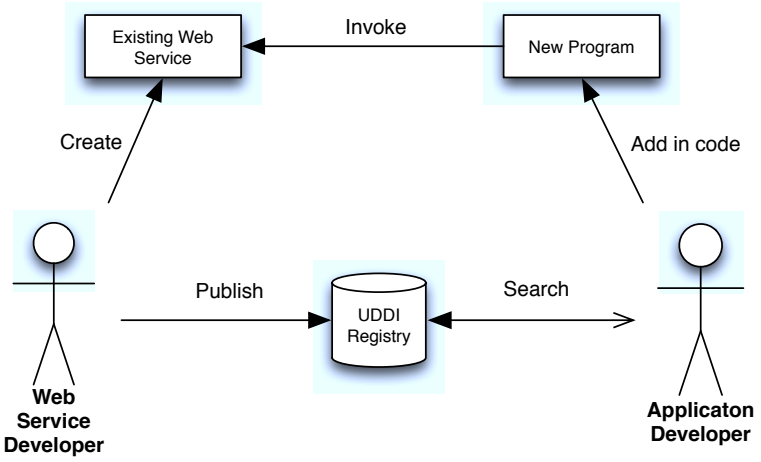

Figure 2.5: QWest Basic SOA Layout

To ensure that the quality of the services in the production registry is to a high standard, QWest implemented multiple UDDI registries with automatic migration of projects. The layout of the four registries and their purposes can be seen in Figure 2.6. Migration between registries is regulated and authorised by a member of QWests' development staff. When this person believes that the project has reached the level of maturity required to pass to the next registry he/she authorises the transfer. The UDDI entry is then serialised to XML and transferred to the new registry while maintaining any unique identifiers. This enables QWest to maintain a high quality of tested services in their production registry without service developers having to concern themselves with migration issues. This work will not comment any further on this technique, other than to say that it is an interesting solution to the problem of ensuring that consumers are provided with a high quality of service.

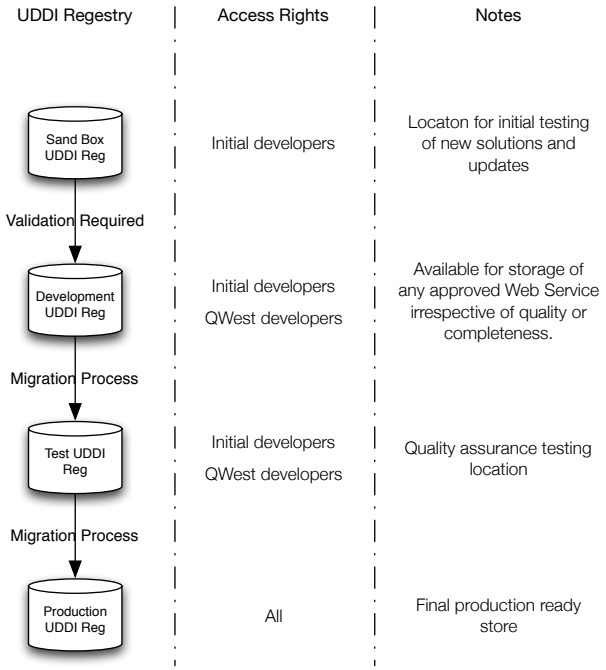

Figure 2.6: QWest UDDI Layout

2.2.1.5 Microservice Architectures

Microservices are services which follow a strict policy of single responsibility and are exposed by well-defined endpoints (Newman, 2015). These tend to be used internally in industrial applications and have experianced rapid growth in useage and maturity in recent years (Dragoni et al., 2016). In many ways, this pattern of use returns the industrial use of services closer to that of the original vision of services as considered by researchers. This trend has been exposed by the willingness of internal DevOps teams to share their experiences with microservice deployments. It is clear that internally many companies are using Microservices on an internal basis to scale solutions and connect resources (Thones, 2015). It may be that this has become easier as there has been an increase in the commercial availability of consumer APIs (Section 2.1.2.1). It is possible that it will soon be easier to find examples of widespread SOA infrastructures, as industry finds wider use for the paradigm.

2.2.1.6 Automated Selection

Finding companies who currently use agents to automatically select Web services for invocation is difficult. Very few developers have completely given over control of selection to automated processes mainly due to the scepticism that people have towards unassisted choice (as was presented in Section 2.2.1.2). However, these are areas where automated

systems have taken over planning and decision making; most notably in the planning and scheduling of tasks with multiple complex constraints and variables.

Scheduling cargo routes is a complex task, especially for sea-going cargo where the costs and loads are extremely large. In recent years many land based courier networks have implemented automated systems for the tracking and routing of loads, in 2005 Tankers International (one of the worlds largest crude oil carriers) implemented an automated system for cargo scheduling (T.Miller et al., 2005). The problems Tankers International (TI) were attempting to overcome were two-fold and primarily stemmed from the manual scheduling system in place prior to automation. First, TI wanted to reduce the number of scheduling errors caused by the complexity inherent in such a task. Second, TI wanted to capture the scheduling knowledge in an automated system so that expertise stayed with the company, irrespective of current employees. TI hired Magenta Technology[6] who designed an ontology for capturing the concepts and relationships required for scheduling and a system of multiple communicating agents, which would be individually responsible for making decisions about a different part of the system. By communicating and acting without outside assistance, this system of agents can quickly and effectively schedule TI's fleet of tankers.

2.2.2 Agents

Critical to the above example is the notion that the agent actor has a system capable of some form of unprompted decision making. Having been given a list of preferences, an agent system should negotiate with providers independently, return and potentially select the best possible solution. Such systems are referred to as software agents. agents have been extensively studied by researchers in the field of AI. Most notably Wooldridge and Jennings have classified software agents as programs which are: autonomous, sociable, reactive and proactive (Wooldridge and Jennings, 1995). That is to say that an agent should be capable of operating without outside assistance by communicating with other systems (agents or otherwise) and do so in reaction to or in anticipation of external stimuli. Agents form the basis of the agent-based software engineering paradigm (Jennings, 2000) and also of AI on the Semantic Web.

Section 1.2 described the similarities and differences between services and agents. To build on this initial set of distinctions, it is possible to use the above clasification of agents to describe services as well defined views (or accessors) on a resource (which could be an agent), where as an agent is a software program with all of the elements from Wooldridge and Jennings (1995). In particular, the notions of autonomy and sociality place agents apart from simple services (in computing terms). Such differences can be sing in the implementation, and design of agent and service based approaches (Dickinson and Wooldridge, 2005).

[6] http://www.magenta-technology.com

2.2.2.1 BDI Agents

One of the most popular architectures for agent design in recent years has been the BDI framework (Rao and Georgeff, 1995). A deliberative architecture loosely based around the folk psychology principles of beliefs, desires and intentions. In this approach, an agent's beliefs are a representation of its knowledge about its environment and its desires are goal states it wishes to bring about (states of its belief base). An agent's intentions are plans, by which it hopes to bring about a goal state. This approach has become almost standard in situations where agents are required to operate in highly dynamic environments, or with incomplete information about their environment (Dignum et al., 2000). As might be expected in this emerging field, there are currently many different implementations and versions of the BDI architecture. Despite their differences, the majority of these retain the three components above and operate using the general workflow outlined in Figure 2.7.

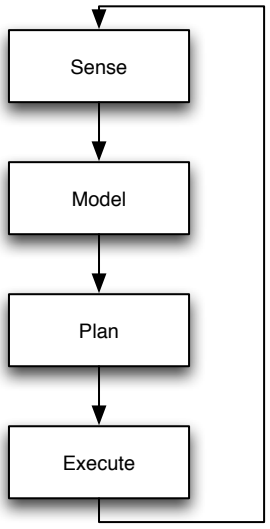

Figure 2.7: Basic BDI Workflow

BDI agents reason about the effects of their actions using an internal world model stored as beliefs. As a result of this, the BDI reasoning cycle starts with the agent sensing the surrounding environment and updating any belief model it has. Following this, the agent will re-evaluate its desires to see if any have been fulfilled, or if it is worth planning to try and fulfil one. Having settled on a goal state it wishes to obtain, the agent will produce a plan to work towards this goal (this plan may already be in existence). Finally, the agent will execute the next step of the current plan. This process repeats indefinitely (or until the agents desires are all fulfilled). Figure 2.7 is a simplified version of the BDI architecture. Many more complex versions exist, including concepts such as messaging (Bordini et al., 2007), obligations (Dignum et al., 2000) and complex planning (deSilva and Padgham, 2005). One of the strengths of this approach is that it allows designers to

implement agents with many different behaviours using one architecture by changing the desires and planning techniques used. This enables both proactive and reactive reasoning to be achieved.

2.2.2.2 Other Agent Architectures

There are may other agent architectures available to designers. For the purpose of this work the BDI model has been single out for investigation. For completeness, however, the most notable of the other architectures are outlined below.

Russell and Norvig (2003) defines an agent as consisting of "program and architecture", where the program must be suitable for a chosen architecture. The four basic types of agent program (defined in the following section) are a starting point from which the majority of agent systems are created.

Simple Reflex Agent This class is the simplest type of agent. Reflex agents are simple reactive programs which operate by responding to external stimuli. In agents of this nature, a response is selected from a prescribed set of condition-action rules by matching the current external stimuli to the activation condition on a rule. Specific rules must be created for specific environments. Although simple, this type of agent is powerful where proactive actions are not needed, or where all the possible rules required to reason over the world can be prescribed.

Model-Based Reflex Agents A simple reflex agent may be useful for very simple situations. However, in the majority of architectures this approach fails to overcome many important obstacles the most important of which is the notion that at any one time an agent will only be able to sense a small part of its environment. This is a problem referred to as "partial observability". To overcome this, model-based agents store an internal representation (or model) of the external world. When stimuli are sensed in the external world the agents' internal model is updated, then an appropriate response is selected (from condition-action rules). The advantage of this system over the simple reflex agent is that a model based approach allows an agent to take into account the state of the whole (known) world when deciding an action. The disadvantage is that the agents world model will need to accurately reflect the state of the surrounding world.

Goal-Based Agents For some agents operating in a purely reactive way by reasoning over the current state of the environment may be too restrictive. Agents may want to be proactive or react in a more reasoned manner. To do this an agents requires goal or end state information. Goal-based agents act to transform the external environment to a state closer to that of a given goal. Actions are decided by first planning all the

requirements (or a subset) to bring about the goal state and then undertaking this plan this task may be trivial (if only one step is needed to bring about the goal) or very complex (depending on the current state the desired state). Goal-based techniques are more computationally intense (as there is an extra planning stage), but enable more flexible agents to be created as action rules can be dynamic and created, destroyed and modified depending on the requirements of the plan.

Utility-Based Agents Simple goal based approaches may not be suitable for more complex situations. The main drawback is that although the agent knows the end goal state it requires, this state may be "too far away" in the short term, or may be dynamic. In these cases, it may help to have a measure of the pay-out/penalty which might occur when an agent moves from the current state to another. This pay-out can be referred to as the utility of moving to a state. Utility-based agents aim to maximise their utility (Kephart and Walsh, 2004). This may either be in the short term while moving towards some final goal, or in the long term as a goal itself. Utility-based agents operate by using a "utility function" to map a real number on to any future state. The agent then takes this into account while planning what action to take. This again adds more complexity, but also increases the flexibility of any produced agent (Gasser and Huhns, 2014).

Norm-Based Agents With the emergence of increasingly complex societies of agents, interest into agent architectures which can handle advanced social situations has grown. Adding concepts of norms to the notions of agency has become popular in recent research (see Section 2.4 for more on representing and reasoning over norms). Normative agents are based on any one (or a combination) of the previous types of agent. That is to say, that there is no single base normative structure for agents; normative agents simply add notions of norms to existing types. These norms are often represented as obligations, with a normative agent obligated to undertake an action at some time during its life-cycle. Numerous normative agent systems have emerged from recent research, most notably are systems based on BDI agent architectures (Boella and van der Torre, 2006b; Boella and Lesmo, 2001; Boella and van der Torre, 2008). Many of these add obligations to form BOID (belief, obligations, intentions and desires) based agent's (Broersen et al., 2001). These systems enable the inclusion of pre-defined or learn-able norms; often with the addition of an extra obligation base.

In most modern agent systems, a number of the above techniques will be applied. For example BDI agents, could be referred to as being model-based goal-based agents where the model is the agents belief base and the agents desires maps directly to goals.

Architecture	Motivations	Internal World Model	Pro/Reactive
BDI	Working over intentions to bring about the desired goal state	Belief base	Either
Simple Reflex	Reacting to external stimuli	None	Reactive
Model-Based	Reacting to changes in stored beliefs	Full	Reactive
Goal-Based	Planning to reach a final goal state	Optional	Either
Utility-Based	Aim to plan to maximise utility	Optional	Either

Table 2.2: Agent Architectures

2.2.2.3 Agent Toolkits

As the agent paradigm becomes increasingly mature, so methods to speed up development deployment and testing are required. Software packages which enable developers to carry out these tasks more easily by offering integrated design, coding, deployment and testing interfaces are often referred to as tool-kits. As agents and agent-based systems grow in complexity, so tool-kits become a necessity. Software packages, programs, or environments which offer agent builders sufficient levels of abstraction to allow them to more rapidly build agent based systems are becoming increasingly common. Such systems offer developers features such as visual programming, run-time testing, debugging environments and code reuse systems, all of which can be a great help. Agent specific tool-kits are needed in addition to regular OOP[7] approaches, as many existing OOP platforms do not support all features of agent programming.

There are many agent tool-kits available today both commercially and free from open source or other communities. There are tool-kits available using most of the popular programming languages and on most of the popular platforms. Specialised toolkits have been created to aid in the development of specific types of agent such as BDI (Bordini et al., 2007), mobile (Wong et al., 1997) and Internet agents (Boudriga and Obaidat, 2004). Among the most popular tool-kits for building type independent agents is JADE (Bellifemine et al., 2001), a Java based tool-kit. JADE enables developers to create multi agent systems by placing agents within containers (which may be located on the same or different systems), then linking multiple containers to produce a multi agent platform. For BDI agent development, the current state of the art is realised by the JASON tool-kit, a multi agent system, built on the AgentSpeak language. JASON enables the rapid

[7] Object-Oriented Programming

production of systems involving multiple BDI agents and may easily also link to JADE containers. For mobile agent design, JADE may also be used, although purpose built mobile-agent tool-kits such as Concordia (Wong et al., 1997) may be preferable. A summary of some of the most popular tool-kits can be found in Table 2.3.

Agent architectures as the basis for normative systems The normative agent frameworks and techniques which stand as precursors to this research have for the most part been designed around existing agent and multi agent methodologies and tool-kits. In the spirit of reuse and extension, common in computer science, many of the tool-kits noted previously have been amended by researchers to include normative concepts and abilities. A review of current notable normative agent frameworks can be found in Section 2.4.15. As an introduction to this and in addition to the above research on agent tool-kits, four agent frameworks have been identified as having been used in researching norms. These original forms are outlined below to give context to future discussions.

2APL (Dastani, 2008) is an extensive BDI based agent programming language and tool-kit with a standalone execution platform and eclipse plug-in. Its rich development environment and well defined BDI nature has led to it being used as a basis for numerous strands of normative agent research. Of particular interest is the separation 2APL places between concepts used to define the multi-agent system and the BDI concepts used to define individual agents. This split allows for complex agents to be easily placed in any number of environments. Agents' internal states are represented as beliefs, goals, actions, plans and events. Beliefs and goals are implemented in a declarative way, while plans and interfaces to external environments are implemented using an imperative programming techniques. This work has looked carefully at 2APL and has considered it usefulness in any final implemented system.

The SMART agent framework (d'Inverno and Luck, 2001) provides a way to specify individual agents and relationships between them. It is important to note that SMART is a framework and not a concrete tool-kit. It provides a way to specify agents and multi-agent systems but does not provide any form of solid constructive models. However, SMART does provide a clear model for the specification of agents and goal-based agent relationships. As such it has often been used as the basis for comprehensive agent toolkits where researchers can utilise the specifications to create well-formed implementations. One project of note to this research is actSMART (Ashri et al., 2005), which built on SMART to form a constructive model for multi-agent system utilising the AgentSpeak(L) language.

As well as the two previous BDI and goal based approaches, numerous role-based approaches are also worth noting. This is especially true as roles play a key part of normative systems. Role-based multi-agent systems provide a clear opportunity for further potential research into normative systems. The BRAIN framework (Cabri et al.,

2006) is an XML-based approach describing role by means of XML files. A second role-based model moves even further towards defining multi-agent systems as role-based systems; ALAADIN (Ferber et al., 2004) defines multi-agents systems as organisations with groups of agents fulfilling roles. This is an escape from the agent-centred view more commonly articulated and offers an interesting look at multi-agent systems as organisations of connected parties rather then individuals chasing goals.

Tool-kit	Language	Type	Notes
Zeus	Java	Multi-Agent	A somewhat basic tool-kit from BT. Open source, fast and easy to use with built in visualisation tools, code libraries and building tools.
JADE	Java	Multi-Agent	A mature and well supported tool-kit. Plug-ins available to enable use with Eclipse[8]. Allows the coordination of multiple FIPA-compliant agents.
METATEM	Prolog	Concurrent agent modelling	A temporal logic based language, which includes support for modelling multiple concurrently executing agents.
FIPA-OS	Java	Multi-Agent	Tool-kit to create FIPA[9] compliant agents. Comes with some examples of FIPA compliant agents.
SoFAR	Java	Multi-Agent Framework	Multi-Agent framework designed for distributed information modelling
PLACA	Lisp	Multi-Agent modelling	Modal logic based language building on the Agent-0 language.
JASON	Java	BDI Multi-Agent	BDI agent framework. Includes plug-ins to allow use with Eclipse and external ontologies[10]. Based on AgentSpeak with links to JADE.
Swarm	Objective-C, Java	Multi-Agent modeling	Agent based modeling simulation package, useful for simulating the interactions and emergent collective behaviours.
COUGAAR	Java	Multi-Agent modeling	Cognitive agent architecture for building large-scale distributed applications based on agents. Originally from DARPA.
Concordia	Java	Mobile-Agent	Queue manager based implementation of a mobile agent framework.

Table 2.3: Agent Tool-kits (adapted from Serenko and Detlor 2003 and Serenko and Detlor 2002 with additions)

2.2.3 Reasoning

Many SOC and agent-based approaches combine numerous heterogeneous distributed components to form complex and potentially dynamic systems. In such systems, components will be expected to represent knowledge, plan actions and react to external stimuli in a timely fashion whilest working towards a final goal. Logic based languages and formal methods can be very effective in the planning, implementation and verification of complex distributed systems (Mascardi et al., 2003). It is currently popular to model agents as intentional systems, many with mental-states allowing for the representation of BDI (Wooldridge and Jennings, 1995; Wooldridge, 2002). There are numerous logic languages that can aid in the design and implementation of BDI systems. Amongst the most popular are modal logics (Vardi, 1986; Gerbrandy, 1999), temporal logics, denotic logics and "The Situation Calculus" (McCarthy, 1969). There has been an amount of interest into the uses for logics in service selection and process management. Most notably research efforts have focused on a sub-field of modal logic epistemic logic and denotic logics. Epistemic logic is concerned with reasoning about knowledge. Research into this field has focused on creating and verifying workflows and compositions (Lomuscio et al., 2007). Denotic logic extends modal logic by adding permissions, obligations and prohibition. Denotic logic has typically been used to reason over legal systems. However, past research efforts have shown it to be appropriate for reasoning about ideal behaviours of complex systems (Mascardi et al., 2003).

2.2.4 Ontologies

Ontologies for the representation of classes of objects and their relationships can be layered. Ontologies provide the semantic support necessary for the provision of many of the notions of agency and AI. Making the processing and encoding of machine readable domain information simpler and richer (McGuinness and van Harmelen, 2004), ontologies conceptualise domain models, storing information for use by applications wishing to process the semantic content of any information (Gruber, 1993).

2.2.4.1 Ontology Design

For an ontology to represent a domain effectively it must be carefully designed. Designing a large ontology is not a simple task. There are five main approaches which may be used during ontology design (Holsapple and Joshi, 2002): inspiration (an individuals' view on the domain), induction (a specific case within the domain), deduction (general principles surrounding the domain), synthesis (using existing ontologies) and collaboration (several individuals' views on the domain). Once a developer has chosen from which perspective they will design their ontology, the next choice is what tool/design language to use to realise their design. The research relies on a combination of the Protégé tool

(Vendetti, 2009) and UML to provide the capabilities needed for ontology design. Both of these technologies have been used extensively and to great success in existing projects (Cranefield et al., 2001; Felicissimo et al., 2005). UML is a standardised language for developing software which enables the visualisation, specification and documentation of artefacts in software systems. There have been arguments which propose that UML is not expressive enough to be used during ontology design. It has been suggested that UML be augmented with other languages to provide additional constraints. OCL was initially paired with UML to aid the definition of constraints (Wang and Chan, 2001).

The Object Constraint Language OCL is a language for describing rules and limitations which apply to UML documents. OCL is a textual language designed to express constraints and rules which cannot be shown in the standard diagrammatic notations used in UML. OCL has been designed to be precise and to complement UML, specifically by adding notions of invariance, pre-conditions and post-conditions (Civello et al., 1998; Hamie et al., 1998). Previous to OCL, UML's only support for the specification of a model's properties and rules was through uninterpreted free text comments, or translating UML documents into a second separate logic based language (such as Z-specification). OCL is presented as a declarative programming language and may be converted to first order predicate logic for optimisation and proving (Beckert et al., 2001). Initially, OCL was proposed as an extension to UML. It is now a textual sublanguage of UML, part of the OMG UML standards set and the new QVT standards package. There have been criticisms of OCL, many pointed at its implementation-oriented approach (Vaziri and Jackson, 2000) and proposing Alloy as a replacement (Jackson, 2002). This work will be drawing on techniques and lessons derived from both OCL and Alloy.

2.2.4.2 Ontological Reasoning

One of the most powerful aspects of knowledge representation using ontological techniques is the ability to reason over data and infer new concepts. The level of inference available depends on how the ontology has been developed (Cranefield et al., 2001). In classical artificial intelligence research, frame-based systems or semantic networks were used to formalise Knowledge Representations. Building on this base there are currently two popular categories of ontology base: rule-based languages relying on predicate logic, leveraging existing well-defined semantics and powerful inference mechanisms and Ontology languages such as OWL, based on Description Logic (Herman, 2007). It is worth noting that when choosing an ontology language, ontology reasoning should come second to the most complex task of aiding design.

First Order Predicate Logic Approaches Rule-based languages grounded in predicate logic have the advantage that they build on a wealth on existing tools and research. Many standard first-order logic automated reasoning systems have become mature enough to be suitable for use in real-world applications. Automated checking tools are widely available and can also easily be manipulated to work with bespoke systems. First-order languages offer two main types of checking, notably: Concept satisfiability checking and subsumption checking. Among the most notable current predicate logic based languages is OCL which has been discussed above.

Description Logic Approaches Many ontology languages such as OWL and OWL-DL fall under the category of description Logics. Description logics have their history in frame based logics and semantic networks as noted above and can be transformed into first-order logic. Arguably description logics can be used to model certain knowledge related tasks in a more intuitive manner than pure first-order logics. Many automated DL reasoners (such as Pellet) offer efficient automated computation of classifications as well as a variety of consistency checks including: concept satisfiability checking, subsumption checking, knowledge-based satisfiability checking and instance checking (Buchheit et al., 1993).

Ontology Reasoners As interest in the World Wide and Semantic Webs has grown, so there has also been in increase in interest about ontologies and ontologically based systems. This has led to the evolution of many competing ontology reasoning systems. There are currently numerous ontology reasoning systems available, each utilising different reasoning and optimisation techniques. Benchmarks of a few different reasoners can be found in Lee et al. (2008a).

Currently, one of the most prominent reasoners is Pellet (Sirin et al., 2007). Pellet is a freely licensed Java-based reasoner supported by tools such as Protégé (Vendetti, 2009) offering complete OWL-DL reasoning, text and SPARQL parsers (Prud'hommeaux and Seaborne, 2008), as well as numerous knowledge-based interfaces. The core of Pellet is a DL reasoner based on tableaux algorithms (Horrocks and Sattler, 2001; Kolovski et al., 2006). The core components of Pellet can be seen in Figure 2.8.

2.2.4.3 Data Storage

Elements of data having their domain and underlying semantic meaning defined formally in an ontology can be represented in many ways. One common method is to use RDF and XML. As RDF and XML are both core technologies on the Semantic Web (see above), this encoding has rapidly become the "standard". Storing data as RDF+XML has its advantages (see Table 2.8) and is especially suited to the transport of data across

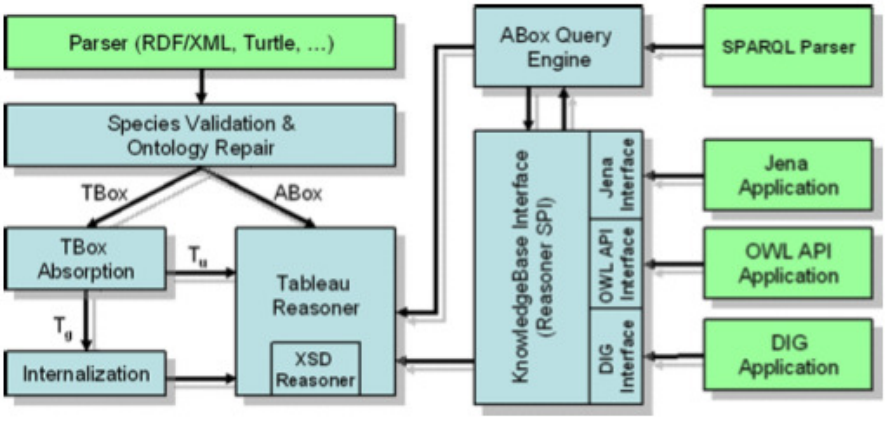

Figure 2.8: Main Components of the Pellet Reasoner (Sirin et al., 2007)

the Web. However, for the storage of large amounts of complex data a more advanced solution is required. For such situations, triple stores are often used to store data as RDF triples (see Section 2.4.8). The ontology data model can be applied to the data to provide semantic meaning. There are numerous commercial triple stores available on-line, a selection of which have been reviewed in Table 2.4.

Product	Pros	Cons
Jena	Well know and supported. Java based. includes rule-based inference engine.	Slow queries and missing web framework.
Sesame	popular with semantic web community. Well documented. Modular.	Can be complex.
Boca	Easy to install. Impressive features; client support, off-line persistence with replication, notification, access control, versioning.	No reasoning support.
Open Anzo	RDF quad store. Good feature support	Some performance issues.
Mulgara	Java based. Scalable. Transaction safe.	Complexity issues.

Table 2.4: Triplestores

2.2.5 Standards and Proposed Languages

Berners-Lee et al. (2001) proposed a Semantic Web where machines communicate in an open fashion using standardised languages and techniques (Berners-Lee et al., 2001). Many of these standards rely on the already dominant XML and RDF specifications.

2.2.5.1 Service Discovery

Many technologies have been designed to enable Web service discovery. From fully brokered approaches, through to open directories and every variation in between. The dominant technology in this field is UDDI. UDDI offers developers a simple structure on which their Web services can be published, found and invoked. UDDI's flexibility, extensibility and stability has enabled researchers to extend UDDI registries by adding new features and technologies, driving popularity in bespoke and research applications if not in general useage (Tonkin, 2005). This work will not focus any further on service discovery (as it is interested in service description and selection). However, this research does acknowledge that Web services are likely to have been selected from a registry before processing.

2.2.5.2 Service Description

Of far more relevance to this work is the field of service description. Service descriptions are required for automated service discovery, selection and invocation to be enabled. With extensions (Luo et al. (2006); Ambrosi et al. (2005); Powles and Krishnaswamy (2005); Nawaz et al. (2007); ShaikhAli et al. (2003)) UDDI can provide for the publishing of described (or anotated) Web services (and it could be said), that WSDL allows services to describe their own interfaces. However to enable truly automated discovery and selection, more expressive and semantically aware description languages are required.

Numerous service discovery languages have been suggested in previous research works. A few of the most prominent have been listed in Table 2.5, followed by an in-depth look at OWL-S which is the description language chosen for use in the rest of this thesis.

In choosing a language for semantically marking up services, there were several key requirements; primarily that it should enable the automated composition of services through reasoning over preferences and abilities. Secondary to this was that it should build on existing mature specifications such as WSDL and UDDI. There are a number of approaches other than OWL-S, most notably IRS-III and WSMF/WSMO (Domingue et al., 2008). IRS-III is a holistic Java framework for publishing, composing, locating and executing semantic Web services. It was rejected for use in this research due to a lack of support and because it would tie any future system into one single framework.

A number of projects have also been started aiming to augment WSDL with semantic information. These efforts have the advantage that they all build on WSDL, a mature and well supported language. Notably the WSDL-S project has produced a technical note (Akkiraju et al., 2005) detailing how WSDL might be augmented. This has lead to SAWSDL which has been created by the W3C "Semantic Annotations for WSDL" working group. SAWSDL (Kopecký et al., 2007) builds on WSDL-S by adding links to OWL-S. Both projects should be commended for their use of existing technologies and open designs. However, they each lack the support and maturity that OWL-S has gained.

WSMF and WSMO have received a great deal of interest in recent years, but do not have the level of research or tooling support of OWL-S (Schröpfer et al., 2007). A brief comparison between OWL-S and WSMO has been included in Section 2.2.5.3. One of the key concepts of WSMF and WSMO are mediators. WSMO architecture is that the goal, web service and ontology components are linked by four types of mediators; mediators linking ontologies to ontologies, mediators linking web services to web services, mediators linking web services to goals and mediators linking goals to goals (Cabral et al., 2004). Mediators are provided by WSMX to resolve heterogeneity problems on data and process level (Cimpian et al., 2006). Meditors are actors which can facilitate interaction between components such as services. This allows elements with missmatched data or processes to be joined to make complex processes, so as to solve complex probles.

WSMO-Lite is inspired by the WSMO ontology, focusing on a small subset of it to define a limited extension of SAWSDL Vitvar et al. (2007). The goal of the SWMO-Lite project is not to create new web syntax for logics, but to show how existing languages might be reused for descriptions of some essential parts of services such as functional and behavioral features (Vitvar et al., 2008). WSMO-Lite aims to simplify the semantic description of services through reuse of exisiting languages and relaxing the requirement on the completeness of any document (Kopeckỳ and Vitvar, 2008). The existance of WSMO-Lite is acknowledged here, however, in order to produce a solution to the governing research question in Section 1.4 a language with complete and formalised process definitions will be required.

Two of the most popular languages for the description of Web service process models; WS-CDL (Kavantzas et al., 2005) and WS-BPEL (Alves et al., 2007). Although these languages are not strictly service description languages, the power they provide developers to define workflows and processes makes them useful for this work. The OWL-S process definitions are more akin to the orchestrated workflows found in WS-BPEL than the choreographies found in WS-CDL. However, both are relevant as they each demonstrate how complex interactions may be modelled. This work is grounded in the notion that norms can be defined for steps in complex interactions. Each of these languages offers an interesting representations of business processes and inter-service interactions which could be used to prescribe behaviours of a service for interrogation by a norm-aware system.

Spec	Current Version	Status	Base	Service Def.	Process Def.	Grounding
WSDL	2.0 (06-2007)	Standard	XML	Some	No	Yes
OWL-S	1.2 (12-2008)	Member Submission	OWL/ XML	Yes	Yes	WSDL
WSMO	1.3 (10-2006)	Member Submission	WSML/ XML	Yes	Some	Not Predefined
WS-CDL	1.0 (11-2005)	Recommendation	XML	Some	Yes	Some
WS-BPEL	2.0 (04-2007)	Standard	XML/ BPEL	Some	Yes	Some

Table 2.5: Web Service Definition and Description Languages

The choice of OWL-S was motivated by its history and by its ability to fulfil all of the requirements for this work. OWL-S was finally chosen for its connections to the existing standards WSDL and UDDI (an OWL-S service grounding is based in WSDL and work such as Luo et al. 2006 have demonstrated that OWL-S can easily be added to UDDI) and because of exisiting research linking it to process selection (Polleres et al., 2005; Ganjisaffar et al., 2006; Klusch and Gerber, 2006; Lomuscio et al., 2008; Cuzzocrea et al., 2011; Paulraj et al., 2012; Vidal et al., 2012). Originating from DAML (McIlraith et al., 2001), OWL-S is arguably the oldest language for the true semantic mark-up of services. Having gone though three previous incarnations (DAML, DAML+OIL and DAML-S) OWL-S is currently standardised by the W3C and enjoys tool support from many companies (including the Stanford Centre for Biomedical Informatics Research with Protégé).

OWL-S is an ontology for semantically describing services on the Semantic Web. The aim of the OWL-S project is to enable the automatic discovery, invocation and composition of services. It achieves these three aims by allowing the semantic marking up of services along with detailed grounding instructions. The OWL-S ontology comprises of three separate parts (Figure 2.9) a service profile describing what the service does a process model describing the interaction required to fulfil any given goal and a WSDL grounding specifying service endpoints and connection methods for the process. Current approaches to service selection and handling using OWL-S focus primarily on the inputs, outputs, preconditions and effects described in the service profile.

One of the primary concerns with OWL-S is the disjointed approach taken when linking the service model and profile model. The creators of OWL-S argue the profile should primarily be used for service selection by a matchmaker and that the service model can then be used to determine how an end user agent should interact (Martin et al., 2004). However, it is the opinion of the authors that the two should have some form of connection as the inputs, outputs, preconditions and effects outlined in the profile should be backed up by, or equate to, information given in the service model.

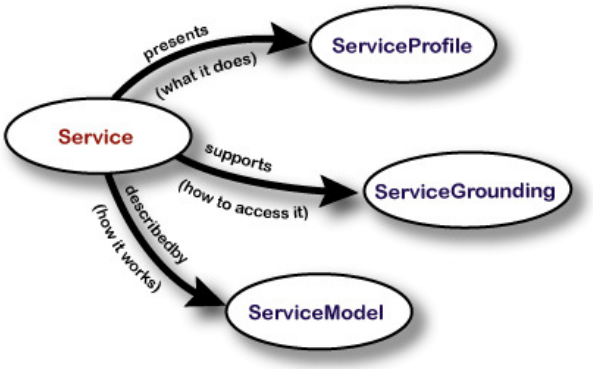

Figure 2.9: OWL-S Top Level Ontology (Martin et al., 2004)

Hypertext and Web API Descriptions Since the evolution of Web 2.0, providers have started to provision RESTful API based endpoints to expose resources. APIs documentation is typicaly only described as text in HTML documents. The lack of machine-readable API descriptions affects the feasibility of semantic annotation and the application of Semantic Web Service automation technologies to the field of RESTful services (Kopeckỳ et al., 2009). Thus research efforts such as MicroWSMO (Kopeckỳ et al., 2009), RESTdesc (Verborgh et al., 2012) and Hydra (Lanthaler and Gütl, 2013) have attempted to enable the creation of semantic RESTful services by structuring service descriptions and associating semantic annotations. This work is acknowledged and but seen as separate from the more holistic semantic Web service efforts which by there nature provide access to more complex resources and enable increasingly complex processes to be modeled.

Issues With OWL-S OWL-S is far from perfect. However, it is the opinion of the authors that it is the most suitable solution for providing semantic mark-up of services. This said, before choosing to support a solution it is important to recognise its failures as well as its successes.

One of the largest issues surrounding the current OWL-S submission is the implementation of the process model. It has been suggested that it is currently insufficient for use in representing complex processes or procedures Andonoff et al. (2005); Krummenacher et al.

(2010). As the process model is designed to enable developers to model exactly what their service does and how it might achieve any required outcomes, this is a critical fault. Research exists questioning many areas of the process model, of most note is the charge that the OWL-S model is unsuitable due to its lack of variables (Balzer et al., 2004). This effects developers abilities to bind variables (IOPEs) and to utilise conditions fully (by reasoning over bound variables). This work acknowledges this issue, but believes that they are overcome since OWL-S relies on external languages and standards to handle complex issues of conditions (such as SWRL). Of more concern is the lack of coupling between the three OWL-S models. There is no check to make sure that the three match, or that IOPEs defined in the profile have to correspond with the process. This is a complex and important issue and one that the authors believe should be pursued further (However this does fall outside of the scope of this thesis).

The lack of exception handling in OWL-S has also been criticised (Cambronero et al., 2009). Process languages such as WSCDL and WSBPEL contain support for complex exception handling; as does WSMO. This is useful for constructing fail-safe products. OWL-S has no explicit exception handling mechanisms, potentially because it is not a fully fledged workflow language. This could be seen as a disadvantage. However, passing the definition of exception handling mechanisms to the service producers (who can expose them as behaviours) allows for simple and flexible processes to be created.

There is also a lack of temporal data in OWL-S. This is included in many other languages and is useful for composition and real-time systems. This deficiency has been realised by the community and attempts have been made to fix it (Pan and Hobbs, 2004). It may also be possible to add time to OWL-S by importing the OWL time ontology, which now exists and can be used for conditional, compositional or real-time definitions.

A last criticism of OWL-S is that its creators have been somewhat vague as to how it might be implemented. There are a number of test cases available, however the majority of these are very poorly documented and there is no single example showing how developers might implement OWL-S from definition to search, location and execution of a service.

This is by no means an exhaustive survey of all the criticism which has been levelled at OWL-S. As with any emerging standard it has come under a large amount of scrutiny and attracted both praise and censure. This work acknowledges the faults with OWL-S and stands by the decision to use it as a base language for the reasons stated in the previous section.

2.2.5.3 OWL-S and WSMO

Put simply OWL-S and WSMO are ontologies (in the case of WSMO a framework in which ontologies can be created) for semantically describing Web services. Currently

OWL-S and WSMO are member submissions to the W3C, meaning that have been developed externally from the W3C and submitted for further consideration. They both cover roughly the same domains and both rely on underlying ontology languages for representation (Ankolekar et al., 2004). In the case of OWL-S, this language is OWL and in the case of WSMO the language the Web service Mark-up Language. The major features of these technologies are outlined in Table 2.6. The most significant two asymmetries between the two languages centre around their underlying structure and representation of individual Web service types. The first difference can be summarised as: OWL-S explicitly defines a set of ontologies that support reasoning about Web services, whilst WSMO defines a conceptual framework within which any ontologies can be created. In practice, this means that OWL-S defines well formed choreography and grounding specifications indicating advanced maturity (de Bruijn et al., 2005). The second major difference is that OWL-S defines a single central service concept and WSMO defines goals (for the representation of preferences), services (for the representation of capabilities) and ontologies. Linking between different WSMO elements to enable interoperability between heterogeneous components are Mediators (Fayçal and Mohamed, 2011; Domingue et al., 2005). Both OWL-S and WSMO have proponents in the Semantic Web community, each camp convinced that their technology is superior. It is clear that currently OWL-S has achieved greater maturity through the definition of the process model and the grounding of Web Services (Polleres et al., 2005). However, only time will tell which will garner greater support and if either will become a standard.

Aspect	OWL-S	WSMO
Ontology Language	OWL	WSML
Purpose	No focus on concrete application domains	Focused goal, specific application domains
Coupling	Tight	Loose (independent definitions of description elements)
Core Concepts	Service	Goal, Service, Ontology
Extensibility	Limited to OWL sub-classing	Extensible
Registry	None	None
Preferences and Capabilities	Not separated	Modelled independently and linked via mediators
Grounding	WSDL	No predefined grounding
Orchestration	Some limited, but well defined dynamic orchestration	Under-defined
Logic Language	Not defined (but can use SWRL, DRS or KIF)	WSMF (F-Logic)

Table 2.6: OWL-S and WSMO Compared (adapted from Lara et al. 2004 with additions)

2.2.5.4 SWRL

SWRL, maintained by DAML and formally submitted to the W3C in 2004, forms the basis for a standard rule language for the Semantic Web. Rule languages are seen as the next logical step in the evolution of the Semantic Web (Horrocks et al., 2004). Historically there have been a few research efforts aiming at integrating description logic systems with rules (Parnas et al., 1984). SWRL takes this research a step further attempting to describe relationships between concepts in highly expressive description logics such as OWL-DL. This is achieved through an approximate union of horn logic and OWL. The resulting language retains all of the expressiveness of OWL-DL and remains easy for both humans and machines to interpret. SWRL is not the first rule language (knowledge representation language) to be proposed. This work also acknowledges the existence of KIF (Ginsberg, 1991), PDDL (Ghallab et al., 1998) and many others as alternatives to SWRL. However, this focus of this the is on SWLR for the reasons outlined in Section 4.2.2.4. Recent work towards SWRL has stalled as developers have focused on the uses for SWRL (in OWL-S for example). However, tool support exists in the form of a number of reasoners and apis with SWRL support.

Rules in SWLR are formed of a body and a head. Essentially any rule should read that when the conditions in the body hold, so the conditions in the head should also hold. An example of such a relationship can be see in Figure 2.10 taken from the W3C SWRL specification. Here the rule combines the properties hasParent and hasBrother to imply the property hasUncle. Thus, if x2 is x1's parent and x2 is also x3's brother, then x1 has x3 as an uncle.

```
<ruleml:imp>
  <ruleml:_rlab ruleml:href="#example1"/>
  <ruleml:_body>
    <swrlx:individualPropertyAtom  swrlx:property="hasParent">
      <ruleml:var>x1</ruleml:var>
      <ruleml:var>x2</ruleml:var>
    </swrlx:individualPropertyAtom>
    <swrlx:individualPropertyAtom  swrlx:property="hasBrother">
      <ruleml:var>x2</ruleml:var>
      <ruleml:var>x3</ruleml:var>
    </swrlx:individualPropertyAtom>
  </ruleml:_body>
  <ruleml:_head>
    <swrlx:individualPropertyAtom  swrlx:property="hasUncle">
      <ruleml:var>x1</ruleml:var>
      <ruleml:var>x3</ruleml:var>
    </swrlx:individualPropertyAtom>
  </ruleml:_head>
</ruleml:imp>
```

Figure 2.10: An Example in SWRL(taken from Horrocks et al. 2004)

2.3 Service Selection

If the goal of OWL-S on the Semantic Web is to enable the automatic discovery and invocation of distributed services then a key component of this process is service selection. In the general model of service composition (Laukkanen and Helin, 2006), outlined in Figure 2.11, service selection falls in the semantic matching phase of the service invocation workflow:

- Requirement identification;
- Semantic matching of services;
- Process creation;
- Execution.

As with service composition, service selection can occur at several different times during the design and invocation of a system (Yang et al., 2003). Fixed selection occurs at design time with partners being hard-coded by developers. This method might be fast and relatively stable, but may lead to a poor allocation of resources (Milanovic and

Malek, 2004). Semi-Fixed selection occurs when a single partner (or short-list of partners) has been selected at design time, but no grounding has been set. In this model, the grounding is collected and bound at runtime. The most efficient allocation of resources can be achieved by explorative service selection. Clients exploiting explorative approaches discover, select and invoke third party services at runtime by evaluating their current intentions and the state of the world in which they exist. However, this approach can fall short if participants are provided with incomplete or incorrect information about the available services. A discussion of the issues surrounding service selection will address all of these problems in Section 2.3.5.

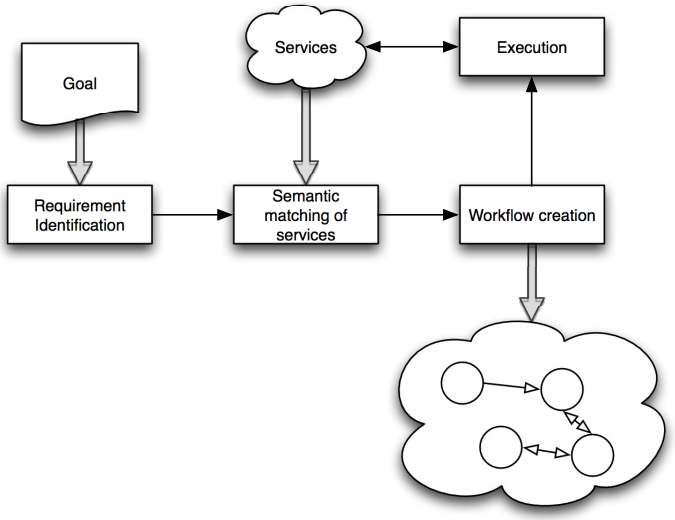

Figure 2.11: The General Model for Service Composition

2.3.1 Enabling Technologies

When studying composition processes in distributed systems it is important to differentiate between service selection and service discovery. Service discovery is the locating and connecting to potentially useful services. Service selection is the process of choosing which service to interact with from a set of discovered services. Service selection is necessary for sets of more than one discovered services, or where the choice of one service, over any other discovered service, may be sub-optimal. In recent years, the majority research into this field looked at solving the "connection problem" (Davis and Smith, 1983) for service discovery. Currently, there are well-established technologies including UDDI with WSDL for service advertising and discovery. The majority of research in the field of service selection has been focused at combining the processes of discovery and selection into one task and giving it to a middle-agent (Decker et al., 1997). These intermediaries come in three types: blackboard (client advertising), matchmaker (provider advertising) and broker. There are many systems available providing these three models. Many of the

available systems are semantically aware and ready for automated use on the Semantic Web (Sycara et al., 1999; Gagnes et al., 2005; Karande and Kalbande, 2014).

2.3.1.1 The Role of OWL-S in Service Discovery And Selection

OWL-S is an ontology of services. Used to aid the description of semantic Web services in a manner which enables their automated discovery, selection and invocation. Within the current service discovery framework of UDDI and WSDL, OWL-S can be added to enable the expressive description of services. The richer descriptions of services provided by OWL-S, along with advanced technologies such as brokers and middle-agents, would enable a greatly improved service discovery system. Service selection is currently not fully handled by UDDI and WSDL. Consumers are forced to rely on information from third party or supplementary sources, as the grounding in WSDL, lacks expressiveness. OWL-S potentially provides a greater level of detail as to the inputs, outputs, preconditions and effects of any service, thereby enabling more accurate service selection.

Representing and Reasoning Over OWL-S Processes For actors such as agents to be able to utilise OWL-S descriptions effectively to aid in service discovery, they must be able to internalise the data contained within. This can at the most basic level be carried out using simple term extraction, or at a broader level by using a concrete API to access all parts of the model (Sirin and Parsia, 2004). However, to enable actors to reason over the proposed actions within the process model, a further step is required: producing a traversable model of the expected processes.

There is currently no single method for the generation of a structure for the representation of an OWL-S process model which has been mutually agreed by the community. Suggestions have often come as by-products from the solutions to other issues. There does seem to be a consensus that a graph or net structure is needed, but no single method has garnered more support than any other. Brogi et al. (2009) best describes an approach which others take (Miao et al., 2008; Narayanan and McIlraith, 2003), in modelling OWL-S process models as Petri nets. In doing so the deficit of tools for the analysis of OWL-S process models can be overcome through the use of the many tools available for the manipulation of Petri nets. Another popular approach is to convert the process model into one of any number of graph structures. Of note among the research efforts using this approach, Barakat et al. (2012b) converts either OWL-S or WSDL requirements to directed graphs which they term “Plan path graphs” (sequential graphs), whilst Cuzzocrea et al. (2011) represents composite processes as graphs (similar in many ways to Petri Nets and enabling composition and complex control constructs). Finally, Hashemian and Mavaddat (2005) offers a novel approach whereby the process model is converted into a finite state machine style state-based model; an interface automata (this last approach is not only the most novel but potentially interesting of the graph

based efforts, as it not only enables composition but also usefully focusses attention on the inputs, outputs and ordering of actions performed by the process model).

2.3.2 Current Service Discovery/Selection Techniques

There is a significant amount of research being conducted in the areas of service discovery and selection techniques. The majority of existing work extends UDDI adding new features or information. The following are notable extensions to UDDI. Each adds a new facet, enabling richer service discovery/selection. There has been an effort to add OWL-S processing to UDDI, Luo et al. (2006) enables actors to semantically search for OWL-S information in a UDDI registry by resolving all ontological relationships before publishing the profile. Ran (2003) adds QoS data to UDDI allowing this non-functional requirement to be taken into account. Several research efforts including ShaikhAli et al. (2003) have suggested extending UDDI by adding an upper tier level of information storing preferences or user defined properties.

More recently there have been a few research efforts which have looked at the wider social issues surrounding service selection. Barakat et al. (2012a) describes a method to be used to being correlation information into service selection. Using such techniques an actor can increase matching reliability and long-term goal achievement.

Paulraj et al. (2012) introduces a novel approach to service selection, by breaking down OWL-S process models into atomic services and matching the profile models of these individuals. Although this approach is a more than valid, it lacks the contextual analysis required to reason over composite service process models in unreliable situations. It also fails to reason over the order of actions or the relationships between actions in a workflow as a composite interaction.

As has been noted in Section 2.3.1 there has been much recent interest in using middle-agents to aid service selection. This notion has become increasingly popular, as researchers grow more interested in agents and agent technologies. However, the only truly mature and deployable technology currently in use is pure UDDI, with current stable implementations from most of the major software vendors (Microsoft, 2009; IBM, 2009). Table 2.7 outlines some notable research efforts into service discovery/selection, including those mentioned above.

Contribution	Citation	Notes
UDDIe	ShaikhAli et al. (2003)	An extension to UDDI to add a "blue pages", to record user defined properties associated with a service.
QoS	Ran (2003)	Adds QoS certifiers to existing UDDI registries.
Personalised recommendations	Powles and Krishnaswamy (2005)	Tracks UDDI usage and provides users with recommendations of useful services
Correlation-aware	Barakat et al. (2012a)	Accounts for quality correlations among Web services
eUUDIr	Ambrosi et al. (2005)	Adds improved ranking abilities to UDDI, results of searches can now be ranked by relevance.
OWLS-MX	Klusch et al. (2005)	Middle-Agent based service selection based on both logic based reasoning and content-based information retrieval techniques for services specified in OWL-S.
Pragmatics	Tamani and Evripidou (2007)	Attempts to add the notion of pragmatics to semantic based service selection. And by doing so, allow any selection process to consider context as well as semantics.
Interface compatibility measures	Garriga et al. (2013)	Primarily interested in gaining a deeper view of integration requirements.
Adding semantic data	Guo et al. (2006)	Suggests adding semantic information to UDDI repositories by way of an RDBMS back-end. Hopes to address the problem of matching capabilities within UDDI.
OWL-S and UDDI	Luo et al. (2006)	Adds OWL-S support to UDDI without the need to modify UDDI.
Web service competition	Najafi et al. (2012)	Allows actors to test numerous services in a given competition environment to determine which might be best to choose.
LARKS	Sycara et al. (1999)	A Middle-Agent based approach for comparing capabilities and requirements.

Table 2.7: Current Research into Service Discovery/Selection

2.3.3 Business Process and Workflow Based Techniques

Section 1.1 introduced workflows as a topic related to and yet separate from this theis. There is intersection between research leading to the answering of the govering question proposed in Section 1.4 and findings by the workflow selection community. In particular research surrounding the modeling of business processes and workflows using petri-nets, workflow reengineering and automated techniques to detect disfunction are all relevent to this thesis. The following subsections detail numerous current research topics from the fields of workflow selection, analysis and complience checking.

2.3.3.1 Workflow Selection

Much of the research into services and workflows centers around the central tenant that to fully achieve a clients requirements services will need to be composed into workflows (Fensel and Bussler, 2002; Cardoso and Sheth, 2003). Althought in this research Web services may expose partial process models, these are deemed to be different to full workflows which may require the collaboration of multiple actors across multiple media to provide the desired result.

Current research into workflow selection often focuses on the differenciation of services using non-functional means (Ludwig, 2012). Information like QoS data is appended to the service description (or stored from previous interactions) and can be used to select between similar providers (Ludwig, 2012; Panagis et al., 2007). Such techniques offer a good way to differenciate between providers in a situation where historical data (Panagis et al., 2007) or non-functional data is avaliable. However, these approaches are still relativly computationaly hard (Ludwig, 2012) and will fail where the required non-functional metrics are missing. Many other techniques rely on using IOPE matching (Kvaløy et al., 2005) or manual intervention (Luo and Tung, 1999; Yu and Reiff-Marganiec, 2008; Reijers et al., 2003).

Of all the research efforts in the field of workflow selection Ehrig et al. (2007) is of particular interest to this thesis. Here, the authors use the aproach of converting a business process into a Petri-net model for similarity checking. It is likely that this will be an approach which will be of use in answering the govering question proposed in Section 1.4, where by process models provided by a service description can be matched using similar methods to norms.

2.3.3.2 Workflow Analysis

Workflow analysis in research typicaly reffers to the analysing and checking of business processes. This is especily true when refering to business process refactoring and

reengineering (Kwan and Balasubramanian, 1998), which may take place during system design (or redesign). Often this is accompanied by the analysis of non-functional data such as timing information and scheduling requirements (Li et al., 2003; Basu and Blanning, 2001).

In the field of workflow analysis, three research efforts stand-out as being particularly relevent to the answering of the govering question proposed in Section 1.4. Li et al. (2003) as with Ehrig et al. (2007) (in the field of workflow selection) utilises Petri-nets to model workflows before processing, this approach is again of interest to this thesis as it enables complex processes to be marshalled into models flexible enough for many types of processing. Joosten (1994) introduces the notion of events, actions and objects within a modeled workflow triggering outcomes when processing, an approach which could be useful if models need to be state driven. Finally, Weigand and Moor (2003) introduces norms as a concept in evaluating language and action based information in workflows.

2.3.3.3 Workflow Compliancy Checking

Within the scope of business, complience checking is a core tool to ensure that processes and the entities enacting them are adhearing to a specified set of rules or regulations. Often this type of checking is carried out at "design-time" (Kharbili et al., 2008; Leyking et al., 2008) to check that a created process adhears to all requirements. Complience regulation is also often separated from the associated business process (Liu et al., 2007), or only applied when the process changes (Awad et al., 2008). Althought often only applied to busoness process documentation, complience checking has been applied to a service based environment. Viriyasitavat and Xu (2014) introduces complience checking for services-orientated workflows based on a trust-based model, a primariy requirement based approach which althought interesting still lacks required expresiveness to answer the govering question proposed in Section 1.4.

2.3.4 Process Mediation Techniques

Process (and more generaly service) mediation techniques have been looked at by numerous researchers in the SOA field. Although not directly linked to the more generic field of service selection as defined by this thesis, process mediation is still an interesting approach enabling the collaboration of two independent actors (services or a service and a consumer) where each party may have differing abilities or requirements.

Process mediation within the scope of research into web services and SOA has primarily been driven by the language WSMO (se section for information on WSMO Section 2.2.5.3). WSMO along with WSMX offers an ontology and execution environment where service provision can be mediated alowing services with with differing requirements to be

coelesed and used together (Shafiq et al., 2007). In such circumstances, mediation supports interoperation between herterogenious services by enabling the resolution of potential data or precondition mismatches between interacting parties (Haller et al., 2005). Cimpian et al. (2006) in particular sees mediation as a key component enabeling wide-spread Web service useage. This thesis does not argue with that hypothosis, but sees selection of partners based upon described process models as being a complementry technology. Finaly Cimpian and Mocan (2005) extends mediaton towards recifying differences in two choreorgraphy descriptions allowing implementing parties to interact, again this is a different appoach, but useful to acknowledge as it demonstrates processing over a predescribed set of actions to reach an outcome in an automated fashion.

2.3.5 Issues With Service Selection

Existing research efforts into service selection have come a long way by adding new features such as enhanced selection techniques to, or extending with extra information, existing UDDI based infrastructures (Ran, 2003; ShaikhAli et al., 2003; Ambrosi et al., 2005; Luo et al., 2006; Powles and Krishnaswamy, 2005). Many of the solutions envisaged by research projects have relied on middle-agents to process requests for services (Sycara et al., 1999; Decker et al., 1997). Middle-agents however are not without their own problems and many current solutions to the problem of service selection are still flawed. Specifically, there are still issues with privacy, trust, politics and the technical underpinnings of some of the applied technologies. These issues are investigated in detail below.

2.3.5.1 Privacy

Most notably the issue of privacy has not been fully addressed (Decker et al., 1997). If an actor is sending information about its preferences (Sycara et al., 1999) to a third party middle-agent, the actor needs to be careful with how much information it sends. Too little information and any matchmaker will be left trying to guess requirement information. Too much information and the requesting actor may expose potentially sensitive data and cause an unnecessary narrowing of any search space. When exposing requirements or capability data to external services, actors must find the balance between these two extremes. As will be discussed in Section 2.3.5.2, third parties cannot be explicitly trusted.

The issues surrounding privacy become more complex when factoring in the current trend towards service personalisation. Current UDDI-based service discovery and selection techniques simply lack the expressiveness required to support personalisation (Balke and Wagner, 2003). Even many recent efforts born in research (using middle-agents) fail to provide adequate support for personalisation.

2.3.5.2 Trust

Trust is also an important issue. How does a system trust a middle-agent to match in a fair and unbiased way? In an open and unmoderated environment, there is always a temptation for one actor to defect against another for personal gain. This is especially true if the defecting actor has decided that the increase in utility that will be gained by defection will be greater than any incurred penalty. Only recently has research attempted to address service selection in areas where this level of uncertainty exists (Stein et al., 2008) and this work only addresses a small section of the problem.

Quite often the issues of trust and QoS are mentioned together. A service which delivers a high level of QoS is often desirable. Determining the QoS of a service is a useful tool in service selection. However, ascertaining the QoS can be difficult, especially in open environments where the source of any information cannot be immediately trusted (Maximilien and Singh, 2004). Research efforts investigating service selection with QoS (Yu et al., 2007) have focused on how QoS values can be generated from aggregated data sources, how QoS may be used where services need to be composed and obtaining QoS data from expert third-parities (Gerding et al., 2009). The issue of QoS fluctuations during the lifetime of an actors' interaction with a service is an interesting one worthy of further examination, but is currently outside of the scope of this research.

To define QoS parameters or other SLAs, frameworks have been suggested for the specification, monitoring and enforcement of agreements between Web services and consumers (potentially other services). An example of one such framework is the Web Service Level Agreement (WSLA) framework which provides for the definition and monitoring of SLAs for Web services (Keller and Ludwig, 2003). Such frameworks are useful to define the limits and requirements of any SOA relationship before interaction, and during interaction to monitor and enforce participant's behaviours. However, within the scope of this research service agreements of this type are an extra step which could be used at a later point in the service selection (or invocation) process. As such agreements are outside of the scope of this thesis.

Trust is usually associated with services, not with service invocations. That is to say that a service will have a single level of trust. This is a very naive view as it is more than possible that a service could be expected to have different levels of trust for different queries (Bonatti and Festa, 2005). For example, a query for a book ISBN would not require the same level of trust as for a query involving paying for the same book using a credit card. Very few systems currently handle this level of information. As with QoS information, very little work has been done towards looking at the issue of trust fluctuating during the lifetime of an actors' interaction with a service.

2.3.5.3 Political Issues

If the technical issues involved in service discovery and selection are fixed, this may be unlikely to lead to an immediate embracing of these technologies. This is because may of the current "real-life" concerns revolve around business, strategic and political issues (Chatterjee et al., 2004). These may be internal business issues, such as relationships between departments or companies, substantiated by written or unwritten contracts or business processes. Such issues will need to be addressed by designers to ensure that programmes utilising automated selection adhere to existing conventions.

Political issues may also be wider reaching than a macro company or department level. It is possible that service selection will need to take into consideration wider macro political issues. Such issues could stem from political forces within individual countries, conventions held by a particular set of people or as the effect or global trade and relations. Such issues have been embraced by other industries and for the Web service community to be embraced globally so too it will need to embrace them. Many international organisations advertise themselves as being "local" around the world (for example HSBC[11] claims to be "the world's local bank"), in that they have knowledge of different political and social conventions around the world. Maybe it is time for "the world's local broker" to be created.

2.3.5.4 Context

Standard OWL-S matchmakers attempt to find the best match between provided service advertisements and requirements (Paolucci et al., 2002). Many matching techniques may be used, either on their own or in combinations to attempt to form the best match. These matching techniques provide a good grounding for automation. However, they lack notions of context (Tamani and Evripidou, 2007). Context is becoming increasingly important in matchmaking, especially in systems where the interests of a human or other intelligent actor are being represented. A few research efforts such as Tamani and Evripidou (2007) have looked at adding context dependent annotations to services. If service automation is to continue to be useful, context is an important issue.

2.3.5.5 Technical Issues

Away from the process of service discovery and selection, there are also problems with the technologies and languages which underpin the use of services on the Semantic Web. Most notably are the current issues with the OWL-S specification, including its lack of variables for modelling processes and missing holistic conceptual model which could enable developers to realise semantic Web services in a faster and more fluid

[11] The Hong Kong and Shanghai Banking Corporation: http://www.hsbc.com/

manner. Balzer et al. (2004) raises this as an issue, explaining that the OWL-S model is purposefully broad so as to enable usage in a wide range of scenarios. When generating concrete scenarios for use however, this flexibility can limit the ability for straight forward realisation (Sabou et al., 2003).

Sycara et al. (2004) describes the "broker paradox" which occurs if a broker wishes to handle services and requests using OWL-S. Essentially, this problem exists because of the inflexibility of the OWL-S specification, which requires that the process model exists in its entirety before invocation. For a broker, this poses the problem of how to advertise a service or offer a solution to a search with an OWL-S response, as the broker has no knowledge of which service will be chosen to fulfil the response. The OWL-S process model, returned from a search or advertised in advance, will need to be formed by combining the original service process model (the provider) with steps from the broker (making it seem like the broker is providing the service and thus hiding the provider). Without any initial knowledge of the provider, the broker will be unable to provide a complete process specification. The "broker paradox" forms a barrier to the effective use of OWL-S in brokered exchanges and must be overcome for OWL-S to be used in such circumstances.

Specific issues arise from the semantic similarities between concepts in service advertisements and requests. A number of papers have looked at the effects of poorly designed matching algorithms on matchmakers and brokers (Paolucci et al., 2002) and have found matching to be difficult and potentially imprecise unless multiple strategies are applied or extreme care is taken when choosing what data to use for matching (Sycara et al., 1999).

Problems have been noted with the current use of standard logic reasoners and standardised descriptions. Current service descriptors (such as OWL-S), are prescribed as being defined with an underlying decidable description logic (OWL-DL or OWL-Lite). This enables standard description logic reasoners to be used for matching. It has been pointed out that this purely DL based approach, when used to reason over such strictly annotated content, may not be enough for service selection (Klusch et al., 2005). Klusch points out that these approaches possibly artificially limit service matching techniques and ignore any implicit semantics found within service descriptions.

Major issues arise from current system's reliance on comparisons between IOPE's. As has been discussed above and in Section 2.3.5.1, choosing what data to expose and how to process it is a hard task. Further, in situations where multiple services all have the same basic requirements (IOPE's), but may operate in different fashions, purely IOPE based selection techniques may fail, or provide unsatisfactory outcomes.

2.3.5.6 Selection in Dynamic Environments

As an additional consideration, it is worth noting that the majority of service selection techniques mentioned so far operate in static worlds. Service selection is conducted during a present execution window when needed. The selecting entity selects a suitable service, but will often lack the necessary constructs to be able to reason over changes in dynamic environments during selection. Techniques utilising approaches from QoS, optimal path and more generic algorithmic procedures will often produce optimal results in synchronous "staged" worlds, but will lack the ability to cope as well in dynamic environments. This issue is especially well illustrated (and potentially solved) in (Barakat et al., 2012b) whereby the service selection problem is modelled as a multi-constrained optimal path selection problem with a multi-constrained Bellman-Ford algorithm applied to form a resolution. This enables the suggested system to react to changes in any observed system during selection, ensuring that the selected composite service is executable, of satisfactory quality and ready for execution. Of particular interest to this research, is the representation of workflows as abstract plans formed by directed graphs annotated using OWL-S and WSDL.

2.3.5.7 Norms and Current Selection Techniques

This thesis is concerned with answering the research objectives as articulated in Section 1.4. As such a reliable mechanism for service selection with the input of normative information is required. Based on the findings of this Section, with reference to the solution requirements in Section 1.4.2, it is clear that although exisiting selection techniques may fit within the Web service stack, they rely heavily on the use of IOPE based techniques and no solution based on normative processing has been identified. Thus there is a need for this thesis to provide a mechanism for the introduction of norm based information into the service selection process in order to fulfill the research objectives in Section 1.4.

2.4 Norms and Expected Behaviours

Societies and collectives[12] are bound to and regulated by norms (Lopez y Lopez and Luck, 2003). As standard patterns or models for behaviour, norms seek to prescribe acceptable practices for actors in a system. Normative rules may be implicit or explicit (Deutsch and Gerard, 1955). Failure to adhere to any given rule may result in punishment; the worst of which is social exclusion. Norms are required within societies of any size to ensure the maintenance of a fair and balanced equilibrium between participants (Sreenath and Singh, 2004). Ordered interactions may not proceed without some form of norms or

[12] A gathering of collaborating parties

prescribed specifications. This brings forth the question: what is the difference between a norm and a specification? Finding the answer to this question is not easy; many conflicting views exist. This work has chosen to define norms and specifications using the metrics employed by the ISO. ISO 9000:2005 defines a standard as "...a document. It is a set of rules that control how people develop and manage materials, products, services, technologies, processes and systems.". The OED[13] defines a norm as "That which is a model or a pattern; a type..." and "A standard or pattern of social behaviour that is accepted in or expected of a group.". For the purposes of this thesis a standard is defined as a static agreed rule, a criterion, a means by which to judge if a participant has adhered to a given task or request. A norm, on the other hand, is a potentially fluid concept, often context dependent, by which participants can be regulated.

Within any society, there will be three main categories of norms and enforcements: self-enforced norms created and monitored internally by a single actor, personally enforced norms enacted between two interacting actors and norms which are specified with punishments sanctioned by the community (social norms) (Kandori, 1992). The most common norms are self and personally enforced norms (Macaulay 1963 points out that these include "social pressure" and "reputation" and are far more prevalent than contracts or social norms).

Norms are powerful regulatory constructs in communities, especially where people are not governed by traditional organizational hierarchies, as is true for complex knowledge creating online networks (A. De Moor and van den Heuvel, 2004).

Norms in the context of computing, and more specifically in the context of activities surfaced by a service, are by the broad nature of computings uses broad in their own definition and application. For example norms may be used to govern which actors are permitted to carry out the edit process whilst utilising a publising service (de Moor and Jeusfeld, 2001), permit work to be carried out with or without a budget present (Chung et al., 2003), or the level to which workflow representations that capture scientific analysis processes should be made explicit (Gil et al., 2007).

2.4.1 Case Studies

Case studies for norms focus on research into the causes and effects of normative facts in various societies of actors. The rest of this section will look at a number of different types of norms, how they might be created and how they might exert pressure on actors to change behaviours.

[13]The OED and ISO 9000:2005 are the two sources of definitions for terms used in the ISO 9000 series of quality management standards. Chosen as they represent a good cross-section of language used in business today.

2.4.1.1 Self-Enforced Norms

Self-enforced or internal norms are created and maintained without interference or regulation from external governing actors. In such circumstances, the only incentive to obey is that obedience confers private benefits; greater than any private cost of compliance. Private costs of self-enforced actions are often very low and any (self) perceived rewards are high. In human society, this notion of self-interest is one of the most powerful human behaviours (Miller, 1999), motivating many research decisions and resulting actions. Such norms can be as obvious as knowing that you need to be able to speak a foreign language to be able to communicate in a foreign country, or as subtle as Sarah playing the piano every morning. In the last example (adapted from Hechter and Opp 2005), there is no external force making Sarah play the piano every morning; Sarah plays the piano because it gives her enjoyment (a benefit). Sarah perceives any effort (cost) taken to play the piano, either in the form of time or physical and mental exertion, to be greatly outweighed by the enjoyment she gains from the action. Recent work into self-enforced norms has also focused on the interactions between self-interest and wider social norms or structures. This can be seen in psychology studies such as Sheeran and Taylor (2006), looking at condom use amongst adults. Such studies have found that the most immediate and important predictor of condom use is that of having the intention to wear one in the future.

2.4.1.2 Private (Peer-to-Peer) Norms

Private (or peer-to-peer) norms occur between two or more actors participating in an interaction. Norms may be specified by one or more of the actors and may apply to one or more of the actors. There is be no governing body regulating private rules, as all regulation is private. This kind of norms especially crops up in interpersonal and business relationships (Kramer, 2006). Here trust and reputation are key to the success of any interaction, allowing parties involved in an interaction to undertake actions with little danger of incurring extra costs or loss of benefits. Breaking a private norm breaks (or greatly damages) the trust between actors, hindering any future iterations (Dabholkar et al., 2008). A simple example of this is that of the violation of a psychological contract (Robinson and Rousseau, 1994). Psychological contracts extend from written contracts but are themselves unwritten (and often also unspoken). They represent norms that parties can expect from a stated contact (such as a contract to work, for which a worker might expect to be rewarded with respect or unstated perks for working exceptionally hard). If either party breaks these norms, the contract still stands, but the degree of trust between parties is degraded (a worker may choose not to work as hard). It must be noted that different cultures have different attitudes to the handling of private norms. This is becoming especially important as China opens its doors to trading with the west. The Chinese handling of interpersonal and business relationships is critically different

from the western way (Shenkar and Ronen, 1987). Chinese logic is far less linear than western logic, leading to a negotiating style which is far more geared towards keeping harmony rather than maximising short-term pay-offs (Chen and Starosta, 1998). Any view of norms will have to be able to accommodate these differences in order to operate in an increasingly global marketplace.

2.4.1.3 Social Norms

Social norms are created by communities and apply to the whole, or a subset, of the participating actors. Enforcement is carried out by the community as a whole through a process known as community enforcement. Community enforcement encourages conformity by causing defection against any one participant to provoke sanctions from all (or a subset) of the rest of the members of the community. An example of this would be a community where deviator's could be "marked" so as to warn other participants. Once an actor defects during an interaction, he/she would be "marked" so that others would be wary about trusting that actor. This reduces the chance the defector then has of interacting and diminishes any likely payout he/she might receive. This provides a disincentive to defect. The opposite may also be true. It could be possible to reward "good" participants by "marking" them as trustworthy, or especially worthwhile business partners. Situations such as these are starting to appear increasingly in modern business. Traditionally much of this "marking" was done through official channels and word of mouth. Now thanks in part to the growth of electronic commerce, growing numbers of people are reviewing and rating businesses. Examples from "real life" include sites such as eBay (Resnick et al., 2000), where buyers and sellers can rate each other, discouraging bad behaviour and defection (such as non-payment and poor item description). This kind of system also exists in financial systems, where credit ratings can be used to coordinate economic institutions (Boot et al., 2006) and help to reduce bad business decisions, or miss-regulation (Levich et al., 2002). Individual societies also have internal mechanisms such as offender registers and prisons, to help identify dangerous or devious individuals and often prevent such participants from having the opportunity to re-offend.

2.4.2 Conversational Norms

As well as norms regulating behaviours, norms may also be used to regulate conversations or dialogues between actors (de Moor, 2002). In many cases, conversational norms will be prescribed as a standard at design time, with the punishment for non-compliance being the complete failure to fulfil conversation. However, in increasingly dynamic environments and systems it may be possible for actors to choose from a number of valid conversational paths, or to alter the current path at runtime. In these cases norms prescribed by society or any central point of control, may be required to govern any interaction.

2.4.3 Business Rules

Norms in business are often referred to as business protocols (or rules). They are statements that define or constrain some aspect of a business process (workflow), which potentially enforce structure on a process mirroring that of an external entity. There has been a substantial amount of research carried out into defining and monitoring business rules (Kolber, 2000) especially in electronic institutions (Cranefield, 2006) and business process languages (Charfi and Mezini, 2004).

2.4.4 Norms and the Law

Social norms are rules prescribed neither by an official source, such as a court or legislature nor enforced by threat of legal force (Posner, 1997), yet are regularly complied with. Laws pre-date political systems, originating as sets of norms among early societies. So laws cannot be discussed without some mention of the norms from which they come and norms should not be discussed without some mention of the laws many of them have lead to.

The study of norms and the law has in the last 20 years gained a great deal of interest (Ellickson, 1998; Pettit, 2002; Sciaraffa, 2005; Drobak, 2006; O'Donnell, 2007). Originally the majority of the work in this field was conducted by anthropologists, sociologists and theoretical economists. However, research has also been conducted by legal theorists. Particular success has been had in the fields of tort law and public policy. Tort law is partially governed by the notion of a negligence standard, that is to say that a defendant will be liable if he or she has failed to meet a particular standard. Evidence had been found linking the level of this standard to a level found as the social norm within a profession or society (Sciaraffa, 2005). Additionally Pettit (2002) notes two issues which law makers should consider when dealing with norms: whether social norms will work for or against a legal system and if new laws will give way to new norms.

2.4.5 Norms and Conventions

In addition to norms, social rules capable of influencing actors may possibly include conventions as well as customs and rituals. The later pairing falls outside of the scope of this research as they are seen to be of less power than the more concrete implementations of norms and conventions. As for how do norms differ from conventions? The answer seems to lie in the origin of each as mental concepts and in the reasons for compliance. (Conte and Castelfranchi, 2006) offers an interesting investigation into the effects of norms and conventions; it can be learned that conventions should be taken as emerging spontaneously and gradually from patterns of behaviours, whereas norms are potentially issued by an overriding authority. The reasons for adhering to norms are often stronger

as a result. Actors often adhere to normative directives as a matter of course to escape sanctions. However, often at an individual and personal level, norms and conventions will appear to spread throughout communities in the same fashion, becoming a belief in the mind of an individual; the only exception might be the drive to adhere to a norm over a convention when provided with an opportunity to defect.

2.4.6 Norms and Policies

Related to and often confused with norms, this research views policies as being important for the definition of well-formed social systems, but not synonymous with norms. This perspective is based on a set of views very close to that articulated by Boissier et al. (2013) that norms govern behaviours within a given organisation or system where as policies guide individual behaviours within a given domain. It is recognised that for the most part, all policies can be aggregated to norms. The inverse, however, is not true. Norms may or may not be formed of policies, depending on where the norm emerged from and what control is exerted. For example Boella and van der Torre (2004) differentiates between regulative and constitutive norms where the former may be policies and the latter not.

2.4.6.1 OWL-Polar

As an important element in the design of any organisation or society, policies have been identified as a potentially key element in the Semantic Web technology stack. Sitting with the trust layer, well defined policy documents should allow actors to exchange domain specific regulatory and behavioural information.

One such attempt at producing a policy description language for wider use is OWL-Polar (Sensoy et al., 2012, 2010). The OWL based OWL-Polar represents policies (and norms) in multi-agent systems. Such an approach lets actors use ontological reasoning mechanisms to make policy based decisions. In OWL-Polar policies are declarations of constraints on the behaviour of components within a governed distributed system. Policies are defined using OWL-DL and feature activation, deactivation conditions and capture the capture the distinction between activities which are required, prohibited and permitted.

This research find agrees with OWL-Polar on a number of factors, including the representation of policies and reasoning using OWL-DL. However, this research finds issue with the way that the authors seem to confuse norms and policies. This research agrees with the findings of Boissier et al. (2013) and others who find that norms are not equal to policies. According to Boissier et al. (2013) policies are much stricter atomic elements, useful for the constraint of individual actors behaviours within a domain; where as

norms (although potentially constructed using numerous policies) provide a way to guide behaviours in the context of a wider system. Additionally there is no room within OWL-Polar for inconsistent policies, or notions of cost to violating a policy, which are two elements this research considers as key for the representation and wider deployment of norms (but not for the representation of policies).

2.4.7 Representing Norms

Conceptually norms are a useful tool in the fields of social science as well as computing. However, to make norms truly useful to researchers and implementers, methods for the representation norms must be provided. Agotnes et al. (2007), Sergot and Craven (2006) and McNamara and Prakken (1999) are a selection of the work available on the use of formal languages to represent norms. The following is a short introduction to several key languages and techniques used for normative representation.

2.4.7.1 Kripke Semantics

Saul Kripke is responsible for numerous important contributions in the field of maths and in particular, modal logic. By far the most important examples of these ideas bear his name: Kripke frames and Kripke models (known collectively as Kripke semantics).

$$\langle \mathbf{W}, \mathbf{R} \rangle$$

Figure 2.12: A Kripke Frame

$$\langle \mathbf{W}, \mathbf{R}, \mathbf{V} \rangle$$

Figure 2.13: Kripke Model

A Kripke frame (Figure 2.12) is a pair where W is a non-empty set of possible worlds and R is a binary relation where $wRw' iff (\omega, \omega') \in R$. That is to say that W is a set of all possible worlds w and w' are possible worlds in this set and w precedes w' such that w' is reachable from w. Kripke models (Figure 2.13) are triples such that W is the set of all possible worlds, R is a binary relation (as with Kripke frames) and V is a mapping $P \rightarrow 2^W$, where P is a set of propositional values. Or more simply put, V is a mapping between any world in W and any satisfiable modal formula.

Kripke's ideas were a breakthrough in maths and his work is explained here as it forms the basis for the model theory of non-classical logics. Without which many of the logics detailed in the following section and used throughout this thesis could not have been created.

2.4.7.2 Deontic Logic

The obvious choice for representing norms is deontic logic (Torre, 2003). Deontic logic is the language of norms, obligations, prohibitions and permissions. A branch of modal logic first proposed by Mally (1926). This original version was based on propositional logic. Often confusion about the exact meaning of deontic terms stems from framing deontic problems as natural language statements (Hansson, 2014), formal methods and systems can help resolve these issues. The first plausible deontic system was proposed in von Wright (1951). Modern SDL is a Kripke-style system based on von Wright's original 1951 language. SDL adds new operators for ought to ($\bigcirc$), permitted to (P) and forbidden (F) to standard modal logic, along with a number of new axioms (Figure 2.14).

A1. All tautologous wffs of the language [TAUT]
A2. $\bigcirc(p \rightarrow q) \rightarrow (\bigcirc p \rightarrow \bigcirc q)$ [OB-K]
A3. $\bigcirc p \rightarrow \neg \bigcirc \neg p$ [OB-D]
R1. $If \vdash pand \vdash p \rightarrow qthen \vdash q$ [MP]
R2. $If \vdash pthen \vdash \bigcirc p$ [OB-NEG]

Figure 2.14: SDL Axiomatization

SDL is not perfect. It suffers from a number of problems, most notably Forrester's paradox (Forrester, 1984), characterised in Figure 2.15.

There are have been numerous updates and additions proposed for SDL. Most aim to solve existing problems or enable richer reasoning (in dynamic environments). Most notable of these evolutions is dyadic deontic logic, which addresses Forrester's problem (Figure 2.15) by formalising contra to duty reasoning and adds new binary deontic operators. There are many other forms of deontic logic which have been suggested in research, including; non-monotonic, paraconsistent, dynamic and contextual. Many of theses type are highly controversial and as yet only used in research. The use of deontic logic in computer science is not without its critics (Broersen and van der Torre, 2012).

Example The following are examples of deontic logic. The first (Figure 2.16) shows pure SDL in use, as it might be in formalising a system, or by mathematicians wishing to study a problem. The second and third examples demonstrate how deontic logic may be transformed into a computable language for interpretation by an automated system (or as a part of a more expansive formalisation of a system).

The second example shows the first of two languages based on deontic logic: LLD. LLD (McCarty, 1989) is a Lisp-like language incorporating dyadic deontic logic constructs with notions of sorts, events and time. Developed by L. T. McCarty in the 1980's, LLD has been designed to incorporate deontic reasoning into reasoning about law and lawful acts. The example in Figure 2.17 states simply that "any corporation which owns cash

Initial sentence:
Abner ought to help Esdras, whom he will kill tomorrow

Take ought to be scoped to the entire sentence
(a) Ought (Abner helps Esdras, whom he will kill tomorrow) $[\bigcirc(a \rightarrow e)]$

Excluding the ought, the sentence now reads
(b) Abner will kill Esdras tomorrow $[a \rightarrow e]$

Therefore
(c) Abner ought to kill Esdras tomorrow $[\bigcirc a \rightarrow \bigcirc e]$
This is incorrect, but correct under the standard principles of deontic reasoning (Fig 2.14)

To fix this
(d) Ought (Abner helps Esdras). Abner will kill Esdras tomorrow. $[\bigcirc a \rightarrow e]$

Figure 2.15: Forrester's Paradox (adapted from Forrester 1984)

A) $\neg Pp \rightarrow Fp$ [not permitted to do p is the same as being forbidden to do p]
B) $a \rightarrow Fb$ [having done a then forbidden to do b]
C) $\bigcirc c \rightarrow (Pd \wedge Fe)$ [obligated to do c then permitted to do d not e]

Figure 2.16: Pure SDL Example

has an obligation to distribute all its cash to its stockholders" (adopted from Wieringa and j. Ch. Meyer 1993).

```
(obligate ?
(own ? (corporation ?X) (cash ?Y))
(distribute-dividend ?) (corporation ?X)))
```

Figure 2.17: Example in Language for Legal Discourse (LLD)

Figure 2.18 shows the final example. Again a language evolved from deontic logic has been chosen. Figure 2.18 illustrates the language ESPLEX, a Prolog-like language, based on deontic logic. ESPLEX was originally proposed in Biagioli et al. (1987). ESPLEX was designed to reason over agricultural tendencies in Italy, so has limited appeal to this work, except as an example of how Prolog-like syntax might be used to produce a deontic logic based language. Figure 2.18 describes a condition under which termination of tenancy is permitted. In this case, the condition is that the tenant must be a small farmer for the terminal to be permitted.

```
Permitted (termination, tenant, tenancy) :-
cond (small farmer, tenant),
proc (termination, tenancy).
```

Figure 2.18: Example in ESPLEX

Deontic Logic and Truth Norms are by there nature in society neither true or false. However, in order to use deontic logic to reason over norms and normative behaviours, a true or false statement must be created. The most common approach to solving this problem is to utilise deontic logic to reason over logical propositions (Broersen et al., 2012). Thus, it is possible to state with a true or false outcome whether something ought to be done. For example, it is possible to stipulate that "An actor ought to stand" with the outcome of the described situation being true or false.

2.4.7.3 Action Logic

The second logic of worth noting in this section is action logic. Developed during the 1950's by Swedish philosopher Stig Kanger (Holmstrom-Hintikka et al., 2009), action logic attempts to answer the question: "is there [a] logic of action?" (Segerberg, 1992). This is a particularly hard question to answer, as the philosophy of action is not complete. Despite utilising differing techniques to come to their conclusions (logic and philosophy of action) the questions they attempt to answer are essentially the same. For action logic to be successful, philosophers will first have to finish exploring action; before mathematicians can formalise it fully. Since Kanger attempted to formalise a logic for action there have been numerous other attempts. Of note is the action logic ACT (Pratt, 1990), which takes quite a simple approach by adding two new operators to regular expression logic: pre-implication $a \rightarrow b$ (had a then b) and post-implication $b \leftarrow a$ (b if-ever a). These two new operators enable the expression of sentences such as;"$king(loves \rightarrow pays)taxes''$ pronounced as "Who loves their king pays their taxes.". This is as far as this thesis will take action logics. There are numerous action logics defined in scholarly works. However, none of them is as well defined, or as fit for the purposes of this thesis as deontic logic.

2.4.8 Storing Norms

From the beginning of conscious evolution, norms have been stored as implicit knowledge within society. More recently many of the norms humans adhere to have been written down, either as terms in contracts, conditions in agreed interactions, or laws passed by some administrative body. This transition has also been witnessed in agent societies (at a much-accelerated rate), where agents have initially had many conventions and norms unintentionally hard-coded into their design by human designers. Future agent or SOC based environments norms will have to be well defined to prevent any one actor from subverting the system and gaining an advantage, either by mistake or maliciously. Section 2.4.7 outlines how norms may be formed and represented formally, either in pure deontic logic or in a deontic based language.

Having decided what norms a system will need and how these norms will be formalised, the next step for any designer is to decide how any norms will be stored. There are

numerous options for storing norms (outlined in Table 2.8). At the most basic level, norms can be formalised and then implemented directly as code or conventions within an agent's structure (or within the structure of an overseeing authority). Such norms are known as hard-coded norms, are static (cannot be changed with time) and represent the majority of norms in use in traditional agent-based systems. Systems involving static norms are useful and easy to set up, but in order to harness the power of norms developers will need to look at dynamic systems. For a normative system to be dynamic, norms within that system will need to evolve over time. Agents will have to have the ability to create, modify, change and select different sets of norms during the life of any interaction, as they would for goals, plans, or any other dynamic information. Table 2.8 outlines a number of options for the storage of dynamic norms.

Technique	Pros	Cons
Hard-coded	Limitless control. Fast.	Cannot be changed at run-time. Non replica-table.
Flat File	Easy to set up	Slow. No built in query or inference tools.
RDF (TripleStore)	Fast. Scalable. Good inference capabilities. Remotely accessible.	Can be complex to set-up
Database	Mature technology. Fast. Good for large quantities of data.	No advanced querying mechanisms or inference.
Belief/Obligation Base	Tight coupling with existing agent architecture	New reasoning structures might be needed. Tied to BDI architecture.

Table 2.8: Storage Techniques for Norms

2.4.8.1 Flat File

Flat file storage is the oldest form of permanent storage still in use today. Quite simply, norms are written to a file on the local system where they can be accessed by the agent at runtime. Norms must be encoded in a machine readable format. XML would probably be preferable as human readability would aid design. Flat files are relatively slow (unless the agent caches the contents after first read) and on their own offer no query structures. However, they do make transferring large numbers of norms between agents easy as the files can be downloaded from one to another.

2.4.8.2 RDF (Triplestore)

Triplestores are essentially purpose-built databases for the storage and processing of RDF (see Section 2.2) triples. Modern triple stores can store billions of triples (the largest of which is currently Oracle Spatial and Graph with Oracle Database 12c with 1.08 trillion triples[14]). Whilst the ability to hold such large data sets would make a triple store useful in the implementation of an off-site norm store, it is arguable how useful this ability would be to an agent on its own. Data-models for information held in triple stores can be defined as ontologies using any of numerous tools. Many triple stores also allow advanced queries via SPARQL, along with useful tools such as inference engines and multiple output formats. RDF and triple stores may be useful for the representation and storage of norms as they offer both rapid access (local and remote) and querying, including enabling the inferring of new norms. However, these techniques do require a triple store to be set-up and running either locally or on a remote server for the storage of norms.

2.4.8.3 Database

Databases are used for to store large amounts of data. Many very stable database systems exist, on virtually every platform imaginable. The main advantage of using databases is that they offer the ability to store large amounts of data in a form which can be accessed or updated quickly for simple queries. Databases do not, however, provide the complex inference or ontology based data modelling that triple stores offer.

2.4.8.4 Belief/Obligation Base

Many of the examples of norms in use in agents in Section 2.4 describe using BDI based agents. In particular, Dignum (1999) and Broersen et al. (2001) directly suggest the addition of systems of norms to BDI architectures. To facilitate the addition of norms to an agent architecture, there needs to be a place to store norms within the agent. The most obvious way to do this is to either model norms as beliefs and intentions (and so fit any new notions of norms into the BDI system), or to add a new store to the BDI system: an obligation base (Broersen et al., 2001). Modelling norms as beliefs and intentions would require no change to the reasoning logic of the agent, however representation and maintenance of norms could be complex. Adding a new obligation base would require new reasoning logic, but would make production and maintenance of norms easier. Both couple norms tightly with the underlying agent infrastructure than the previous methods and have been popular with recent agent research.

[14]http://esw.w3.org/topic/LargeTripleStores accessed 25/10/15

2.4.9 Reasoning over norms

As this work has shown in previous sections within the field of AI, norms can loosely be defined as denotic statements limiting the actions of participating actors (Wieringa and j. Ch. Meyer, 1993). Norms in agent-based systems are mechanisms to influence the behaviour of agents within an environment (Lopez y Lopez et al., 2002). The above sections have shown how norms can be represented and stored in an agent-based situation. Being able to accurately and efficiently represent and store norms forms the basis of any normative system. Additionally developers must provide systems to reason over these norms and systems to enable the outcomes of any reasoning to influence the behaviour of the underlying agent.

How an agent reacts to events, its beliefs, its desires, behaviours and underlying characteristics are all dependent on their style of reasoning (Hindess, 1988). Different styles of reasoning can lead to different outputs from the same set of inputs. One style might be applicable to one situation and not to another. Section 2.2.2 outlines a few basic agent types and their reasoning systems. Choosing the right reasoning system is as important as choosing any other part of the agent. Choosing an appropriate reasoning system for reasoning over norms is just as important. The meaning and nature of any prescribed norms can be altered by the reasoning system used. Often developers will also have to develop reasoning systems which can handle the detection of new situations and produce norms from these; a task which will require not only reasoning over existing norms, but also reasoning over whether a norm might have come into existence due to a change in circumstances.

Many normative systems for agents (including those listed in Section 2.4.15) implement norms either by adding them to the agents reasoning processes directly (as in Broersen et al. 2001), or by creating an independent normative module which can then be integrated into an existing system (as in Andrighetto et al. 2007a). Obviously implementing an agent from the bottom up to work with norms gives developers closer ties between norms and overall behaviours. However, it may be costly to recreate an agent completely and if the agent's normative structures need changing, a modular approach would be simpler and more flexible. The majority of normative systems are based on one of the original core agent types listed in Section 2.2.2. The choice can be influenced by a number of factors, including representation and storage techniques (systems with advanced reasoning structures, such as triple stores might lend themselves better to BDI or goal based approaches) and whether designers wish to implement a component to learn new norms from external sources (which might be purely reactive).

Gasparini et al. (2015) focuses on the efficient application of model checking to analyse properties of normative systems specifications. Although the reasoning over norms in this research is not focused on driving the actions of a particular agent (rather this research looks in to the validation of systems as a whole), the authors use of finite Kripke

structures, echos the intended path of this research and so is noted by the authors of this thesis.

2.4.9.1 Violations

For any norm-based system to be useful it must be able to deal with violations of norms. Violations can occur against an agent triggered by a third party, such as contract violations (Governatori and Milosevic, 2005), or can be triggered by the agent itself choosing to violate one of its own norms (Torre, 2003). Detecting external violations is a relatively simple procedure. The agent knows what to expect from any external actors and if it observes differently a violation is triggered. What to do with this violation is a more complex question. The agent may wish to ignore it, or to punish it, the agent may even wish to accept it and treat the violation as the emergence of a new norm. It will be the job of the agent to reason over each course of action and decide on the most suitable plan for the particular situation.

In some situations, an agent may wish to violate its own internal norms. This might occur if the agent thinks that the potential benefits of non-compliance exceed the costs incurred for the associated violation. Such circumstances are important; both to a highly social agent wishing to maximise its potential profits and to an agent wishing to communicate with agents who might share norms incompatible with its own.

In the design of any complex norm-based system, it is important to recognise that dealing with violations is as important as dealing with the norms themselves.

2.4.9.2 Conflict Resolution

Normative conflicts occur when two or more norms, activated at the same time, operate to create a situation where for one to be fulfilled, another must be violated. Normative conflicts should, where possible, be avoided (Vasconcelos et al., 2009) since any plan generated as a solution will by definition have to violate at least one norm and thus be sub-optimal (on a system-wide basis). There are many existing research articles which have been written looking at conflict resolution. Many suggest attempting to solve the problem of conflict at design time through the careful creation of normative catalogues (Vasconcelos et al., 2012). This it would seem is only suitable for closed systems with limited dynamism. More complex systems require real-time norm regulation and conflict resolution (García-Camino et al., 2008). This research agrees with this view and submits that conflict resolution should be considered when building any normative system, especially one where normative emergence/creation is to be allowed.

2.4.10 Self-regulation

If self-interest is the singularly most powerful motive influencing actors (Miller, 1999), so it follows that self-regulation will be the most important form of governance to any single actor. Often the initial question an actor will have to ask itself when planning an action is, "what do I want to do?", or in a perfect world, "what is the best course of actions for me to take?". An intelligent actor will decide during planning whether there is any benefit to be had from defecting against some known norm. These decisions form the basis of self-regulation.

Self-regulation enables an actor to govern their own adherence to norms. Actors may self-regulate adherence to both internal and external norms. With regards to internal norms, it is up to the actor to decide whether violation will cause a cost higher than any potential benefit. External norms tend to be more complex as they can affect several participants. In the case of external norms, an actor will need to decide if violating a norm will bring punishments from other actors greater than the benefit (that is if other participants will actually observe the violation in the first place). In both cases, it is up to the actor to decide if it is worth violating by measuring any predicted cost against any predicted value. Self-regulation and self-determinism are both likely to be complex tasks, especially for external norms where the effects of any violation may be felt for a prolonged period of time (and thus predicting any future effects may be difficult).

Self-regulation could also be construed as being "pure" self-regulation, whereby actors are responsible for making sure that they rarely violate any norms (any system in which norms can never be violated would be infeasible). In this way, no central control would be needed as it could be taken that every actor would adhere to all of the norms laid out in the system. This view of self-regulation is somewhat narrow and short-sighted as it leaves no room for violations and suggests that the norms in play should really be concrete laws.

2.4.11 Social Monitoring

If self-regulation enables actors to govern and reason about their own behaviours, social monitoring enables actors to observe and reason about external actions. Most intelligent actors existing in a larger system will have some form of external input or monitoring ability. In a normative context, this means that actors can use the norms they have accepted to evaluate others' behaviours (Conte and Dignum, 2001). If the general theory of norms is that of prescriptions impinging on actors minds in order to regulate their actions, then social monitoring enables the observation of normative compliance of others to both aid internal reasoning and affect change in the surrounding system.

There are numerous reasons why normative actors may wish to undertake social monitoring: to ensure normative compliance, to detect violations and collect any benefits

where sanctions are favourable, to share norms, to defend shared norms (especially where the violation of a shared norm leads to sanctions which may effect the actor), to fulfil a responsibility as laid down by an overarching norm, or to aid in reasoning when considering violating a norm. All of the above are valid and socially aware normative actors which may utilise one or many, either to gain benefits or escape sanctions.

2.4.12 Norm Emergence

Of all the areas of norms and normative research, the behaviours and conventions facilitating the emergence of norms within systems of norm-aware actors have received particularly high levels of interest in the past few years. In social science, there is currently no consensus for any of the general mechanism involved in norm emergence, norm compliance or enforcement. The study of emergent social behaviour has benefited greatly from computational and simulation-based modelling, afforded by computing specialities (Conte et al., 2012). In particular, the study of norm emergence has become one of the most prolific domains of investigation.

Andrighetto et al. (2010) suggest a definition for norms as "behaviours which spread through society along with their associated prescriptions". Norms are first detected then internalised and they become salient with regards to an actor coming under their control. Drawing attention to a norm and making it salient is intended to elicit an appropriate behaviour. This works particularly well as internalising a norm requires a level of detail about the behaviours and beliefs of actors governed by that norm.

As mentioned previously social monitoring involves actors observing the external world to detect the behaviours of others before reasoning over these in relation to held normative beliefs. Actors with both normative awareness partaking in social interactions will almost always enforce and propagate their norms into the social environment. More or less deliberately, they act as norm issuers(Conte and Dignum, 2001). Many of the most basic norms (such as meta-norms like reciprocity and respect) emerge naturally in societies in response to social monitoring of related actors. Additionally, the monitoring of social behaviours and reasoning with regards to averages and trends might lead to the formation of norms which represent current excepted behaviours without the need for a governing authority. This later element of normative creation is often referred to as normative immergence (Andrighetto et al., 2009).

The emergent and immergent natures of norms within groups of social actors leads to the inevitable dynamism of shared environments. As a result, numerous research efforts have focused on handling the evolution of normative systems (Boella et al., 2009). Here actors must be able to respond to changing norms and required behaviours. Many research efforts see normative system evolution and modification to be of crucial importance to the robust operation of any norm-aware social system (Oren et al., 2010). This research

will not comment any further on this topic (as it is held to be mostly out of scope with regards to the selection of services which are likely to be relatively static in their normative nature), other than to acknowledge that it exists as a concern and fertile area for future investigation.

Recent research into multi-agent systems and norms has focused on the area of norm emergence. Of the two methods for norm creation in multi agent systems (offline during design and online synthesis using techniques for detecting norm emergence), norm synthesis is has garnered the interest of researchers (Morales et al., 2015b). It seen as being able to produce reliable normative facts given a set of interacting agents where aspects of the world system are unknown (Morales et al., 2015a).

2.4.13 Standards

There are very few agreed standards for norms, norm processing, or normative agents. Within research, there have been many proposed languages and frameworks for the incorporation of norms into agents and other computer systems (Castelfranchi et al., 1999; Broersen et al., 2001; Lopez y Lopez et al., 2001; Kollingbaum and Norman, 2003; Lopez y Lopez and Luck, 2003; Lopez y Lopez and Marquez, 2004; Felicissimo et al., 2005; Boella and van der Torre, 2006a). However, none of these proposals has made it into wider use and any standards defined from them are a long way off.

It could be argued that standard deontic logic (von Wright, 1951) could be regarded as a "standard" as it is the underpinning for much of the work in the field. However, with many versions and variations available aiming to solve the problems of SDL (Forrester, 1984; Wieringa and j. Ch. Meyer, 1993; Meyer et al., 1994; Torre, 2003; Holmstrom-Hintikka et al., 2009), it is probably more useful to treat SDL as a starting point from which new standards can be created.

2.4.14 Tool Support

As there are no agreed standards for implementing norms in agents, so there are no standard tool sets either. There are numerous tool sets and extensions to existing systems proposed in literature, for example in Broersen et al. (2001) where the BOID architecture is proposed as an extension to existing BDI architectures. Many of these extensions have been proposed as modules which can easily be added to an existing system (Andrighetto et al., 2007a). There is little tool support currently available with regards to the implementation of norms in agent-based system. Tools supporting the obligation to roles in electronic institutions are one of the few examples of implemented norms. Systems such as J-Moise+ based around the Jason BDI framework provide some support for defining any obligations an agent has to undertake to fulfil a role in an

organisation. This is a very limited view of the usefulness of norms. However, it is likely that as norms become more widely used in agents, so increasing numbers of tools will support normative concepts, or at least provide the ability to integrate normative reasoning through external modules.

2.4.15 Normative Agents

As with any type of computer system, formal methods and processes for the definition of norms and normative reasoning in agents would aid their design, production and testing. However, as norm based agents are relatively new and the field of norms in agents is an emerging one, there is currently no single method for defining or handling norms in normative agents. The majority of research carried out into normative agents is based around deontic logics and BDI architectures (Boella and van der Torre, 2006b; Boella and Lesmo, 2001). Outlined in the following sections are a few of the current norm based agent research efforts, with a specific focus on the logic employed by each.

This review has chosen to focus attention on the most relevant existing systems with relation to norms used in agents, web services, the semantic web and other open environments. It is acknowledged that there are large bodies of work which centre their research on other areas and applications of norms. For example, Bulling and Dastani (2011) makes some interesting and valid points surrounding game structures for the abstraction of formal normative environments and forming an equilibrium between norms and behaviours. However, this and many other areas are too far removed from notions of formal norms in agencies as to be prescribed as being outside of the scope of this review.

2.4.15.1 Adding Norms to BDI

Dignum (1999) offers some interesting insights into the addition of norms to agent design and forms the basis for many modern papers. Dignum splits norms into three levels using deontic logic to model concepts for each. He states that there are private norms, contract norms and convention level norms and it can be taken that these are equivalent to the three levels of norms discussed in Section 2.4. Dignum et al. (2000) goes on to augment the BDI architecture with obligations (as a form of belief). Here, obligations are triggered by a new type of internal event: a deontic event. Deontic events are created by changes in the norms, obligations and beliefs and cause plans with matching deontic preconditions to be invoked.

In other work, Dignum has noted that growing importance of social commitments in Electronic institutions (Dignum, 2002). Where social commitments primarily rely on obligations and permissions with regards to communicative acts.

2.4.15.2 Norms in Multi-Agent Systems

Beyond focusing initially on research surrounding the basis of norms, this thesis is primarily interested in their potential use for exerting control in online systems involving one or more semantically aware actor. As has been previously stated, broader topics of norms in computational modelling and theory have been discounted. Doing so presents a more focused view of norms as statements which impose a type of limit on the behaviours of actors in a given organisation or system. This is most interestingly presented when norms are integrated into collections of intelligent socially aware agents. Boella and van der Torre (2006b) introduces the concept of normative multi-agent systems as collections of agents whose interactions are norm-governed and who can, if they so choose, deviate from a set ideal. He also presents four action models for actors in multi-agent system: teleological (rational goal-directed), normatively regulated (grouped and following obligations), dramaturgical (considering the agents' inner world and outer presentation) and communicative (using language to bring about understanding).

Most of the research efforts discussed in the rest of this chapter follow the patterns and definitions laid down in (Boella and van der Torre, 2006b). As such, many different multi-agent examples will be encountered, all sharing similar goals, but often differing in their approach. This comparison will be used as a basis on which to build any future innovations.

2.4.15.3 Rational Agents

Rational agents can loosely be defined as those following behaviours which lead to some calculable final utility. In normative agents, rationality allows the prediction of the potential actions of an actor based on his/her current beliefs and a known set of decisions. As the agent will always try to maximise their own utility (in the short and/or long-term), it is possible to predict how it might behave when presented with a set of normative restrictions (Alechina et al., 2012). Socially rational agents depend on the ability to predict the actions of others (Boella, 2003). This research finds the notion of rationality to be useful as it suggests that norms can foster cooperation even without directly effecting behaviour to that extent. Boella (2003) adds to this notion of rationally by suggesting that not only can agents observe the actions of others and predict paths, but that this ability can be used to anticipate actions and foster cooperation. This is often referred to as anticipatory cooperation (Castelfranchi, 1997).

2.4.15.4 Normative System Change

In this chapter the importance of social monitoring (Section 2.4.11) and the creation of norms thought emergence (Section 2.4.12) has been previously explained. For normative

agent-based solutions to be of use, they will need to be able to handle change and exploit normative emergence to foster greater utility among those participating. Such behaviours enable not only the introduction of new norms but also the exclusion or removal of derelict ones (improving overall system performance). Boella et al. (2009) focusses on normative system change by proposing a framework for normative system change enabling norm adoption and revision by using a rule-based input/output approach. This includes the introduction of notions of redundancy (where two norms are equivalent) and implication (where the combination of a set of norms implicates another norm). These are ideas that this research finds particularly compelling for the long term maintenance of complex normative systems.

2.4.15.5 EMIL-A

Outlined in Andrighetto et al. (2007a) and Andrighetto et al. (2007b), EMIL-A aims to deal with the emergence of norms in agent societies by providing a normative architecture for agents. This architecture could be thought of as a sub-module of a larger architecture. Essentially based on the BDI architecture, EMIL-A provides a set of normative beliefs, desires and intentions, as well as norm recognition, adoption and planning techniques. EMIL-A enables the storing and representing of norms and the adoption of new norms. Norms may be received from external sources as deontic statements, or activated via internal inputs. To facilitate the detection of (potentially previously unknown) norms from external sources, the EMIL-A recogniser utilises two structures: a normative board and a normative frame. As external norms are received, so they are matched to existing norms by the normative board. If a match is found, this norm is instantiated. If no match is found, the norm is then passed on to the normative frame, which recognises it and categorises it though a process of simulation. Through simulation the normative frame can forecast the effects any norm might have to the system before instantiating it. If the agent accepts the simulation, then a new normative belief is added to the norm belief base.

Andrighetto et al. (2009) used EMIL-A to successfully demonstrate how norms which do not correspond to common actions can emerge and how conflicting norms can coexist and compete with each other. This example shows the interesting and innovative results which can be gained from simulating behaviour in multi-agent systems and that simulations involving norms can help researchers to understand the potential benefits of a normative approach.

EMIL-A is still in its infancy, but it provides an interesting insight into how a BDI architecture might be implemented to form a norm module for an agent and how such a module could handle new and evolving norms.

2.4.15.6 NoA

Kollingbaum and Norman (2003) presents an agent architecture for the development of agents with normative capabilities. The prescribed Normative Agent Architecture supports the inclusion of norms through a language for the specification of norms and plans and an interpreter capable of interpreting and executing plans written in the NoA language. NoA is loosely based on the AgentSpeak(L) language and platform, with the addition of norms and a few other modifications (enabling the declaration of all effects of a plan in the plan and adding initial distinctions between states and actions). As NoA is based on the BDI architecture AgentSpeak(L), it too is BDI based. A NoA agent works towards a goal state (or set of potential states), as with all BDI agents. Norms are activated when the agents belief's match their activation preconditions. Activated norms can then influence the behaviour of an agent either by motivating goal generation or by restricting the options available to an agent during planning. This is a far simpler approach than EMIL-A but lacks the advanced handling of potentially unknown external sources of norms (although it does not rule out the inclusion of new norms into its planning/belief structure).

2.4.15.7 Norms During Runtime

Meneguzzi and Luck (2009) also uses AgentSpeak(L) but to look at a different and potentially overlooked area of agent behaviour, that of the impact of norms on the practical reasoning of agents. This paper is of the opinion that if agents are to operate in open environments, they need to be able to adapt to changes in constraints and restrictions at runtime. To achieve this, the authors alter the standard BDI model to handle norms at newly accepted (and activated) norms at runtime. In doing so, the current plan execution can be suspended while new plans (complying with the additional norms) are produced. This approach is initially similar in nature to that of NoA (Kollingbaum and Norman, 2003) except that here the agents focus is not on fulfilling norms, but on generating plans to reach a final desired goal where the plans are shaped by normative pressures.

2.4.15.8 N-2APL

2APL is an extensive multi-agent language and tool-kit which has been mentioned previously in Section 2.2.2.3. Alechina et al. (2012) proposes an extension to 2APL to allow the creation of rational normative agents. These additions include support for beliefs, goals, plans, norms, sanctions and deadlines. Similar to the extension of other BDI platforms, N-2APL stands out not only in its ability to leverage the existing 2APL tool-kit and provide for socially rational actors but also as a general framework for the representation of norms with deadlines where behaviour prediction is needed.

2.4.15.9 n-BDI

proposed in Criado et al. (2010) and improved in Criado et al. (2011), n-BDI extends a multi-context graded BDI architecture with norms. This extension to BDI enables agents to identify norms involving behaviours, infer the content of the norm and decide on compliance. Relative priorities can be assigned to actions and norms, providing agents with a mechanism for ordering plans and thus addressing conflicts between norms and mental propositions (Luck et al., 2012).

2.4.15.10 BOID

Broersen et al. (2001) proposed an extension of Thomason's BDP (Thomason, 2000) architecture, adding obligations. This new architecture was referred to as the BOID architecture. The BOID architecture decomposes the handling of norms into a number of sub-systems (for beliefs, obligations, intentions and desires) and an overriding control system which is tasked with coordinating these sub-systems and resolving conflicts. BOID is designed such that conflicts are expected. Possible conflicts are summarised in Table 2.9. The conflicts in Table 2.9 form a good basis for all conflicts in normative systems where obligations are modelled. The control sub-system in BOID expects conflicts (in fact BOID agents are designed to be based in complex environments, where they are overloaded with information) and attempts to utilise a number of coherent strategies to solve these conflicts, either by re-planning or choosing to violate a particular norm.

Conflict	Example
B O I D	It is obligatory to be honest. It is obligatory to be polite. If I am honest, then it would be polite.
BO	It is obligatory to see my mother-in-law this weekend. But I think I have no time to go.
BI	-
BD	-
OI	It is obligatory to see my mother-in-law this weekend. But, I already have a plan to go to a conference. If I go to the conference, I cannot go to see my mother-in-law.
OD	It is only permitted to smoke in a smoking area. I desire to smoke in my office. However, my office is a non-smoking area.
ID	-
BOI	It is obligatory that if I smoke, I smoke in a smoking area. I intend to smoke. However, I know that there is no smoking area here.
BOD	It is obligatory that if I smoke, I smoke in a smoking area. I desire to smoke. However, I know that there is no smoking area here.
BID	-
OID	I intend to go to a conference. I desire to stay in a luxury hotel. However, it is obligatory for me that if I go to the conference, I do not stay in a luxury hotel.
BOID	I intend to go to a conference. It is obligatory for me not to spend too much money for the conference. Namely, either I should pay for a cheap flight ticket and stay in a better hotel, or I should pay for an expensive flight ticket and stay in a budget hotel. I desire to stay in a better hotel. But, I know that the secretary has booked an expensive flight ticket for me.

Table 2.9: Conflict Types (adapted from Torre 2003)

2.4.15.11 Extending SMART

Lopez y Lopez et al. (2006) is one of a number of research efforts which extends the long-standing SMART agent framework (d'Inverno and Luck, 2001) to include norms.

In particular, this work features a strong theoretical base detailing many features of norms which this research agrees with. This includes the definition of a norm as being "characterised by their prescriptiveness, sociality and social pressure", when norms tell agents how to behave when more than one actor is involved and where some form of social pressure is involved. Additionally since many social rules can be defined through the level to which they prescribe behaviours, sociality and social pressure (such as obligations, prohibitions and social laws) normative models can be used to represent a wide array of rules imposed by societies.

2.4.15.12 Agent Planning

Reasoning over norms with regards to the explicit outcomes of an agent's plans (and thus influencing plan choice) has previously been review seen in this research in BDI based systems such as those in Meneguzzi and Luck (2009) and Kollingbaum and Norman (2003). Since many normative agent frameworks are turning to BDI (and other planning based approaches) as a foundation on which to develop concrete normative systems, it is not surprising that there has been a trend away from the development of frameworks for the representation of norms towards the use of norms in planning. During investigations many such research approaches have been evaluated, the following are two of note.

Like Meneguzzi and Luck (2009) Panagiotidi and Vazquez-Salceda (2012) focuses on the addition of practical normative reasoning to agent planning (from the perspective of a generic STRIPS and PDDL based planner). This approach uses norms to place restrictions on the planner whilst enabling an agent to take in to account costs involved when violating norms and utility gained from finishing plans. In this way, the designed planner can decide a path which provides the actor with maximum utility even when norms must be violated. This inclusion of "soft" restrictions into the planning stage is of particular interest, as are the results from Panagiotidi and Vazquez-Salceda (2012) that doing so poses only a small over-head on the planner.

In a similar off-line planning vein, Oren et al. (2011) adopts a utility based approach to plan selection. Informally, given a plan represented as an AND/OR tree, with actions specified as constrained predicates the effects of executing steps in a can be evaluated with regards to normative constraints. This differs from Panagiotidi and Vazquez-Salceda (2012) in representation and mechanisms used, but the outcome is similar; planning while accounting for defeasible normative constraints and attempting to find the most suitable outcome (highest utility).

2.4.15.13 Norms as Constraints on Agents

Meneguzzi et al. (2012) (also Meneguzzi et al. 2010) describes a BDI based norm representation using constraints to limit the action of agents. This approach is interesting

because constraints are introduced during planing to determine which plans are suitable for execution and effectively restricts the acceptable domains of operation. During planning nu-BDI agents annotate plans with constraints placed on them by norms. nu-BDI however, departs from the traditional model of constraints by allowing the normative notion of violations to exist. That is to say that where no acceptable plan is found, incremental violation of norms is allowed so that the agents final goal can be met. This ability separates this view of normative constraints from that of policies or strict constraints on behaviour, enabling agents' to reach acceptable goal states in complex and unforeseeable situations.

2.4.15.14 Procedural Norms and Agents

Procedural norms are constraints addressed to agents playing a social role in the normative system. Such norms might be prescribed to motivate agents to recognise sanctions, adhere to other norms, or recognise violations. Boella and van der Torre (2008) offers a BOID inspired conditional rule-based framework to novel effect and introduces the notion of delegation of power to multi-agent systems. This is much like extended social monitoring and aims to achieve the social order using specified regulative norms. Providing agents with normative facts requiring the monitoring of others (actors or norms) not only empowers the individuals but also acts to empower the issuing normative authority and strengthen the system as a whole. This last idea raises two interesting points. First that the strength of a normative system can best be improved by increasing social awareness and responsibility for norms. Second normative systems where actors can react to and effect the planning of others will enable greater utility for the organisation (group or society) as a whole (as well as individual actors in the long run).

2.5 Conclusion

This chapter has introduced some of the key concepts and technologies enabling the use of normative concepts in both mathematics and computing. Many of these ideas will be used to heavily influence the development of key elements within this work. In particular, notions of normative facts and deontic logic will be used as a base on which to build.

To conclude this chapter, this review will examine the extent to which the current state of research into service-based computing, agents and norms fulfils the original research requirements as outlined in Section 1.4.

Service Oriented Architectures and Composition Web services offer numerous description and composition languages on which to build complex distributed systems and engage with foreign actors. Although lacking in some ways, it is clear from the

research contained within this chapter that Web services will offer a solid foundation on which to build a framework for partner selection.

Service Descriptions OWL-S has been singled out by this research as the most promising approach for the description of services. Its strong process model gives a solid base on which workflows of behaviours can also be built. Critically, ideas from profiles can also be borrowed to help give authoritative context to expected processes. It is admitted that OWL-S has its detractors and is not perfect. However, the solid based of research on which it is based and existing tool-kits for its use make it a sound choice for a service description language.

Norm Descriptions and Reasoning Much of this chapter has been devoted to looking at the field of norms and normative reasoning. This is an area which has received a great deal of interest in recent years and a vast expanse of new research is available. It has been identified that there are numerous avenues for future research in this field. A lack of consensus is apparent with many researchers approaching similar problems with differing (and some potentially conflicting) opinions. This thesis chooses to differentiate norms from other social rules such as conventions (Section 2.4.5) and policies (Section 2.4.6), taking the view that norms are mechanisms to drive the behaviour of agents, especially in those cases when their behaviour effects other agents and where norms can be characterised by their prescriptiveness, sociality and social pressure (Lopez y Lopez et al., 2006).

When looking to the area of partner selection and normative agent reasoning this thesis will draw on research discovered in this chapter (Andrighetto et al., 2007a; Kollingbaum and Norman, 2003; Meneguzzi and Luck, 2009; Meneguzzi et al., 2012; Barakat et al., 2012b), by incorporating normative reasoning into a BDI agent system and learning from the field of agent planning. However, despite Barakat et al. (2012b) utilising OWL-S as a mechanism for process definition, this review has yet to see a representation of norms which would fully enable the wide use, dissemination and reuse on the scale of an open Semantic web. Many approaches lack the formalisms, expressive and extendible nature which would be required for large-scale semantic use. Of the languages found, those which have uses in service based computing (OWL-Polar: Section 2.4.6.1) are considered overly ridged and policy based for use in a system which considers flexibility and utility maximisation as key.

Partner Selection In the field of partner and service selection this review has identified a clear gap in the use of normative beliefs to evaluate potential collaborators (or suitable service endpoints). This field is close to that of norm-guided planning and related to that of social monitoring. However, both of these areas fail to exactly fulfil the

requirements for the evaluation of potential interactions (modeled as a specified process) with regards to external partners. This view of norms as defeasible and agents as utility driven has been articulated by researchers before, but it is believed that these notions can be brought together to aid partner and service selection.

In summary, this review confirms the research situation and complication laid out in Section 1.4; that the areas of service selection and normative computing are suitably mature and complementary, but crucially currently disjointed. This chapter has also reiterated the need for service selection techniques to handle complex proposed process models (workflows), that service oriented approaches and service descriptions are mature enough to warrant the investigation of advanced social rule-based selection for the improvement of partner choice.

Technologies which fulfill some of the requirements in Section 1.4.2 have been identified; OWL-S fits in to the Web service stack and can provde a machine readable service description language with well defined Actions and Processes. However, as has been articulated in this summary the state of the art in service selection, Agents and norms is currently disjointed and no single solution exists to answer the research objectives in Section 1.4.

The rest of this thesis will build on the research contained within this chapter and will focus on producing a language for normative process based expectations which is closely tied with well-formed service based notions and technologies (such as OWL-S). Future chapters will work to complete a system which sits among the semantic web stack and enables the selection of services and partners taking into account potentially changing bases of norms which may or may not require violations to be accepted.

Chapter 3

Model

Norm-based agents allow notions of normality and normal behaviours to influence and motivate their plans and actions. Agents utilising norm based approaches allow their plans to be shaped not only by concrete beliefs and rules but also by more fluid notions of what might be ideal (or normal) for a situation. The norms which influence such agents are representations of social concepts existing between agents in any agent society. The same can be said to be true of any norm implemented in any society by any actor. That is to say that (within the scope of this thesis), norms can be treated as social concepts (not hard and fast beliefs), being both role and society specific (norms change between societies and are specific to certain actors fulfilling a set role in the current society).

This work is influenced heavily by agent research. Parallels can be drawn with work by prominent researchers in the field of norm-based agents (such as: Dignum 1999; Broersen et al. 2001; Kollingbaum and Norman 2003; Andrighetto et al. 2007a; Meneguzzi et al. 2012; Alechina et al. 2012; Meneguzzi et al. 2015). The general structure for the model of norms use here has been synthesised using input from Kollingbaum and Norman (2003), with additional modelling for violations taken from da Silva (2008) and Torre (2003). However, it must be noted that this work is not intended as research into service selection in norm-based agents. The purpose of this work is to investigate norm-based service selection as a separate topic and produce a model which can be deployed on any system regardless of underlying architecture.

The abstract model proposed in this chapter has been designed having learned lesson from previous work as highlighted in Chapter 2. The model will have to be expressive enough as to enable an actor to represent norms acting over any combination of activities which might be reasonably expected to occur. Careful consideration has been given to the solution requirements as detailed in Section 1.4.2, so as to ensure that the finished model aids in answering the research question stated in Section 1.4. In particular the model produced here should fulfil (either fully or partialy as a portion of the final solution) the following requirements;

- Provide a machine readable mechanism for the representation of norms to be used in reasoning over process expectations.
- Have ties to existing Semantic Web technologies so as to fit within the existing Web service stack (Section 2.1).
- Provide the ability to reason over both complex and simple sets of service based process model inputs.
- Consume over both complex and simple sets of normative inputs.
- Operate so as to enable the collection of data to answer the governing research questions posed in Section 1.4.

This chapter outlines a model for norm-governed service selection. It includes many notions key to this thesis and forms the basis for much subsequent work.

Scope of These Proposals The field of service selection contains numerous potential avenues for research. As has been outlined in Section 2.3.5, there are many issues surrounding service selection which have not been solved by present research. This thesis proposes a system for service selection (Chapter 5) which aims to solve many of these issues by implementing a range of techniques and advances. However, designing and implementing this architecture in its entirety would be unrealistic in the time frame given for this thesis. As such, one part of this proposed system has been chosen for design and implementation; the normative selection module. This module has been chosen as it fulfils the second research sub-question in that it presents a mechanism for the reasoning of norms with regards to expected processes which enables norm biased services selection to be performed (more details can be found in Chapter 5). This thesis will focus on the use of prescribed norms to describe expected behaviours and how these might be matched to any proposed process definition provided by a service. It should be noted that in no way are the norms modelled here intended to place any concrete contractual or policy-based restrictions on implementing actors. As will be seen throughout this thesis it is the intention of NPE to provide a guide for actors with access to normative facts and mechanisms to reason over facts, to aid with the selection of potential partner services.

3.1 An Abstract Model for Norm-Based Service Selection

The following sections outline the proposed abstract model for norm-based service selection. This model aims to provide a number of new features computing systems can leverage to aid the selection of appropriate services - including:

- An explicit mechanism for the representation of normal behaviours;
- A procedure to note reward or cost as the result of violations of represented norms detected in process model definitions submitted by a potential partner; and
- A mechanism to filter potential services before utilisation (so long as these services provide an expected process definition or workflow).

An actor, wishing to utilise norm-based service selection must be socially aware (endowed with the ability to communicate and interact with other actor and services) and be able to:

- Seek out potential partners (services);
- Receive and interpret process definitions from potential partners;
- Store a set of norms for expected behaviours;
- Process intended colaboration process using internally stored sets of norms, comparing tasks to be expected with tasks proposed as part of a process definition from an external partner; and
- Update internal norms from external sources.

In developing a system for norm-based service selection, all of the above items must be addressed. Initially, a basic abstract model of norm-governed agency is developed to aid the production of the above system and to form the basis of a norm-based system for service selection.

To uniquely identify concepts and ideas within the following model, the identifier label is defined (Definition 1). *Label* is in its own right a concept, however, it is separate from the core concepts involved in this model (norms, roles, actions, exceptions and actors). Within the scope of this abstract model, the superset *Labels* and its subsets(L^{Norms}, L^{Actors}, L^{Roles} and $L^{Exceptions}$) are all finite. This stems from the one-to-one mapping between each label and an element in the model (be it an actor, norm, exception, or role). As there must be a finite set of elements in the model for it to be computable, so there must be a finite set of labels.

Definition 1. A label l is a globally unique identifier which may be understood by all participants in any exchange. Thus, let $LABELS = \{l_1, .., l_n\}$ be a finite set of labels. Then $L^{Actors} = \{l_1^{Actor}, .., l_n^{Actor}\}$ is a finite set of labels for actors where $L^{Actors} \subseteq LABELS$. The set $L^{Roles} = \{l_1^{Role}, .., l_n^{Role}\}$ is a finite set of labels for roles where $L^{Roles} \subseteq LABELS$. The set $L^{Actions} = \{l_1^{Action}, .., l_n^{Action}\}$ is a finite set of labels for actions where $L^{Actions} \subseteq LABELS$. Finally The set $L^{Exceptions} = \{l_1^{Exception}, .., l_n^{Exception}\}$ is a finite set of labels for exceptions where $L^{Exceptions} \subseteq LABELS$.

3.1.1 Norms

Basic normative ideas and the concept of a norm are essential to any norm-based system. This is just as true for the following norm-based service selection model, as it is for norm-based agents. The following model relies on three core norms; Permit, Forbid and Obliged.

- Permissions (Permit) explicitly allow an actor to undertake a specified action;
- Forbid explicitly denies actors permission to undertake a specified action. In this model, forbidden actions can be seen as both a motivator not to undertake an action and as placing a restriction on the set of permissible actions; and
- Obliged actions force actors to undertake an action.

Any action not covered by one of the above norms can be assumed to be unrestricted with regards to this model. Norms cannot be simply combined. They may, however, be nested in certain orders.

In this thesis norms are seen as "soft" constraints on behaviours which are used (in the case of NPE) before interaction to evaluate provided process definitions (as part of a reasoning or simulation process). Thus any discussion of activities or actions to be undertaken refers to activities and actions to be undertaken at some point in the future and under the control of prescribed norms during simulation or pre-engagement reasoning.

Actions are role specific. Actions are activities to be undertaken by an actor fulfilling the role addressed by the current norm. For the purpose of this model, actions are treated as labels only. Labels for actions come from the finite set $L^{Actions}$.

Definition 2. An action A defines a prescribed activity to be undertaken by an actor. Every action A has exactly one actor A^{Actor}. The set $ACTIONS = \{A_1, .., A_n\}$ is a finite set of actions available in the current world. Actions are identified by labels from the set L^{Action}. Within the current world, a set of identified actions exists such that $ACTIONS \subseteq L^{Actions}$. Each action is regulated by a norm. Using the label defined in Definition 4 for the Permission norm, it is now possible to define the following four valid formulations for activities within a norm declaration:

- $perm(a)$ = "Permitted to undertake action a";
- $perm(\neg a)$ = "Permitted to not undertake action a";
- $\neg perm(a)$ = "Not permitted to undertake action a"; and
- $\neg perm(\neg a)$ = "Not permitted not to undertake action a".

Actors fulfil roles within a system. Actors are identified by labels from the finite set L^{Actors}. Norm specifications address roles, not individual actors. However, actors must be defined in order to identify violators in any implementing system.

Definition 3. Actors undertake actions. Actors are identified by labels from the set L^{Actors}. Within the current model, the set $ACTORS \subseteq L^{Actors}$ exists denoting all actors. Every norm specification applies to actions undertaken by one actor to a set of potentially empty other actors.

With the above definitions in place, it is now possible to define a conceptual representation of a norm. This model caters for three core types of norms: permissions (perm), obligations (obli) and forbidden (forb). All other restrictions originate from these core types.

Definition 4. The set $L^{Norms} = \{perm, obli, forb\}$ is the set of labels used to identify the three core norms: Permission (perm), Obligation (obli) and Forbidden (forb). Having acquired labels for the three core norms used within this system, it is now possible to define a few straight forward rules for the interaction of norms:

- $perm(a) \equiv \neg forb(a)$
- $perm(a) \wedge perm(b) \equiv perm(a \wedge b)$
- $obli(a) \equiv forb(\neg a)$
- $\neg perm(\neg a) \equiv obli(a)$
- $\neg perm(a) \not\equiv perm(\neg a)$ $(also\, true\, for\, obli\, and\, forb)$
- $perm(a \rightarrow b) \not\equiv perm(b \rightarrow a)$ $(permitted\, a\, then\, b)$
- $obli(a \rightarrow b \wedge c) \equiv (obli(a) \rightarrow obli(b \wedge c))$

Thus a norm can the be defined as a tuple:

Definition 5. A norm is a tuple $N = \langle n, l^{Actions}, R, c, d, e \rangle$

Where:

- $n \in L^{Norms}$ (the label attributed to the current type of norm: perm, obli, or forb);
- $l^{Actions}$ is a specified set action labels where each $l \in L^{Actions}$;
- $R \in L^{Roles}$ is the role played by the actor which is controlled by the given norm;
- $c \in L^{Conditions}$ is a condition under which the action may be initiated and the norm activated. The boolean condition "true" can be used to denote a norm which will always be triggered given a matching role and activity;

- $d \in L^{Deactivations}$ is a deactivation condition, under which an active norm will be deactivated. This condition has no significance when the norm is inactive. The boolean condition "false" may be used to signify a norm which once activated will always hold for a matching role and activity; and
- $e \in L^{Exceptions}$ is an exception to be triggered if the norm is violated.

Definition 6. Every set of norms (NORMS) comprises a set of norm specifications such that $NORMS = \{N_1, .., N_i\}$ where every norm N must be a valid norm according to the above syntax and no conflicts may occur between norms in any individual set (of norms).

Conflicting norms will be explained in detail in Section 3.1.4.

With the aforementioned definitions in place, norms can be explicitly defined using formal semantics. For example, the normative statement in Figure 3.1 expresses that: Actors fulfilling the role R are obliged to undertake the action a when the precondition c is matched. Breaking this norm will trigger exception e. The norm will stay active so long as the deactivation condition d is not true. Such a formalisation allows this abstract system to interpret the given norm as a given belief and apply it to a plan using a provided matching algorithm.

$$obli(a, R, c, d, e)$$

Figure 3.1: Example Norm 1

Norms may also be defined with negated action statements, such as in the case of Figure 3.2 which states: actors fulfilling role R are permitted not to undertake the action a when there are matching preconditions c. In this example, it is likely that no exception would be thrown as permission is a relatively open normative preposition (permission not to act does not entail forbidden to act).

$$obli(\neg a, R, c, d, e)$$

Figure 3.2: Example Norm 2

The previous two examples demonstrate how norms can be formally represented. However, they lack the detail of a "real world" example. Figure 3.3 gives a more realistic perspective of what a norm might look like. In this case, the norm specifies that: actors fulfilling the role "student" are forbidden from undertaking the action "read_newspaper" whilst the precondition "in_lecture" holds true. This norm becomes invalid when the deactivation condition "lecturer_exit" evaluates to true. If this norm is violated the exception "inappropriate_reading_exception" is thrown.

$forb(read\_newspaper, student, in\_lecture, lecturer\_exit, inappropriate\_reading\_exception)$

Figure 3.3: Example Norm 3

3.1.1.1 Triggering

When processing a process model the underlying normative architecture generates a plan containing a sequential list of actions and states, along with sets of roles and initial beliefs (the underlying processes will be explained in detail in Section 5). This plan is then checked against the normative model created in the architecture. Norms are triggered when the plan features an actor fulfilling role R, undertaking action a, where R and a match the required role and action of a norm in the system and the preconditions of that norm are true. If the role and preconditions all match, then the norm is triggered. Upon triggering, the current norm is moved to the ACTIVE set (Definition 8). If the current norm is violated, its exception clause is triggered (Section 3.1.3).

Definition 7. Each norm has an activation condition c such that $c \in L^{Conditions}$. Conditions can be evaluated by an implementing system to evaluate either true or false; $c = \{true, false\}$. When an activation condition is met, the norm is the eligible for activation.

Definition 8. $ACTIVE$ is the set of all active (triggered) norms, such that $ACTIVE = \{N_1, .., N_i\}$. It must be true that $ACTIVE \subseteq NORMS$ so that any norm in $ACTIVE$ must also exist in the superset $NORMS$; the global set of all norms in the system.

3.1.1.2 State and Action Triggered Norms

When a (simulated) system is monitored via states, norms may be triggered by observable changes in the model or in the global state. A state reflects all the current information the system has about beliefs and the surrounding environment in which it lives (for example for a BDI agent this will be reflected in the state of the agents' beliefs, desires and intentions). Norms stored by an implementing system contain a representation of a state or partial state. Transitioning between states, the new state is captured and the system searches through its norm store checking if this new state matches any of the conditional states in any norm. If a norm with a state the same as the current one is found (or is a subset of the current), the attached norm is triggered. Logically this approach is grounded in Situation Calculus (McCarthy, 1969) and frame based logic. In a state-based system, it is expected that the reasoner will only have a partial view of the global state and any representation of state held by a norm can also be partial. State-based triggering of norms is useful in situations where states are simple, where an actor might receive large amounts of external data, or an actor is waiting on external stimuli.

Action triggered norms rely on actors undertaking actions or detecting actions to be triggered. When an actor undertakes an action the action is detected either by the actor itself or a third party. Norms contain triggering actions within their headers. When an action is detected, the actor scans its list of norms looking for a norm with a matching action in its header. If a matching action is found in the norm repository the associated norm is triggered. Action triggering can be useful for planning applications as well as for environments where the state of the environment may be complex. Action triggering can be supplemented with situation information by using preconditions within the norm (if an actor is in situation x and undertakes action a for example), allowing for triggering to be more precise.

This model utilises action based triggering, with the ability to monitor the situation through an optional precondition. This approach was chosen as the system is essentially a planner and no actual situation data will be available during running.

3.1.1.3 Deactivating Norms

As has been described in the two preceding subsections, once a norm has been triggered it is placed into the set $ACTIVE$. The set $ACTIVE$ contains all triggered norms. Only these active norms can be applied in the validation of a plan. As well as an activation condition, norms also contain a deactivation condition. This value is ignored until the norm is in the set $ACTIVE$ (conversely the value of the activation condition is ignored once the norm is in the set $ACTIVE$). Once active, the deactivation condition can then be evaluated. If the deactivation condition is evaluated to "true", the norm is then removed from the set active and becomes deactivated (and the activation condition again starts to be tested). The deactivation condition may never be set to boolean "true" (as that would invalidate the norm), it may, however, be set to boolean "false" to signal a norm which once active will always hold for a given combination of role and action.

Definition 9. Each norm has a deactivation condition d such that $d \in L^{Deactivations}$. Deactivating conditions can be evaluated by an implementing system to evaluate either true or false; $d = \{true, false\}$. When a deactivation condition is met for an active norm, the norm is the eligible for deactivation.

3.1.2 Roles

Within any system, there is a finite set of roles. In this model roles are identified uniquely using labels from the finite set L^{Roles}. Norms are role specific - every norm must have at least one role under which it can be triggered. At the most basic level norms can be seen as characteristics of their parent role, limiting the actions an actor in that role can undertake.

Definition 10. Roles are identified by labels from the set L^{Roles}. Norms are role sensitive. Each actor holds a non-empty set of roles such that $ROLES \subseteq L^{Roles}$. Norms apply to actors, as a result, every norm applies to a non-empty set of roles.

3.1.3 Violations

During reasoning or simulation of any action as part of a provided process definition, violating an active norm (or more accurately, the detection of a potential violation) causes an exception to be triggered. Only norms currently in the set $ACTIVE$ may be violated and thus trigger exceptions. Exceptions trigger new norms or force actors to undertake pre-set actions. Exceptions can alternatively take the form of a numeric penalty denoting the severity of the violation. Any norm can have a maximum of one exception (although exceptions can be nested and so can norms). Exceptions are identified using labels from the finite set $L^{Exceptions}$. A norm specification will hold a single exception $e \in L^{Exceptions}$. Triggering an exception may or may not force the abandonment processing loop.

Definition 11. Exceptions are identified by labels from the set $L^{Exceptions}$. A norm specification holds a single exception such that $e \in L^{Exceptions}$.

Where norms are nested, violations can cause exceptions to be triggered in one norm, which are then passed to those at higher levels. In nests featuring norms with no exceptions (null exceptions only, not exceptions which are empty), violations will be passed up to the parent norm until a norm with an exception clause is found and triggered. Or more simply, norms with no exceptions defined, once violated, throw a default exception which halts all processes and alerts norms at higher levels, until one which can handle violations is found.

In later chapters, it will be shown that exceptions are a powerful tool for assigning a cost to normative violations. One of the key ideas behind the approach advocated by NPE is that actors should use norms to help reason over the cost of achieving a goal and that adherence to normative facts should be as a result of reasoning over the costs of violations, rather than as a result of the norm emergence its self. As an actor reasons over a proposed process model with regards to the norms he/she holds internally, any (non-terminal) norm violation will be recorded and the cost summed to form a final utility score for the process. This can be used to add weight to an actors' choices when selecting between similar services.

3.1.4 Consistency of Norms

For any norm based service selection model to be successful, the norms defined, stored and handled by the system must be consistent. Consistency applies to both the norms

and the actions which the norms regulate. To ensure that all norms in use are consistent, it is imperative that the initial norm base is checked at start-up and re-checked every time a new norm is added. Norms need to be consistent individually with regards to the actions they regulate and consistent with regards to any interdependencies which may exist between multiple norms. Figure 3.4 would cause a norm base to become inconsistent and must never be allowed to occur (N1 permits action a at the same time as N2 forbids action a for the same role and precondition. So if an actor fulfilling the role R were to undertake action a despite N1 permitting it, an exception would still be thrown, as N2 would be violated).

N1. $perm(a, R, c, d, e1)$
N2. $forb(a, R, c, d, e2)$

Figure 3.4: Two Norms Inconsistent With Each Other

Consistency checking for norms in a norm base of any model should be assisted by an initial process of normalisation for any normative statements included preventing the same norm being defined in several ways and to ensure that there are several consistent methods for defining the majority of required norms. This process should simplify all the norms in the base to a number of default types (see the following subsection). This process makes consistency checking easier to fulfil and simplifies the norm base.

It is important to note that there are two types of consistency: local consistency and global consistency. Norms are required to be locally consistent (every norm in a local set must be consistent with every other). No norm can be added which might cause the local set of norms to become inconsistent. Globally, norms within the local set will inevitably be inconsistent with regards to every other possible norm. Global consistency is achieved through the use of namespaces and other identifiers.

This thesis will not discuss normative conflicts any further as they fall outside of its scope. It is assumed that any norm bases used for simulations or testing will have previously been tested for logical constancy and deemed to be fit. This action is seen as a precursor to the testing of process models using norms (which will occur during service selection). For information as to techniques and methods which might be applied to solve issues of normative consistency, the reader is directed to existing work on this theme (Vasconcelos et al., 2009; García-Camino et al., 2008; Vasconcelos et al., 2012; Gasparini et al., 2015).

3.1.4.1 Normative Equivalence and Syntactic Transformations

The abstract model for norm-based service selection outlined here specifies three distinct classes of norms: perm, obli and forb. It also enables actions to be negated within the norm specification (as in Figure 3.2) and norms themselves to be negated. This approach has been chosen to aid transparency by reducing the need for negated actions where

possible. The motivation behind this decision was that if this abstract model is eventually represented (for initial specification and transit) as XML (or another Semantic Web "friendly" language), removing the need for as many negated actions as possible would make for data which would be easier to model, search and author by hand. As a result, it may be possible to represent the same normative statement in several ways. Most notably, the two statements in Figure 3.5 are identical. Both state that the actor is forbidden from undertaking action "a".

N1. $obli(\neg a, R, c, d, e)$
N2. $forb(a, R, c, d, e)$

Figure 3.5: Equivalent Norms

As has been mentioned in Section 3.1.4, prior to consistency checking, all norms inserted into the system will have to undergo a process of normalisation to ensure that all of the norms are represented in an agreed manner. This requires a set of syntactic transformations to be applied. The following transformations are suggested as being appropriate (Figure 3.6). It should be noted that the transformations applied determine the exact norms being inserted into the model. As such the nature of these transformations will probably be determined by the requirements of the underlying system. The transformations listed in Figure 3.6 should serve as general guidance.

T1. $\neg perm(a, R, c, d, e) \equiv forb(a, R, c, d, e)$
T2. $\neg perm(\neg a, R, c, d, e) \equiv obli(a, R, c, d, e)$
T3. $\neg forb(a, R, c, d, e) \equiv perm(a, R, c, d, e)$
T4. $\neg forb(\neg a, R, c, d, e) \equiv perm(\neg a, R, c, d, e)$
T5. $obli(\neg a, R, c, d, e) \equiv forb(a, R, c, d, e)$
T6. $forb(\neg a, R, c, d, e) \equiv obli(a, R, c, d, e)$

Figure 3.6: Syntactic Norm Transformations

The analysis conducted to give the general transformations listed in Figure 3.6 also unearthed a number of norms which in an open system would have lower significance than others. 3 examples are shown in Figure 3.7 and include norms which articulate beliefs such as N2: "not having the obligation to do or not to do something". How these norms are handled will depend on the underlying system. However, it could be the case that such norms are simply removed from the input due to their insignificance.

N1. $\neg obli(a, R, c, d, e)$
N2. $\neg obli(\neg a, R, c, d, e)$
N3. $perm(\neg a, R, c, d, e)$

Figure 3.7: Insignificant Norms

3.2 A System for Norm-Based Service Selection

Definition 12. Services are identified by labels from the set $L^{SERVICE}$. In so far as an actor wishing to consume a service is concerned, a service is a tuple; $SERVICE = \langle l^{Service}, SDESC, GROUNDING \rangle$

Where:

- $l^{Service}$ is an internal identifier for the service;
- $SDESC$ is a service description for the service (defined in Definition 13); and
- $GROUNDING$ is the concrete grounding for the external service.

Given the above abstract model a module, for norm-based service selection can now be formulated for implementation in generic agent-based systems. This module is based on the above abstract model, with the notion of a $SERVICE$ added to represent external services being considered by the implementing actor. Critical to an actor's representation of any external service, is the service description entity ($SDESC$). Defined in Definition 14, a service definition holds two models containing information about the capabilities and requirements of the service. The profile model describes what it is the service "does". The process model describes how it intends to undertake the tasks detailed in the profile model and what steps (if any) are required to be taken by the implementing actor to ensure the correct execution of a required task. At this point, as this is still an abstract system, no underlying service description language is defined. It is also possible that the profile model for any given service may have been created by the implementing actor itself. This, for example, may occur if the actor has received a list of service process models (without profiles).

The $GROUNDING$ provides a concrete location and interface over which the service can be interacted with. The structure of the $GROUNDING$ is outside of the scope of this thesis, it is expected that it would be implemented as WSDL. The $GROUNDING$ has been added to the tuple $SERVICE$ to show where grounding information would be stored during deployment any final NPE solution. It is possible that the grounding might be used during testing to hold information about any communication protocols used.

This research takes the position held by Lomuscio et al. (2008) and others that service description can form contracts for the representation of proposed actions to be undertaken during interaction with an advertising service. As such the norms modelled by NPE are not contracts for behaviours; they are "soft" restrictions which may be applied to a simulation of a proposed process model (as contracted by the service description). In this way, partners can be chosen by rating provided service descriptions.

Definition 13. Service descriptions are identified by labels from the set L^{SDESC}. An actor wishing to use an external service holds a set of service description for such services such that $SDESCS \subseteq L^{SDESC}$.

Definition 14. A single service description $SDESC$ is a pair such that; $SDESC = \langle PROFILE, PROCESS \rangle$

Where:

- $PROFILE$ represents the supplied profile model for this service;
- $PROCESS$ represents the process model for this service.

It is this service description which the norm-based selection module uses to validate a service. The service profile model is used to choose which norms are applicable, then the process model is validated against these norms. This norm validation process as described in pseudocode in Figure 3.8.

```
pubic Hashmap gradeServices(Services s){
        Hashmap results = new Hashmap();

        while(s.next()){
                Norms n = fetchNorms(s.Current.Profile);
                if(n == null){
                        results.add(s.Current.ID, Result.NullNorms);
                        continue;
                }

                Result r = validate(s.Current.Process, n);
                results.add(s.Current.ID, r);
        }

        return results;
}
```

Figure 3.8: Norm Based Service Grading Pseudocode

The pseudocode in Figure 3.8 shows an abstract module for service grading. Service grading forms the core of this norm-based service selection architecture. The validation method, which is explained in detail in Section 5.1, in itself forms the most important and complex part of the whole service selection process. Scoring and the final selection procedure will be covered in Section 5.1 and Section 5.2 respectively. As a reference, Figure 3.9 shows how the norm-based services selection module functions as a whole and where the grading module shown in Figure 3.8 lies. Please note that the method fetchRequirements() step is not part of this architecture but is included for readability. It should be noted that both these pieces of pseudocode have been created using the concept of an abstract system and are only the start of any concrete implementation.

```
public Service selectService(Task t){
        Requirements req = fetchRequirements(t);
        Services s = fetchServices(req);

        Hashmap results = gradeServices(s);
        ServiceID id = selectService(results);

        return s.find(id);
}
```

Figure 3.9: Norm Based Service Selection Pseudocode

```
public Results validate(Process p, Norms n){
        Model m = createModel(p);
        NormStore active;
        ExceptionStore exceptions;
        Results r;

        while(m.next()){
                try{
                        checkDeactive(m.Current,n) = dn;
                        if(dn != null){
                                active.remove(dn);
                        }
                        checkActivate(m.Current,n) = an;
                        if(an != null){
                                active.add(an)
                        }
                        checkViolations(m.Current,n);
                }catch(Exception e){
                        exceptions.add(e);
                        r.update(e);
                }
        }
        return r;
}
```

Figure 3.10: Norm Based Validation Pseudocode

3.3 Conclusion

This chapter introduces a model of norms which is has been heavily influenced by the existing normative research detailed in Chapter 2. These norms bring together well-formed notions from role and utility-based approaches. Actions are bound through the utilisation of the constraints obligations, permissions and forbidden. This model of norms is then used to form a fuller abstract model for norm based service selection.

The model for normative service selection enables actors to use the aforementioned concepts of normative facts to reason over activities proposed by others. To enable this reasoning, actors are likely to simulate or process service descriptions while taking into account any held normative beliefs. In advocating the use of this model this thesis does not suggest that agents use this directly for planning; indirect use is preferred with utility based calculations about the proposed actions of others produced and to used to influence planning.

The abstract model introduced within this chapter is proposed as the start of an answer to the first research sub-question posed in Section 1.4. To this point, it has been shown how a model for norms can be produced for the general representation of norms with regards to actions and how such a model can be used in reasoning. The second half of this question relating to use by semantically aware actors will be explained in the proceeding chapter. It has previously been shown in Chapter 2 that there is currently a lack of a full abstract model for the easy evaluation of proposed actions by (potentially static or socially unaware) intelligent agents. Having proposed this model the next few chapters will outline the generation of a concrete language for the representation of norms in semantically rich connected environments.

In summary, the model for norms provided by this thesis defines a tuple N following Definition 5 which is reiterated in Figure 3.11

A norm is a tuple $N = \langle n, l^{Actions}, R, c, d, e \rangle$
Where:

- $n \in L^{Norms}$ (the label attributed to the current type of norm: perm, obli, or forb);
- $l^{Actions}$ is a specified set action labels where each $l \in L^{Actions}$;
- $R \in L^{Roles}$ is the role played by the actor which is controlled by the given norm;
- $c \in L^{Conditions}$ is a condition under which the action may be initiated and the norm activated. The boolean condition "true" can be used to denote a norm which will always be triggered given a matching role and activity;
- $d \in L^{Deactivations}$ is a deactivation condition, under which an active norm will be deactivated. This condition has no significance when the norm is inactive. The boolean condition "false" may be used to signify a norm which once activated will always hold for a matching role and activity; and
- $e \in L^{Exceptions}$ is an exception to be triggered if the norm is violated.

And: $NORMS = \{N_1, .., N_i\}$

Figure 3.11: Model for Norms

Chapter 4

Ontology

Developing systems that make use of norm-based techniques during service selection requires more than the abstract model outlined in Chapter 3. For any system to be viable and useful, developers will require both a concrete implementation of the abstract model (Chapter 5) and a language in which to define and store any norms to be implemented. It is true that a system could be created without a base (external) language for norm definitions and only basic internal representations. Such a system, however, would have a very limited appeal in wider applications. Creating a language in which norms can be represented and interpreted by both computers and humans enables the rapid creation of a norm base for any existing or new system. For any language to be viable for the construction of a norm base it should be:

- Machine readable;
- Human readable;
- Transportable;
- (Lightweight);
- Based on existing standards.

This chapter outlines a language for the description of norms in regards to expected processes. The language for Norm-based Process Expectations (NPE), aims to provide a mechanism to easily store sets of norms which relate to expected practices during an interaction. This language has been built on a number of existing protocols and aims only to fulfil the role of describing norms for expected practices, not real-time detected activities.

This language has been produced as no currently available ontology or language construct fulfils the research sub-question "How can norms be represented so as to best relate

to the proposed process description of semantically aware actors?". This research has identified that a model of norms can be synthesised out of existing normative research. However, a disconnect exists between this work and work undertaken for the selection of services with rich semantic descriptions. This language fulfils this initial research goal and provides a solid base on which a reasoning system (with regards to expected processes) can be produced.

In producing NPE, two associated languages have been produced: an initial language (described in the following section) and an XML variant. The design was split into two parts to make any explanations easier and to improve understanding of key concepts.

4.1 The Initial Language

The initial NPE language is Prolog-based and intended to be used to demonstrate the concepts behind NPE. Prolog was chosen as its syntax already consists of sets of clauses (rules and goals) which map onto norms and concepts as outlined in Chapter 3. The Prolog used to describe NPE is heavily influenced by the language ESPLEX (information on which can be found in Section 2.4.7.2).

Alongside every Prolog definition is the corresponding definition in Prolog-based NPE. Prolog-based NPE is the first of the two languages designed as part of this work. Following strictly the definitions for a norm prescribed in Chapter 3, Prolog-based NPE is essentially a distilled version of the pure Prolog representation of a normative action. A complete normative statement can be represented as a single Prolog-based NPE rule.

4.1.1 Example Situation

An example situation will be used to illustrate the type of actions and processes norms defined in an NPE document might be expected to govern. For the purposes of this thesis, it was decided that a realistic example of a service providing an interface to a business such as a shop should be used. The following example is based on a fictional B2C service providing an interface to an existing Web-based store. The store will be named Frango and the corresponding service - FrangoOrderService. This example will be used repeatedly throughout this thesis. It can be assumed that there are numerous companies in the same market as Frango and that they also provide Web services similar to FrangoOrderService.

The process and profile models for this example have been created with reference to the Congo example OWL-S service from DAML (Narayanan, 2005) and the Amazon book price OWL-S documents from Mindswap (Mindswap, 2004).

Figure 4.1: Basic FrangoOrderService Processes and Execution Order

4.1.1.1 Profile Model

Frango is defined as an E-business implementing a Web service interface to enable machine to machine transactions. This service is referred to as the FrangoOrderService. The FrangoOrderService consists of a service invocation followed by a number of interactions between Frango and a consumer.

To enable the description of simpler scenarios, a cut-down version of the FrangoOrder-Service is also provided. Referred to as FrangoExpressOrderService, this service only requires one action by the consumer.

4.1.1.2 Process Model

The FrangoOrderService consists of a number of sub-processes composed into one business interaction. The required processes are shown in Figure 4.1. For an order to be successful, each step must be completed. A complete description of the FrangoOrderService can be found in the form of the OWL-S document frango.owl in Appendix A.1.1.

The process model for the FrangoExpressOrderService contains only a service invocation and corresponding reply. All of the information required to fulfil the purchase of a book is sent from the consumer to the FrangoExpressOrderService as part of the service invocation.

4.1.1.3 Normalised Pseudocode

Figure 4.2 shows the FrangoOrderService process model in normalised pseudocode (ALGOL 60 based). It should be noted that this process shows the steps as expected by the consumer and that full details of each step (and message) can be found in the document frango.owl in Appendix A.1.1.

```
1. SEND message #ProductSearch with product details to FrangoOrderService
    endpoint

2. IF
   A) RESPONSE is #ProductOutOfStock
      THEN
      1. END interaction
   B) RESPONSE #ProductInStock AND ProductID
THEN

3.
   A) SEND message #PutItemInCart with ProductID to FrangoOrderService endpoint
   B) RESPONSE #ProductInCart AND CartID


4. SEND message #Login with login credentials to FrangoOrderService endpoint

5. IF
   A) RESPONSE is #InvalidLogin
      THEN
      1. END interaction
   B) RESPONSE is CustomerID
THEN

6. SEND message #ShippingDetails with shipping details AND CustomerID AND CartID
    to FrangoOrderService endpoint

7.
   A) SEND message #PaymentDetails with payment details AND CartID to
    FrangoOrderService endpoint
   B) RESPONSE #RequestForConfirm

8. SEND message #ConfirmOrder with CustomerID AND CartID to FrangoOrderService
    endpoint

9. IF
   A) RESPONSE is #OrderShipped
      THEN
      1) END interaction
   B) RESPONSE is #OrderFailed
      THEN
      1) END interaction
```

Figure 4.2: The FrangoOrderService (Normalised Pseudo-Code)

4.1.2 Basic Syntax

Prolog is a declarative programming language based heavily on formal logic. Declarative languages are built up of sets of relations between facts and rules. In Prolog, these rules and facts are very weakly typed, enabling programmers to create powerful descriptive programs at the expense of having to be obviously careful how relationships are defined. Relationships in Prolog may be binary as in Figure 4.3 which roughly equates to "the capital of the UK is London". In this relationship the relationship name is "has_capital", the subject is "uk" and the object is "london". It should be noted that the "." character is important, as it terminates the relationship (fact).

Ternary relationships are also possible. One common example is that of Figure 4.4 which equates to: "Fred sends a message to Tim". Here again, the relationship is terminated with a ".".

```
has_capital(uk,london).
```

Figure 4.3: A Binary Relationship in Prolog

```
sends(fred,tim,msg).
```

Figure 4.4: A Ternary Relationship in Prolog

Multiple clauses can also be held under the same relationship, such as in Figure 4.5. As will be demonstrated later in this section, such clauses can be seen as logically dis-joined.

```
has_capital(uk,london).
has_capital(spain,madrid).
has_capital(france,paris).
```

Figure 4.5: A Multiple Binary Relationships in Prolog

Relationships and facts are on their own useful but do not allow programmers to produce suitably descriptive programs. For a program to be useful the notion of a rule needs to be introduced. The basic structure of a Prolog rule is that of a head and a body separated by the character ":-" (Figure 4.6). In Prolog, the ":-" roughly evaluates to a logical *if*, such that head is true if the body is true. For example in Figure 4.3, the relationship "the capital of the UK is London" is already true, if a programmer were to add a body with the relationship "above_water(london)" then the rule "London is the capital of the UK if London is above water" would be created.

```
head :- body
```

Figure 4.6: The Structure of a Prolog Rule

Rules can be made more complex by the inclusion of multiple subgoals (conjoined goals) within the body. Such relationships can be equated to logical conjunction, for example, Figure 4.7 shows the rule: "a person will pay a builder *if* the builder builds the person a house *and* the house has good build quality". Note that in this example, relationships within the body are separated by the character "," which equates to logical conjunction.

```
pays(person,builder) :-
                        builds(builder,person,house),
                        good(house,build_quality).
```

Figure 4.7: A Prolog Rule With Multiple Subgoals

Prolog also supports disjunctive sets of rules similar to those in Figure 4.5. Figure 4.8 shows a number of rules which disjoin to make the statement: "a person is British if that person is Welsh *or* that person is English *or* that person is Scottish *or* that person is Northern Irish". Again, note that the rule statements are separated by the character ".".

```
british(person) :- welsh(person).
british(person) :- english(person).
british(person) :- scottish(person).
british(person) :- northern_irish(person).
```

Figure 4.8: Multiple Prolog Rules

4.1.3 Prolog-based NPE

Pure Prolog is adequate for the description of rules evolving norms. A wealth of stable Prolog interpreters and mature programming standards make testing rules easy and reliable. There are, however, a number of drawbacks to using pure Prolog to describe the normative rules outlined in Chapter 3. Most notable are issues surrounding the readability of highly normalised Prolog and the lack of exception handling/reporting in Prolog. Both of these issues are important when it comes to representing norms which are likely to be authored by hand and which are likely to be violated when implemented.

To overcome the issues surrounding the implementation of norms in Prolog, a Prolog-based variant of the NPE language is proposed. Prolog-based NPE is as the name suggests, based on the Prolog language and its basic constructs. Norms are specified using Prolog facts and tuples including all the relevant information. Prolog-based NPE improves on the pure Prolog representation by increasing readability; a single Prolog fact for each norm required to be specified. Correctly formatting each fact can improve readability, without the need to change the structure of any norm. An example of correct formatting and the basic structure of Prolog-based NPE can be found in Figure 4.9. As well as improving readability, Prolog-based NPE also adds the notion of exceptions to facts. Correct processing of exceptions will require a specially modified Prolog interpreter. However, the syntax of Prolog has been adhered to as best as it can be so as to enable a new processor to be synthesised out of an old Prolog one, as well as to enable syntax checking without the need for a new processor.

Prolog-based NPE is the starting point in the investigation of how to represent norms in both machine and human readable form. It is, however, not perfect; the resultant Prolog facts are large and non-normalised. If designed without care, multiple facts could represent the same normative statement (Section 3.1.4). However, many of these problems could be overcome with the careful design of facts.

The model defined in Definition 5 is the basis for Prolog-based NPE. The Prolog facts are ordered in exactly the same form as this model (Figure 4.9). NPE facts are designed to be used with the abstract model outlined in Chapter 3.

```
<npe_document> ::= <IDENTIFIER> "{" <norm_fact> "}".

<norm_fact> ::= ( "perm"
                | "obli"
                | "forb"
                )
                "("
                   <action> "'"
                   <roles> ","
                   <preconditions> ","
                   <deactivation_conditions> ","
                   <exception>
                ")."

<action> ::= "#"<IDENTIFIER>

<roles> ::= "[" <role> ( "," <role> )* "]"

<role> ::= <IDENTIFIER>

<preconditions> ::= <condition>

<deactivation_conditions> ::= <condition>

<condition> ::= <prolog_predicate>

<prolog_predicate> ::= <atom> | <atom>"(" <term_list> ")"
<term_list> ::= <term> | <term_list> "," <term>
<term> ::= <variable> | <structure>
<structure> ::= <atom>"(" <term_list> ")"
<atom> ::= <variable>
<variable> ::= <text> (<text>)*
<text> ::= <char> | <charupper> | <digit> | <symbol>
<char> ::= a | b | c | d | ... | y | z
<charupper> ::= A | B | C | D | ... | Y | Z
<digit> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
<symbol> ::= _ | #

<exception> ::= "throw(error(" <IDENTIFIER> "))"
```

Figure 4.9: Prolog-based NPE (In BNF Form)

4.1.3.1 Permitted

```
perm(edit_news,user):-
        cond(user,admin),
        act(edit_news).
act(edit_news):-
        cond(user,loggedin).
```

(a) Prolog

```
perm(
        edit_news,
        admin,
        loggedin,
        notloggedin,
        throw(error(accessviolation))).
```

(b) Prolog-based NPE

Figure 4.10: An Example of the Norm Permit

Actions permitted within a system are exposed in NPE as facts with the name "perm". Such facts represent actions which are permitted when the prescribed role and precondition match a current set of beliefs or observations. Permitted facts become deactivated whenever the prescribed deactivation condition is met (for any matching role). An example of a permitted norm described as both Prolog and Prolog-based NPE can be found in Figure 4.10.

4.1.3.2 Forbidden

```
forb(ask_user_password,seller):-
        act(ask_user_password).
act(ask_user_password):-
        cond(msg,using),
        cond(encryption,using,plaintext).
```

(a) Prolog

```
forb(
        ask_user_password,
        seller,
        plaintext,
        ssl,
        throw(error(unsecured_error))).
```

(b) Prolog-based NPE

Figure 4.11: An Example of the Norm Forbidden

Actions forbidden within a system are exposed in NPE as facts with the name "forb". Such facts represent actions which are forbidden when the prescribed role and precondition match a current set of beliefs or observations. Forbidden facts become deactivated (permitted) whenever the prescribed deactivation condition is met (for any matching role). An example of a forbidden norm described as both Prolog and Prolog-based NPE can be found in Figure 4.11.

4.1.3.3 Obligated

```
obli(send_goods_to_buyer,seller):-
        cond(buyer,has_paid),
        act(send_goods_to_buyer).
act(send_goods_to_buyer):-
        cond(goods,in_stock),
        cond(order,not_complete).
```

(a) Prolog

```
obli(
        send_goods_to_buyer,
        seller,
        goods_in_stock,
        order_complete,
        throw(error(goods_still_to_send))).
```

(b) Prolog-based NPE

Figure 4.12: An Example of the Norm Obligated

Actions obligated to be undertaken within a system are exposed in NPE as facts with the name "obli". Such facts represent actions which are obligated when the prescribed role and precondition match a current set of beliefs or observations. Obligated actions within facts become deactivated whenever the prescribed deactivation condition is met (for any matching role). An example of a "obli" norm described as both Prolog and Prolog-based NPE can be found in Figure 4.12.

4.1.4 Example

At this point, an example is required to clarify the applicability of Prolog-based NPE to solving the problem of norm representation. A number of examples have already demonstrated the various features of Prolog-based NPE (Figures 4.10, 4.11 and 4.12). The following examples aim to demonstrate how NPE can be used in real-world applications and demonstrate how it can be used as part of a system to evaluate a service description.

4.1.4.1 Basic

An initial single norm example, grounded in the real-world, has been created. Figure 4.13 shows a norm which governs a basic everyday shopping interaction. The norm states that a shopper is obliged to pay (#pay) for goods when they hold the belief that they have finished their shopping (#shopping_complete). This norm is deactivated when an exchange of credit has been complete (#transfer_complete). If this norm is violated a failure_to_pay_exception is thrown (in the real-world, could possibly also result in the action #police_called). This norm roughly equates to having to pay for goods when shopping is finished. For an explanation of the use of the "#" symbol and keywords see Section 4.1.5.

```
obli(
        #pay_for_goods,
        Shopper,
        in(#shopping_complete),
        once(#transfer_currency),
        throw(error(failure_topay_exception))).
```

Figure 4.13: An Example of the Norm Obligated, Grounded in the Real World

4.1.4.2 Complex

The preceding example is over simplified for use in a semantic Web situation. This more complex example aims to demonstrate how NPE could be used to evaluate a formally defined service process model. This example is based on the Frango example situation as defined in Section 4.1.1 and Appendix A.1.1. Figure 4.14 demonstrates a partial Prolog-based NPE document consisting of three norms relating to the Frango example. One norm of each type (perm, obli and forb) is demonstrated to demonstrate how each might work. All of the norms are applicable to actions defined in the OWL-S process model in Appendix A.1.1. Essentially, the three work together to state that during the interaction a user is obligated to put an item into their shopping cart, a seller is permitted to load the users profile if one exists and once it is loaded this permission no longer holds (although once the norm is deactivated this does not necessarily mean that

the action is forbidden), finally the user is forbidden from adding any more items to the cart once the transaction is complete. These norms have all been created by studying the OWL-S document for Frango in Appendix A.1.1. The process model for Frango will evaluate to valid if checked against these norms. Violation of any of these norms would result in exceptions being thrown and the belief being set that the service violates normal expectations for the process being undertaken.

```
obli(
        #PutItemInCart,
        user,
        in(#Buy_Sequence),
        once(#PutItemInCart),
        throw(error(no_items_in_cart_execption))).

perm(
        #LoadUserProfile,
        seller_service,
        if(#ProfileExists),
        once(#LoadUserProfile),
        throw(error(signin_exception))).

forb(
        #PutItemInCart,
        user,
        in(#Finalize_Buy),
        false,
        throw(error(transaction_complete_exception))).
```

Figure 4.14: An Example of a Set of Norms

4.1.5 Keywords

Prolog-based NPE maintains, where possible, valid Prolog syntax. This limitation aims to ensure that interpreters can be easily created and the testing of documents is simple. On the whole, this objective has been achieved successfully, however some changes are necessary to facilitate higher degrees of coupling with process descriptions and handling of complex scenarios. Most notable is the inclusion of identifiers preceded by the character "#" linking NPE facts to regions or tasks in an underlying process model. These identifiers can be processed normally (handled as literals or ignored) by many standard Prolog interpreters. A specialised NPE processor would then make use of these identifiers for linking to a given process model. A set of specialised queries has been added for condition processing. Again these are normal Prolog statements, but they have special meaning in an NPE setting. As with the above identifiers, these queries would be passed to a specialised NPE interpreter for further evaluation. Most of these queries relate to evaluating the contents of current states, the agents belief base, or the current process model being evaluated. These keywords include:

- in(#region)

  - A query to test if at this time in the evaluation of the given process model, the agent is "in" the given task "#region".

- Once(#task)
  - True once the agent evaluates that the task "#task" is being undertaken

- if(#belief)
  - True if the belief "#belief" is found in the belief base of the agent. This may be the agents actual belief base, or one created during the evaluation of the current process model.

4.1.6 Issues with NPE

Prolog-based NPE is not perfect. It has been designed to demonstrate how a language might be created to represent norms within the abstract model outlined in Chapter 3. A critical evaluation of Prolog-based NPE has been carried out and the following issues identified:

- Lack of namespaces/handling of data from multiple domains;
  - When designing a language for use over the Semantic Web, the ability to handle data from multiple locations is key. Namespaces enable data from multiple domains to be uniquely identified. Elements from separate domains can have the same local name but will be differentiated globally by a combination of namespace and local name (more on this in Section 4.2.2.1). The handling of namespaces provided by Prolog is poor. A form of namespace handling can be added with the aid of modules (or even an extension to the language itself). These solutions are at best temporary and the lack of well-defined namespace handling is proving to be an obstacle to Prolog's success both for use in this work and in the context of the wider Semantic Web.

- Poor exception handling;
  - ISO Prolog provides a built-in exception handling mechanism through the catch/3 and throw/1 control constructs. However, the technique applied is somewhat limited and certainly does not give the control expected by any developer used to modern object-oriented languages. In the scope of the current research, exceptions are seen as calls thrown by the interpreter when it detects that a norm has been violated. The exception call is defined within the triggering norm. The exception handling involved in Prolog is somewhat over simplified. When an exception is called, the Prolog interpreter walks back up the stack to find a catch where the exception is processed. Viewing

these types of exceptions in this way is unsatisfactory for the needs of this work. The construct call/1 could be used. However, this would not be as suitable as a dedicated exception handling process.

In Prolog, it is also trickier to add the necessary constructs to out model of exceptions in order to enable the reasoning over the cost of any violations. One of the key points to the NPE system described within this thesis is that actors should be able to use norms to add cost-based weights to given process models to aid reasoning with regards to normative facts present in any group. Prolog does not make this impossible, but it does make it harder than a more expressive approach might.

- Poor Semantic Web Integration;
  - As has been discussed above, Prolog has (relatively) poor namespace and domain handling capabilities. This is just one of a number of issues which make Prolog representations of data less suitable for use on the semantic web. It should be noted that Prolog is an extremely suitable platform for developing Semantic Web applications and has been used successfully for underlying application implementation in previous projects (Wielemaker et al. (2007)). However, it is not best suited for the exchange of information across the Semantic Web.
- Only One Action Allowed Per Rule;
  - One major limitation of this approach in that in its current form Prolog-based NPE would only be able to handle one action per normative fact. This is appropriate if a specification is reasoning over relatively uncomplicated situations, however, many cases may contain whole flows which may be forbidden, or prescribed. Prolog does allow the use of lists which could enable actions to be put in sequence. This covers a few basic situations, but the majority of more complex cases are still left without a valid representation.
- No Rule Nesting;
  - As well as limiting developers to one action per normative fact, Prolog-based NPE also limits how developers can nest rules. It is envisaged that in future rules could be nested within each other, triggered by exceptions and chained to form more complex structures. It would be possible to do some basic chaining using Prolog-based NPE, however, any more complex structures would require significantly more work (both in language design and interpreter implementation).
- Lack of Domain Specific and Profile Based Information;
  - In an item above it was noted that Prolog lacks the descriptive nature required useful integration onto the semantic web. Leading on from this point it is

stated that a key factor in the acceptance of a specific norm is often the weight gained by being assigned by a specific authority. In Prolog, it is likely that this weight would have to originate from the URI or other locator used to first obtain the normative fact. This is not a bad approach, but it lacks the flexibility that would be gained with the introduction of a fuller profile model (capable of linking an authoritative actor with a group of norms).

4.2 An Ontology for Norm-Based Process Expectations

Prolog-based NPE is a relatively straightforward language. It has been designed to represent elements in the abstract model outlined previously in Chapter 3. However, with the uptake of technologies supporting the semantic web gaining momentum, a Prolog-based language lacks many of the features needed to take NPE to a wider audience.

Building on the ideas put forward by Prolog-based NPE and aiming to produce an accurate representation of the norms required to implement the abstract model proposed in Chapter 3, the NPE language (NPE-L) is an ontology for norm representation serialised in RDF+XML. The use of RDF and XML enables an easy transition to any Semantic Web platform (both are ubiquitous Semantic Web technologies (see Section 2.2)). As with the model for norms described in Chapter 3 this ontology will have to enable the representation of norms acting over any reasonably expectable actions be those atomic or composite. Designing an ontology for norms and then serialising to XML is preferable to serialising Prolog-based NPE to XML as it enables storage and reasoning solutions to be reused from existing sources, as well as ensuring that an efficient representation of norms is achieved. NPE-L aims to solve many of the issues highlighted above including improving reuse, namespace handling and the representation of actions. Details of these improvements can be found in the following sections.

The ontology proposed in this section has been designed giving consideration to the solution requirements as detailed in Section 1.4.2, so as to ensure that the finished ontology aids in answering the research question stated in Section 1.4. In particular the ontology produced here should fulfil (either fully or partialy as a portion of the final solution) the following requirements;

- Provide a machine readable mechanism for the representation of norms to be used in reasoning over process expectations.
- Have ties to existing Semantic Web technologies so as to fit within the existing Web service stack (Section 2.1).
- Provide the ability to reason over both complex and simple sets of service based process model inputs.

- Consume over both complex and simple sets of normative inputs.
- Operate so as to enable the collection of data to answer the governing research questions posed in Section 1.4.

4.2.1 An Ontology for Norm Representation

This work has chosen to structure the ontology for norms in such a way as to enable multiple norms to be collected together into one specification. The structure of the OWL-S ontology for services was also taken into consideration during the design phase. The motivations behind these design decisions were based around the need to provide a coherent structure for storing and transporting multiple (possibly) connected norms, a wish to stick as closely to the model outlined in Chapter 3 as possible and an obligation to provide users with the four pieces of information about each norm:

- An overview the norm;
- The activities governed by the norm;
- Exceptions triggered when a norm is violated;
- The norm itself.

4.2.1.1 The Catalogue

The Catalogue class is the basis for any store of norms. Its job is purely organisational. On its own a catalogue has no functional use. Catalogues are intended to be used to group similar norms (norms acting on similar activities, roles, or simply originating from the same source) to aid processing and transport. Figure 4.15 outlines the basic structure of a catalogue. Essentially a catalogue groups four sets of data together: the Profile, Activities, Norms and Exceptions. The Profile presents the catalogue and outlines the norms held within. A set of activities governed by the enclosed norms is included for matching to processes or activities in a description. Any exceptions which might be triggered by the violation of an enclosed norm are also administered. The purposes of and motivations for, each of these sets is presented in the following selections.

The CatalogProfile is analogous to the OWL-S ServiceProfile. Essentially it describes the creator/owner of the catalogue and the abilities (norms) offered within. Profiles are associated with catalogues on a one to one basis. Every catalogue must have exactly one profile for the inclusion of identification data and to aid rapid discovery. The current version of NPE-L includes three main items of data in the profile to aid catalogue discovery: the catalogue name, the roles covered by any enclosed norms and a list of the norms themselves (in the form of identifiers). The key elements of the CatalogProfile

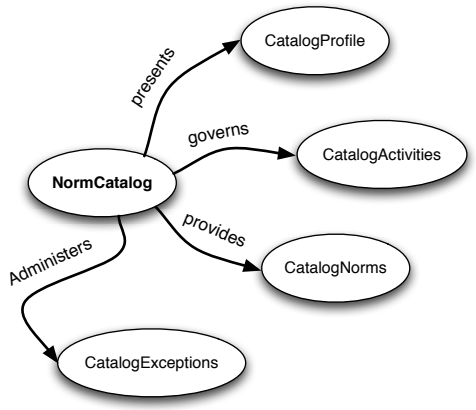

Figure 4.15: Top Level Norm Catalogue Ontology

class are shown in Figure 4.16. Activities have currently been left out of the profile. This decision was motivated by a wish to make the profile as lightweight as possible and the matching issues outlined in Section 5.1.1.1.

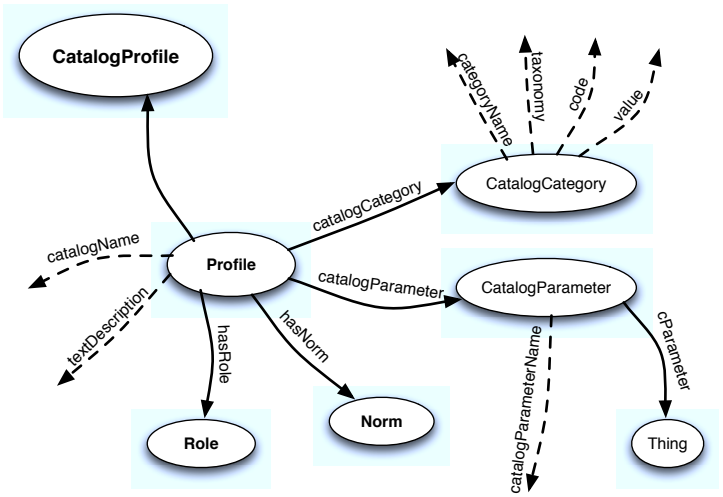

Figure 4.16: Selected Classes From The Catalogue Profile Ontology

Authorities One of the key instruments provided by the catalogue is the description of an originating creator. This is key as it enables a degree of authority to be assigned to the norms held within. As has been described in Section 2.4.12, authority gives weight to norms, especially when they are first appropriated by an actor or emerge in a group. NPE provides a way to assert an authority for a group of norms.

4.2.1.2 Activities

Activities (Figure 4.17) represent any task or process which is to be invoked in a norm residing within the current catalogue. Activities may be actual activities governed by a norm or part of an in-condition or deactivation condition. Activities map to OWL-S processes. The selection of OWL-S as an underlying language for the modelling activities has been driven by the wish to re-use an existing well-formed solution and to have a language which can be parsed in a similar fashion to any input language over which it may match. In addition to this, this work has been influenced by Barakat et al. (2012b) whose use of OWL-S in reactive service selection suits the needs of NPE. Matching algorithms (detailed in Section 5.1.1.1) match activities to OWL-S processes primarily through the use of inputs, outputs and names. For a greater depth of matching, activities also encapsulate a reduced version of OWL-S processes. These processes lack details of inputs or outputs (as they reside as part of the encapsulating activity) and only feature basic logic.

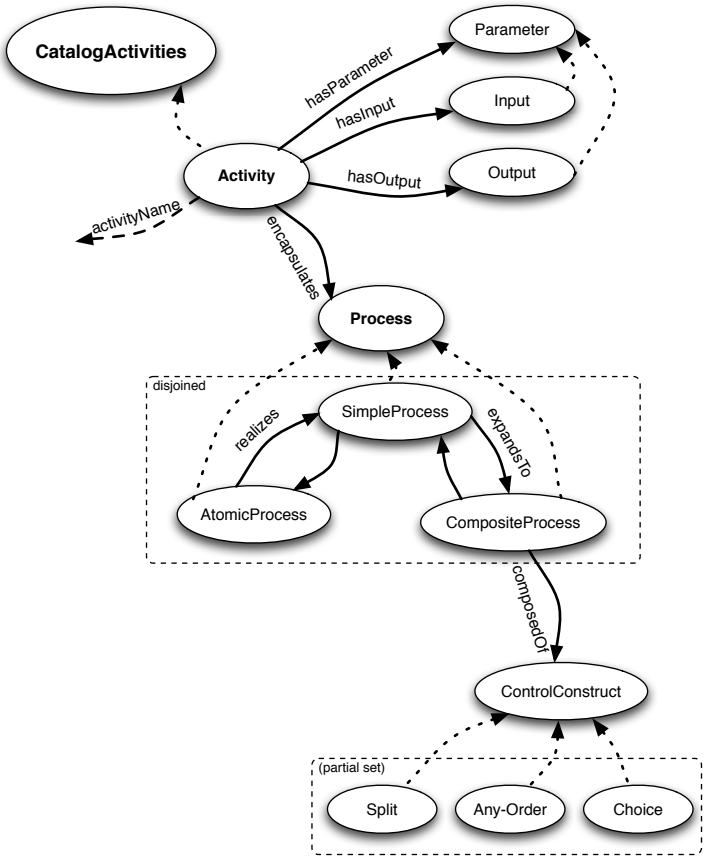

Figure 4.17: Selected Classes From The Catalogue Activity Ontology

4.2.1.3 Norms

Every other element in the NPE-L ontology acts to support the representation of norms. Norms are described using the CatalogNorm ontology outlined in Figure 4.18. NPE-L adheres to the definitions of a norm set out in Section 3.1.1. Despite the apparent additional complexity involved in NPE-L definitions, norms are still represented as a tuple as described in Definition 5. Norms in NPE-L link an activity element with a role, restriction, exception and two conditions. Each class represents a different restriction (forb, obli and perm). Conditions within the in-conditions and deactivation conditions of a norm are handled by sections of SWRL (see Section 2.2.5.4). By moving the definition of each part of the norm to a separate section, elements can be reused and whole catalogues of norms can be created more efficiently. One of the major changes between NPE-L and previous representation of norms used in this work is the inclusion of a norm identifier. This (along with URI namespaces) enables developers to globally identify each individual norm.

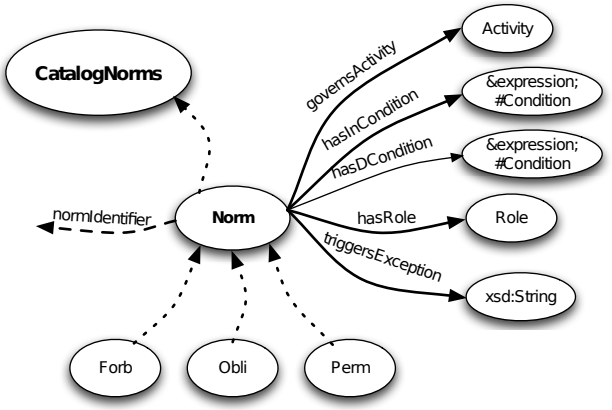

Figure 4.18: Selected Classes From The Norm Ontology

4.2.1.4 Exceptions

One of the biggest advances of NPE-L over Prolog-based NPE has come in the form of improved exception representation. Exceptions provide an indication as to the potential cost which could ensue from violating a norm. This cost is given in the current version as an integer. It is the intention of this thesis that, as an agent processes a set of norms, a score would be kept which would increase as more norms were violated. Norms with greater significance (to security, the correct running of a task, or any final outcome) can be associated with exceptions of a higher penalty cost. It is also important to note that this system also enables agents to be rewarded for breaking norms through negative penalties. Violating a norm and triggering an exception may also be unrecoverable (the evaluation process will have to stop and the process definition deemed unsuitable). This

situation might not be appropriate in many cases, so exceptions also have the choice to be recoverable. Thus an exception may add the penalty to a score and then carry on with the evaluation as if nothing happened. During processing it is this score which eventually becomes the cost of consuming the proposed service. Actors can then use the score to compare process descriptions (provided by services) with regards to the potential norm violations which might be required and select a final path accordingly.

It is important to note that exceptions and violations caught by the NPE system occur before runtime in a simulated environment used for reasoning over provided process descriptions. In such a situation, it is possible to be flexible with how an actor handled violations. At this point in the service selection, process actors can choose to ignore violations or adhere to held normative beliefs. In NPE violations only cause the perceived cost of consuming a service to rise (or processing to stop and the service to be noted as unsuitable). How an underlying actor uses this information is outside of the scope of this research.

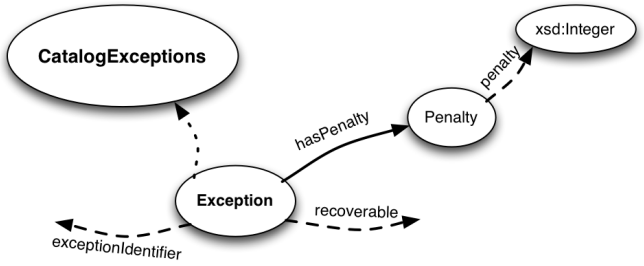

Figure 4.19: Selected Classes From The Exception Ontology

4.2.2 Additional Advantages Over Prolog-Based NPE

Prolog-based NPE has been developed to demonstrate that the abstract model outlined in Chapter 3 can be implemented in the form of a concrete language for norm exchange. Prolog-based NPE also aims to demonstrate how norms may be represented as facts in an underlying memory based model. However, as Section 4.1.6 has shown, there are problems with Prolog-based NPE. NPE-L improves on Prolog-based NPE in numerous ways. A number of these improvements have been shown above; the rest are detailed in the following subsections.

4.2.2.1 Namespaces and Identifiers

NPE-L is built on a set of technologies designed to enable the Semantic Web. Data on the Semantic Web is meant to be moved from machine to machine. For norms to be shared and exchanged on a large scale over a global network such as the Internet,

there must be a way to uniquely identify each individual norm and associated element. For technologies built on Semantic Web notions (such as XML and RDF), this unique identification is enabled through the use of namespaces and identifiers.

Namespaces are essentially conceptual containers which can be used to group together multiple elements. Namespaces should be globally unique and associated in some way with the creator/owner (a discussion about the true nature of namespaces falls outside of the scope of this thesis. For current purposes, it should be regarded that namespaces are globally unique and that they stem from the domain of the implementing author). Elements within a namespace are identified by a unique identifier. Identifiers are unique within the holding namespace. An element's global identifier is a combination of the namespace and its local identifier. As the namespace is globally unique and the identifier is locally unique, so the global identifier is globally unique, meaning that any element can be addressed using a single address. This allows developers to link elements (for example activities to norms) both locally and globally enabling improved reuse and addressing of existing elements.

4.2.2.2 Nesting

Nesting is a useful feature for single or small collections of norms enabling the whole collection to be represented in one document. In a standard NPE-L representation activities, roles, exceptions and even conditions are defined separately from the implementing norm. Elements external to the norm are imported through the use of identifiers. This is good for large catalogues, but for single norms or smaller (simpler) collections this degree of normalisation may be unnecessary. In these cases, it may be suitable to have these elements declared within the norm its self. Nesting allows this to be achieved. Nesting may also allow a future version of NPE-L to nest several norms together to form chains of norms, triggered either simultaneously, or in sequence following an exception being triggered (this feature is not currently implemented).

4.2.2.3 Storage

In designing the above ontology for norms, attention has been given to ensure that the finished artefact will be able to utilise one of the numerous technologies for storage and reasoning of data on the semantic web. A detailed analysis of a selection of available storage technologies can be found in Section 2.2.4.3 and Section 2.4.8.

The transportation of NPE-L norms between actors is likely to take place with the catalogue being encoded as RDF+XML. This form is accepted as a standard form and thus has the widest support among semantic Web frameworks and offers the best mix of human and machine readability. XML is also the technology usually used to encode

OWL-S documents for exchange and transfer and thus keeps the ties NPE-L has with OWL-S strong.

4.2.2.4 Reasoning

As has been noted above, NPE-L has been designed with storage and reasoning in mind. A detailed investigation into the reasoning techniques available for use with ontologies on the Semantic Web can be found in Section 2.2.4.

SWRL Within any norm there are two conditions which need to be described: in-conditions and deactivation conditions (InCondition and DCondition respectively). In designing how these conditions are to be articulated in NPE-L, the basic principles have been taken from expression handling in OWL-S. Conditions are defined as XML literals, which contain Expressions. The class Expression maps a condition expression to any one of a number of rule or knowledge languages (primarily SWRL, KIF or PDDL). NPE-L has been designed to operate with SWRL as the primary language for conditions descriptions. This decision was made as SWRL offers the best mix of features for the description of rules and also because of its base in OWL (DL and Lite) and its use in OWL-S. An introduction to SWRL can be found in Section 2.2.5.4.

Keywords NPE-L keywords are derived from Prolog-based NPE keywords. The two systems share the same common goal: to enable reasoning over transitions and events in an underlying world model. NPE-L encodes keywords as predicates in much the same way as its Prolog-based sibling does. Whereas NPE-L keywords are encoded in a chosen rule language, independent of NPE. For the current version of NPE-L it is suggested that the rule language SWRL is used.

The term "keyword" is actually misleading, as the keyword part is technically the property of the predicate. By keyword what is meant is key-predicate(name). Predicates which utilise keywords can have one or more variables. In NPE-L a predicate with one variable dictates that the predicate will be reasoning over the actions of the current actor or any global events. Effectively the scope of a single variable predicate is limited to all events which would usually only be visible to the current actor. In the majority of cases, it is foreseen that this will be used to limit the scope of the keyword to reasoning over a single role, communication channel, or state. Three basic Prolog-based NPE keywords have been extended in NPE-L. There are now four areas over which NPE can reason: Tasks, Data, Roles and Norms. Table 4.1 outlines the list of currently valid keywords. It should be noted that the implementation of specific keywords is left up to the NPE-Engine designer, so even though NPE-L supports a keyword, the NPE module may not.

Keyword	1-arity	2-arity	Description
In	in(region)	in(user,region)	True if the actor believes they are currently "in" the given region (task or process).
Once	once(task)	once(user,task)	Resolves to true once the given task has been performed once more (from the time the norm is activated).
Completed	completed(task)	completed(user, task)	Resolves if the task has been completed at any point in time so far.
Exists	exists(task)	exists(user,task)	True if the given task exists in the current description.
Send	send(msg)	send(user,msg)	True once the message "msg" has been sent.
Rcvd	rcvd(msg)	rcvd(user, msg)	True if the message "msg" has been received.
Join	join(role)	join(user, role)	When a user with the role "role" enters the interaction evaluates true.
Left	left(role)	left(user, role)	Same as join, but for a user exiting the interaction in a given role.
Active	active(norm)	active(role, norm)	True when the given norm becomes active.
Deactivate	deactivate(norm)	deactivate(role, norm)	True when the norm specified is deactivated (valid even if the norm is not currently active).
Violate	violate(norm)	violate(role, norm)	True on norm violation. Could be a better what of handling norms and allows the violation of one norm to trigger others.

Table 4.1: NPE-L Keywords

4.2.2.5 Importing External Rule Sets

As has been noted above, NPE-L documents existing outside of any containing system (during transport) are likely to be encoded as RDF+XML. This technology was chosen for its wide acceptance and good balance of human and machine readability (as has been stated in Section 4.2.2.6).

4.2.2.6 Syntax

As information in an NPE ontology is stored as RDF triples, this work recommends that for transport specification RDF+XML syntax is used. The choice of XML has been discussed previously in this thesis and is motivated primarily by both its wide acceptance as a Semantic Web standard and by its ease of readability by both humans and computers. Other encodings could be used, however, their suitability and availability would depend on the triple-store being used for storage.

4.2.3 Example

A minimal example is proposed based on the basic example given in Section 4.1.4.1 (Figure 4.13). This example states a buyer's obligation to pay for goods once he/she has finished shopping. Selected fragments of the complete specification are presented from Section 4.2.3.1 onwards. The full Catalogue can be found in Appendix A.3.

When looking at NPE-L and comparing it to Prolog-based NPE the similarities between the two with regards to the norms themselves must be observed. Figure 4.24 presents the norm specifying the obligation to pay as outlined above. As both NPE-L and Prolog-based NPE are based on the same abstract model (Chapter 3), they obviously both contain the same elements. This trait means that the norms specified in NPE-L will map exactly to the abstract model defined previously and so will be suitable for use in any concrete model implementing the abstract.

The major differences between Prolog-based NPE and NPE-L exist in the supporting elements around the norm. With Prolog-based NPE there is virtually no support and most of the work is required to be done by the original programmer. With NPE-L supporting elements allow norms to be defined with increased accuracy, without complicating the norm itself. Most of these definitions will be separate from the norm but linked back to it with URIs and identifiers. In the case of the norm being used as an example (Figure 4.24), the activity "ExampleCatalog_Activity" (also known as #PutItemInCart) which can be seen in Figure 4.20 and an exception which can be seen in Figure 4.21 are both defined externaly from the main body of the norm.

One feature of NPE-L which cannot be articulated in a Prolog-based NPE document is the ability to have a catalogue of norms with a profile for advertising and locating. The catalogue has been introduced above. As this example only contains one norm, the Catalogue (Figure 4.22) and Profile (Figure 4.23) are minimal documents. A section of both is shown the aforementioned figures.

4.2.3.1 Activity

A partial NPE-L activity definition for the activity ExampleCatalog_Activity can be found in Figure 4.20. This activity is regulated by the norm ExampleCatalog_ObliPutInCart (Figure 4.24). It is possible for activities to be regulated by one, many, or nil norms. In this catalogue, there is only one norm and only one activity (a de-facto one-to-one relationship). This activity is analogous to an equivalent OWL-S process; forming a basic representation of the key components required to obtain a match. For brevity numerous I/O definitions have been omitted and can be found in the full listing (Appendix A.3.4).

```
<activity:Activity rdf:ID="ExampleCatalog_Activity">
        <activity:activityName>
                Item_Purchase
        </activity:activityName>
        <activity:hasInput rdf:ID="ItemCodeInput">
                <activity:Input>
                        <activity:parameterType rdf:datatype="&xsd;#anyURI">
                                #ItemCode
                        </activity:parameterType>
                </activity:Input>
        </activity:hasInput>
        <activity:hasOutput rdf:ID="FrangoOutput">
                <activity:Output>
                        <activity:parameterType rdf:datatype="&xsd;#anyURI">
                                #BuyOutputType
                        </activity:parameterType>
                </activity:Output>
        </activity:hasOutput>
</activity:Activity>
```

Figure 4.20: An Example Activity in NPE-L

4.2.3.2 Exception

In the current version of NPE-L, every norm must have a corresponding exception. The exception ExampleCatalog_NoItemsException (Figure 4.21) is the exception called on violation of the norm ExampleCatalog_ObliPutInCart. It should be noted that it is expected that exceptions will be recoverable unless otherwise stated (in the default implementation of the NPE framework). As an example of how the variable recoverable could be set, this exception explicitly states its recoverable nature. Exceptions also have penalties associated with their triggering. In this case, the penalty has the integer value 25. Handling this penalty is left up to each individual implementation.

4.2.3.3 Catalogue

Catalogues encompass all elements of an NPE-L specification. Both functional and non-functional components are bound together using this class, allowing easier transport and storage of multiple norms. In this example, the catalog (ExampleCatalog

```
<exception:Exception rdf:ID="ExampleCatalog_NoItemsException">
        <exception:Recoverable>
                true
        </exception:Recoverable>
        <exception:hasPenalty>
                <penalty:Penalty ref:datatype="&xsd;#Integer">
                        25
                </penalty:Penalty>
        </exception:hasPenalty>
</exception:Exception>
```

Figure 4.21: An Example Exception in NPE-L

(Figure 4.22)), has one of each of the following: a profile (ExampleProfile), an activity (ExampleCatalog_Activity), a norm (ExampleCatalog_ObliPutInCart) and an exception (ExampleCatalog_NoItemsException).

```
<catalog:Catalog rdf:ID="ExampleCatalog">
        <!-- Reference to the Profile -->
        <catalog:presents rdf:resource="&profile;#ExampleProfile"/>
        <!-- Reference to the Activities involved -->
        <catalog:governs rdf:resource="&activities;#ExampleCatalog_Activity"/>
        <!-- Reference to the Norms -->
        <catalog:provides rdf:resource="&norms;#ExampleCatalog_ObliPutInCart"/>
        <!-- Reference to the Exceptions -->
        <catalog:administers rdf:resource="&exceptions;#ExampleCatalog_NoItemsException"/>
</catalog:Catalog>
```

Figure 4.22: An Example NPE-L Catalogue

4.2.3.4 Profile

A profile enables catalogues and their contents to be advertised and identified in collections or registries. Figure 4.23 shows the profile for ExampleCatalog. It should be noted that in this example the profile contains a reference to the services NAICS[1] category for easier automated retrieval (when looking through a registry utilising the NAICS category system).

4.2.3.5 Norm

Figure 4.24 presents an example of a norm described in NPE-L. Norms link all of the functional components of NPE-L into (normative) facts. The norm in Figure 4.24 (with the ID: ExampleCatalog_ObliPutInCart) essentially represents an obligation to place an item in the cart whilst shopping. In order for this norm to be valid, the activity ExampleCatalog_Activity must be defined (Figure 4.20) along with the exception ExampleCatalog_NoItemsException (Figure 4.21) and a role (not shown). Alongside

[1] North American Industry Classification System (http://www.census.gov/eos/www/naics/)

```
<profile:Profile rdf:ID="ExampleProfile">
        <!-- reference to the service specification -->
        <catalog:presentedBy rdf:resource="&catalog;#ExampleCatalog"/>

        <profile:textDescription>
                        The catalog provides an example of how NPE-L might be deployed in a real
    world situation.
        </profile:textDescription>

        <profile:contactInformation>
                <actor:Actor rdf:ID="ExampleCatalog_Contacts">
                <actor:name>Example Owner</actor:name>
                <actor:title> Service Representative </actor:title>
                <actor:email>act@or.com</actor:email>
                </actor:Actor>
        </profile:contactInformation>

        <profile:serviceCategory>
                <role:BusinessCategory rdf:ID="NAICS-category">
                        <role:value>
                                E-tailers
                        </role:value>
                        <role:code>
                                454111
                        </role:code>
                </role:BusinessCategory >
        </profile:serviceCategory>

        <profile:hasRole rdf:resource="&roles;#ExampleServerRole"/>
        <profile:hasRole rdf:resource="&roles;#ExampleClientRole"/>

        <profile:hasNorm rdf:resource="&norms;#ExampleNorm"/>
</profile:Profile>
```

Figure 4.23: An Example NPE-L Profile

these elements, in-conditions and deactivation conditions have also been created in SWRL. In this example the norm would be activated when the condition "in(#Buy_Sequence)" is triggered. This condition equates to the actor believing that it is currently in the process "#Buy_Sequence". The SWRL version of the condition is slightly more complex; it includes an argument which specifies the role belonging to the actor (see Section 4.2.2.4)

4.3 Related Research

As has been noted previously, the underlying model proposed in Chapter 3 on which the language and ontology contained in this chapter are based draws on existing research in its synthesis. As such the model and connected concrete implementation share relations to many existing solutions. However, as has been identified before, none of these solutions fully help to fulfil the research questions outlined in Section 1.4. As such the proposed model, language and ontology having been produced.

Of the related research which is worth noting, two existing languages are worth considering. OWL-Polar (Sensoy et al., 2010) is the closest language to NPE-L in terms of the creation of a language for the representation of restrictions with regards to actors who are

```
    <norm:Obli rdf:ID="ExampleCatalog_ObliPutInCart">
            <norm:governsActivity rdf:resource="&activities;#ExampleCatalog_Activity" />
            <norm:hasRole rdf:resource="&roles;#Example_ClientRole" />
            <norm:hasInCondition>
                    <expr:SWRL-Condition>
                            <rdfs:label>
                                    in(#Buy_Sequence)
                            </rdfs:label>
                            <rdfs:comment>
                                    Test to see if at the time this norm is applied the agent
believes it will be in the #Buy_Sequence. This is equivelent to the expression: in_region(
user, #Buy_Sequence).
                            </rdfs:comment>
                            <expr:expressionLanguage rdf:resource="&expr;#SWRL" />
                            <expr:expressionBody rdf:parseType="Literal">
                                    <swrl:IndividualPropertyAtom>
                                            <swrl:propertyPredicate rdf:resource="#in_region"
/>
                                            <swrl:argument1 rdf:resource="&roles;#
Example_ClientRole" />
                                            <swrl:argument2 rdf:resource="&activities;#
ExampleCatalog_Activity_BuySequence" />
                                    </swrl:IndividualPropertyAtom>
                            </expr:expressionBody>
                    </expr:SWRL-Condition>
            </norm:hasInCondition>
            <norm:hasDCondition>
                    <expr:SWRL-Condition>
                            <rdfs:label>
                                    once(#PutItemInCart)
                            </rdfs:label>
                            <rdfs:comment>
                                    Test to see if at the time this norm is active that it
will be deactivated by the following condition. This is equivelent to the expression: once(
user, #PutInCart).
                            </rdfs:comment>
                            <expr:expressionLanguage rdf:resource="&expr;#SWRL" />
                            <expr:expressionBody rdf:parseType="Literal">
                                    <swrl:IndividualPropertyAtom>
```

Figure 4.24: An Example Norm in NPE-L

endowed with semantic reasoning. However, OWL-Polar lacks the representation of costs for violations, the flexibility of the model proposed by NPE and light-weight focussed approach required for this research. Barakat et al. (2012b) suggests a mechanism for the selection of services potentially described in OWL-S but does not suggest a language or model which fulfils any part of the original research questions as identified at the start of this thesis.

4.4 Conclusion

This chapter has proposed a concrete language for the representation of norms with regards to expected actions of services. It has been shown that a formal Prolog approach can be used for the modelling of internal reasoning. Building on this an ontology for normative facts with regards to reasoning over expected processes of services has been created. The XML-based representation of this ontology (NPE-L) will form the basis

for the representation of norms for the rest of this thesis. NPE-L has been designed to sit with the Semantic Web and Web service technology stacks in much the same way as policy and rule languages (albeit whilst allowing for greater flexibility in how actors treat derived facts). In producing NPE-L this thesis has drawn on research conducted by the OWL-S community in order to enable easier integration into existing service consumers and simplify the representation of expected processes.

This representation will form a solid base on which a concrete implementation of a normative selection system can be produced. It should also be noted that this ontology lays out the domain of norms for service behaviours, irrespective of service selection and so could easily be re-purposed for use in describing norms in a variety of other situations (see Chapter 11.5).

Taken in their entirety, the concepts proposed in this and in the preceding chapter now form the basis of a system for the representation of and reasoning over norms with regards to expected processes. Having fulfilled the first of the research sub-questions as described in Section 1.4 and identified formally in Chapter 2, the following chapters will propose a full system for the selection of services based on beliefs drawn from normative facts. The following chapters will use the NPE language and model to create a system which can be used to validate the hypothesis that norms can be used to help with the selection of services in semantically rich environments.

Chapter 5

Norm-Based Selection Module

Previous chapters have proposed an abstract model for the representation and handling of norms (Chapter 3) and a set of languages for the storage and transportation of normative facts (Chapter 4). This thesis lays out the case for a normative service selection process. The following is a concrete implementation of the proposed architecture, utilising the ontology designed in Chapter 4.2. When viewing this implementation is it important to remember that it is framed from the point of view of a simulation (or reasoning phase) occurring before any actual interactions occur. As such norms and violations only affect the implementing actors beliefs about a proposed partner service. Thus the norms utilised within do not directly affect behaviour, but influence it by providing a cost analysis of potential process (before selection).

This implementation has been designed as an independent module which may be utilised by any software component requiring the ability to rate proposed partners by the perceived normality of the tasks they propose. The reasons behind choosing a modular design paradigm will be discussed in Section 6.3. A basic flow diagram for this proposed norm module can be found in Figure 6.1. Although the initial motivations for this module stem from research leading up to the design of a potential complete system for normative agent-based service selection, this module is intended for use in any system which can provide the following inputs and handle the following outputs:

- Inputs
  - Semantic Service Descriptions (one or many)
  - Access to a source of norms described in NPE-L (triple-store or other ontology of norms)
- Outputs
  - A document containing a reference to a service description and a score

As such, clear and well-described interfaces will be produced enabling the rapid deployment of this module. This will aid both testing of the module and eventual adoption and implementation into existing platforms.

Service Description Languages There are numerous service, process and workflow description languages. A review of a number of these languages can be found in Section 2.1. It is the eventual goal of the authors to make the normative selection module proposed herein language independent. However, this feature falls outside the scope of this thesis. As such, the decision has been made to use OWL-S as the description language of choice. The mechanisms surrounding the choice of OWL-S have discussed in Section 4.2.1.2.

OWL-S has played a leading role in the formation of the NPE-L ontology and language. NPE-L activities are directly linked to OWL-S processes. This decision aims to make the production of a matching algorithm easier in a final system. As such, OWL-S and NPE-L are appropriate technologies to use in this norm-based selection system.

5.1 Design

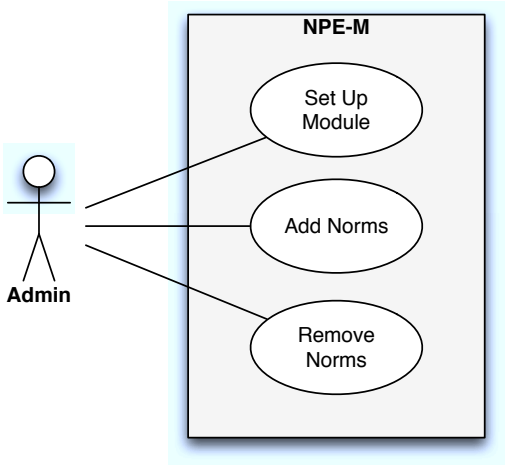

Figure 5.1: NPE Admin Use Case Diagram

To aid the identification of externally visible requirements for the NPE module (NPE-M) two use cases have been created (Figure 5.1 and Figure 5.2). These use cases are restricted in scope to covering the use of NPE-M during this thesis only. They are intended to capture all of the elements required to enable testing and deployment for simulations. Two actors have been identified: an administrator (responsible for initial set-up and deployment) and an agent system. It should be noted that the agent system has no access to the internals of NPE-M. The agent use case has been designed specifically so that the agent has a "black box" view of NPE-M, with well-defined access interfaces.

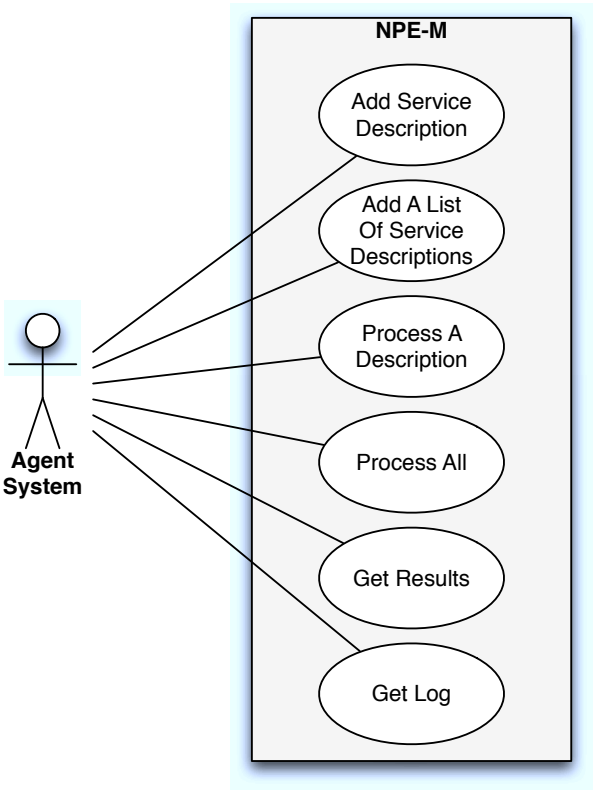

Figure 5.2: NPE Agent Use Case Diagram

In designing a normative service selection system (see Chapter 6), the decision was made to base the design on a set of modules. The module NPE-M described in this chapter is one of those modules. It was also desired that any modules within the proposed system could be taken and implemented as a standalone component in an agent-based system. As such, NPE-M has been designed to be both part of the proposed system and to be used as a standalone module for the grading of service descriptions. This design decision is reflected in the use case in Figure 5.2. NPE-M exposes enough functionality to enable the loading and processing of descriptions and norms, while keeping the majority of the processing hidden. The main aim of this is to allow NPE-M to be deployed into existing agent-based systems with the minimum of disruption.

The basic flow of an NPE runtime cycle has been designed to be relatively simple. The processing sequence can be seen outlined in Figure 5.3. One noticeable feature of this Figure is that it introduces the notion of a mechanism for the recording of results. After an NPE module has been initialised, the next step is to load a set (one or many) of OWL-S documents. These document sets are stored within the module and for each one a corresponding internal object is created. Results elements are created in memory with a reference to an OWL-S description, an empty model (in which to store a processed OWL-S description) and a score of zero.

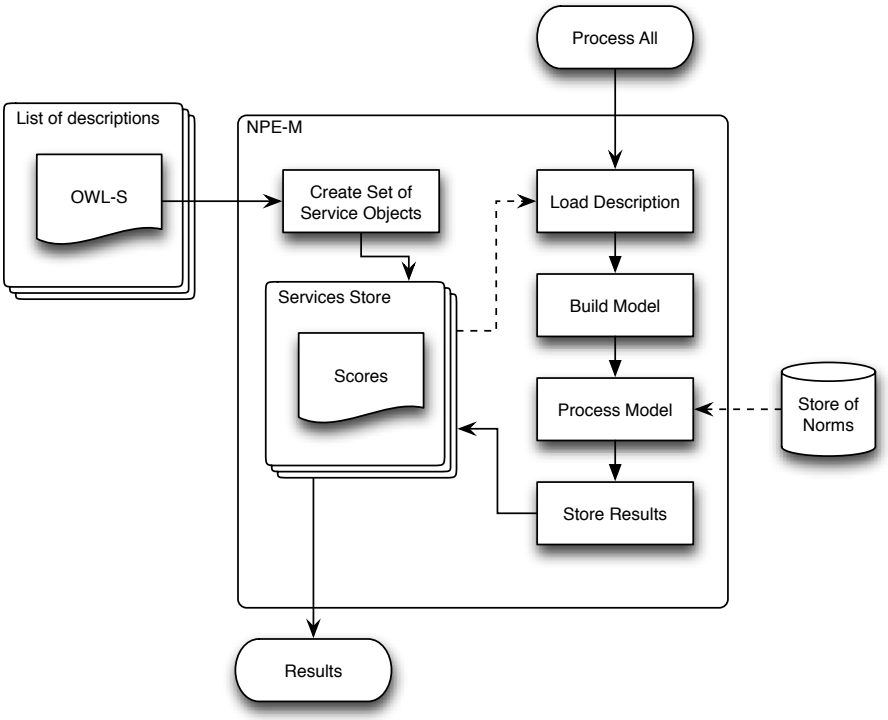

Figure 5.3: NPE Processing System

When the agent triggers processing the list of stored elements is traversed. Step by step a model is produced from the OWL-S process (Section 5.1.1); this model is traversed (simulated) by the NPE engine and at every step is checked against a set of norms (Section 5.1.5.1); finally, the score is stored back in the internal element. It is important to note that the results are stored but not yet returned to the agent. It is not until the agent requests the results that the final scores will be formally generated and returned. The decision to store the results and wait for the agent to ask before returning the final result set was made as a potential aid to concurrency. It is envisaged that an agent would start an NPE processing cycle in a separate thread and return once the processing had been completed.

The following sections outline in detail the processes behind the NPE-M workflow (Figure 5.3). The most complex sub-modules are those of model creation (Section 5.1.1) and of norm matching (Section 5.1.5.1). It should also be noted, at this point, that every section in this workflow has been designed as a separate module. This is especially important for the model generator, which currently only supports OWL-S processes, but in the future it is anticipated that it may also be able to handle processes descriptions (and workflows) in other languages.

5.1.1 Model Generation

Before any normative analysis can be carried out, the provided OWL-S description needs to be internalised. This step is required so that the system can interpret and reason over concepts held within the document. There are many varied ways which would allow the system to access the document. Straightforward parsing, was considered initially in this work, but subsequently discounted due to its inability to adequately handle complex processes. The approach chosen was to use situation calculus to represent the actions and control flows in use and then to convert this into an internal representation relying on a Petri net. This approach was influenced heavily by the following works: Narayanan and McIlraith (2003), Miao et al. (2008), Brogi et al. (2009) and Vidal et al. (2012). The following paragraphs will split this process into two steps (conversion to Situation Calculus and further processing to a Petri net based representation) and analyse each one with regards to its use in NPE-M.

Situation Calculus (McCarthy, 1969; Pirri and Reiter, 1998), is a popular modal logic language, enabling the representations of situations, actions and fluents (functions). In Situation Calculus, situations are achieved through the chaining together of actions. All possible worlds start in the situation S_0; the initial situation. Actors move between situations by performing actions. Fluents are extra and enable the representation of functions and relations when input values may change from one situation to the next. Situation Calculus is useful in modelling OWL-S process models since individual atomic processes in OWL-S map directly to actions in situation calculus. In Situation Calculus, input and outputs to and from actions are also represented in a method which is conducive to the representation of OWL-S; outputs are taken to be information which is being relayed to the client (agent) and any inputs must be known to the client before the associated action can be triggered. Much of this work has been influenced by Narayanan and McIlraith (2003). This research provides a fuller explanation of how OWL-S can be mapped using Situation Calculus. Not all of the research in Narayanan and McIlraith (2003) is used during the implementation of the NPE module. However, Situation Calculus has been a useful step in the design of a technique for the extraction and representation of OWL-S processes.

Having made the conversion from OWL-S to Situation Calculus, a suitably compact and yet expressive storage model is required. It was decided that a graph structure would provide all of the capabilities required by NPE-M. Most notable are the requirements that any structure must be able to represent control structures, activities and also be traversable (to simulate an execution of the OWL-S description). Again NPE has drawn on existing research (Narayanan and McIlraith, 2003; Miao et al., 2008; Brogi et al., 2009) and has chosen to represent the OWL-S process as a Petri net. A Petri net is a graph consisting of two different types of nodes; places and transitions. As with any graph, nodes are joined by edges (directional). Places hold tokens and predicates about

the surrounding world, whereas transitions are active components. Put simply, OWL-S processes map to NPE-L activities, which in turn can be converted to Situation Calculus actions, which finally map to transitions in a Petri net. Within a Petri net, access to transitions is limited through the use of places. A transition may not be triggered until the predicate within the place is true, or the place has the requisite number of "tokens" (where tokens are pieces of information required to preceded). This final limitation links to the requirement in Situation Calculus for all of the required inputs to be present for an action to occur. By mixing places and transitions, Petri nets can model all of the control constructs required by OWL-S.

In the implementation, several changes have been made to existing Petri net solutions. Most notably tokens are used to store the history of a specific execution path and hold their own state variables (including pools of active and inactive norms) and NPE scores. Additionally, Petri net conversions from OWL-S such as those in Narayanan and McIlraith (2003) and Miao et al. (2008) represent atomic processes within a wider process model as subnets (with individual processes being modelled as input and output transitions along either side of an external place). This research has chosen not to model to such granularity as it is not required in order to reason over the basic actions in a process. In the case of NPE, atomic processes are modelled as single transitions encapsulating all elements between the input and output transitions used in Narayanan and McIlraith (2003) and Miao et al. (2008).

The implementation proposed in this thesis builds on the above technologies to provide a structure that maps an OWL-S process model to an internal Petri net representation with links to NPE-L. OWL-S I/O is enabled through the use of the OWL-S API (Srinivasan, 2009). It is envisaged that by building on existing well-founded technologies, a solid grounding can be formed for NPE-M.

Processing Complex Process Descriptions OWL-S not only provides for the modelling of simple linear processes, complex conditional and looping paths are allowed. It is likely that such paths will have to be considered by the reasoning system. One problem with this is that these complex conditionals rely on termination or choice conditions detailed by a chosen rule language (SWRL for example). Deciding what conditions are likely to be met and which paths might be taken, is computationally difficult before consuming any service and may, in fact, be impossible (where conditions rely on the outputs from previous actions provided by an external actor). As such this research has decided to limit modelling of complex conditional and looping activities in OWL-S. Choice and If-Else conditionals will be handled as alternative paths; all of which will be explored by the NPE system. On reaching a conditional statement the NPE system copies any tokens currently in use forking their world states, so all paths can be analysed. For loops (such as Repeat-While and Repeat-Until) the processing engine will flattened the path into a simple If-Else path where one branch models undertaking out

an activity once and the second branch models undertaking an activity twice (and in the case of Repeat-While, a third branch models never undertaking an activity). This can then be processed in the same manner as a standard If-Else construct. Creation of any additional places and transitions needed to handle the processing of complex flows (such as loops) is undertaken during the initial creation of the Petri net based model. It is acknowledged that this approach might not be the most efficient (interleaving the processing of paths may be more efficient in the long run). However, this approach fulfils all of the requirements needed to test the NPE.

Another consideration is how to handle the merging of branches within a Petri net tree (such as those generated by Split-Join conditionals). At this point, NPE does not support Joins within a process model. The Petri net representation which has been used has the ability to easy model concurrent processing within a process model. However, joining paths in this system would require the tokens from each path to be merged into one single token which could be moved forward. This would entail merging multiple disjoint world states into one (including active norms, in active norms and execution histories). This research chooses not to support multiple disjoint world states to limit the scope of this thesis (it is assumed that adequate testing and validation of NPE can be conducted without this feature). However, it is noted that it is an area of great interest and worthy of investigation in future research.

The above constraints count for all complex process description constraints within OWL-S. It is worth noting that processing can refer back to the original OWL-S process model so that if a norm were to act to place a restriction on a composite activity involving a complex path this whole path (as defined in NPE-L as an activity) can be taken into consideration. This final consideration is not modelled during this version of NPE but is hypothesised as a potential need.

5.1.1.1 OWL-S Process and NPE-L Activity Matching

Activity matching is the mechanism by which OWL-S processes are matched to NPE-L activities. In the design of NPE-L, activities have been made analogous to OWL-S processes to aid matching (Section 4.2.1.2). For norms to be applied to an OWL-S description, the processes contained within it must initially be converted to NPE-L activities. Only OWL-S processes with associated activities can be regulated by NPE-L norms. Processes which have no equivalent activities cannot be monitored using NPE-L norms (or trigger changes in the NPE-M world state).

The following matching method is derived from the hybrid process matching mechanism presented in Sycara et al. (1999) (and Klusch et al. 2005). At the heart of this technique is a set of filters outlined in Figure 5.4. These filters have decreasing levels of accuracy and constraints, with the “exact” filter being the most accurate (and thus the least likely

to find a match in an uncertain environment) and with the hybrid approaches utilising subsumption and relaxing many of the constraints surrounding matching.

The majority of the matching surrounds the comparing of I/O parameters for processes and attempting to find a match based on these. A brief explanation of each of the filters can be found in Definition 15 to Definition 19:

Definition 15. OWL-S Process P exactly matches NPE-L activity A iff $EXACT = \forall\ IN_P\ \exists\ IN_A : IN_P = IN_A \bigwedge \forall\ OUT_A\ \exists\ OUT_P : OUT_A = OUT_P$. The process and activity are an exact match with respects to their formal semantics. All that remains to test is any extra data (name, catalogue, role, etc.).

Definition 16. A plug-in match relaxes the constraint: inputs for the Process P must be the same as for the matching Activity A ($IN_P = IN_A$). This match requires that the Process requires fewer inputs than the Activity, thus ensuring that the Activity can at all times represent the Process with a degree of accuracy. In this example $\geq$ is the symbol for subsumption. Thus, Process P plugs into the Activity A iff $PLUGIN = \forall\ IN_P\ \exists\ IN_A : IN_P \geq IN_A \bigwedge \forall\ OUT_A\ \exists\ OUT_P : OUT_A = OUT_P$. This filter observes the requirement that the outputs must match. Other versions alter this requirement with the limitation: $OUT_P \in LSC(OUT_A)$ stating that the outputs of P must be the same or very close to the concepts in OUT_A.

Definition 17. This filter moves on one step from the previous filter. Now not only is the input data for the activity subsumed by that of the process, but henceforth this restriction also applies to the output data as well, so that a match can be found iff $SUBSUMES = \forall\ IN_P\ \exists\ IN_A : IN_P \geq IN_A \bigwedge \forall\ OUT_A\ \exists\ OUT_P : OUT_A \geq OUT_P$. Essentially this means that the process provides more specific data than the activity and can thus be matched to it.

Definition 18. Subsumed-By and its successor (Nearest-Neighbour) are both hybrid matching methods. That is to say that they not only rely on concept matching, but also on a degree of similarity matching between the process and activity. Subsumed-By reverts back to a modified version of the model used by Plug-in matching where the inputs may be the same or more specific. However, this time the outputs may be direct semantic superiors of the concepts and a further modification made is to add an extra check in validating the two inputs against a similarity function ($SIM_{IR}(P, A)$). Thus a match can be found iff $SUBSUMED - BY = \forall\ IN_P\ \exists\ IN_A : IN_P \geq IN_A \bigwedge \forall\ OUT_A\ \exists\ OUT_P : (OUT_A = OUT_P \bigvee OUT_P \exists LGC(OUT_A)) \bigwedge SIM_{IR}(P, A) \geq \alpha$.

Definition 19. Nearest-Neighbour provides one more relaxation on top of the previous Subsumed-By filter. Now both the inputs and outputs may be the same or more specific, but more importantly the similarity function is joined with $\bigvee$. Meaning that the semantic matching can be disregarded for a simple IR filter. Thus, a Nearest-Neighbour match can be found iff $NEAREST - NEIGHBOUR = \forall\ IN_P\ \exists\ IN_A : IN_P \geq IN_A \bigwedge \forall\ OUT_A\ \exists\ OUT_P : OUT_A \geq OUT_P \bigwedge SIM_{IR}(P, A) \geq \alpha$.

Definition 20. A fail suggests that there are no NPE-L activities which match any of the provided OWL-S process using the rules in Definition 15 to Definition 19.

These techniques are augmented by a simple matching algorithm based on matching the IDs (names) associated with each process. This is a fairly simple test based on the SQL command LIKE which looks at the similarity of the name of the process and the name of the activity and determines if the two might be associated. This test is used to weight the results of the filtered tests above. This additional filter is not fully implemented in this initial version of NPE-M, as it is not required to test NPM.

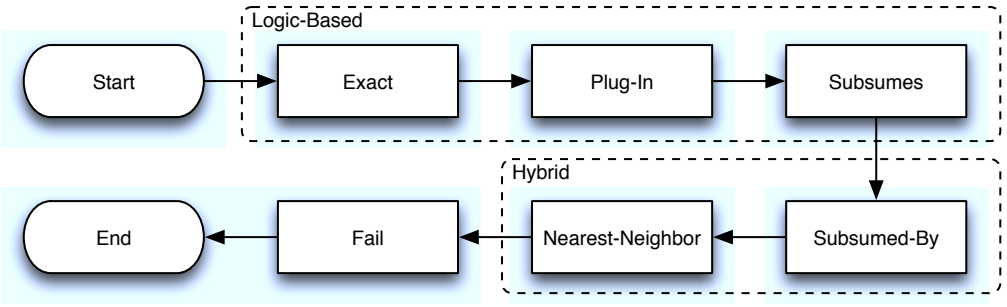

Figure 5.4: Activity Matching Flow

5.1.2 Loading Descriptions

Section 5.1.1 explains that before processing, OWL-S models are first converted to Situation Calculus and then to Petri net representations. This two step approach simplifies the process and provides a for an intermediatory step where all OWL-S processes are represented as situations with associated inputs, preconditions, outputs and effects.

This transformation is achived by extracting all OWL-S atomic processes and mapping them with the form $a(y)$ (where input parameters of an atomic process are the parameters y of action a) to classes in memory. Preconditions are then extacted as necassary conditions attached to the action in the current situation (concurrent preconditions are joined $\bigwedge$). Effects are extracted as axioms attached to the origin action joined to all other different combinations of actions and conditions that would make the effect possible. With the description injested, the in-memory list of Situaion Calculus axioms is flattened to four sets; places, transitions, inputs and outputs. Transitions are mapped as the join of external inputs possible for a place and the current internal state (knowledge). Thus a Petri net can be constructed in memory for each description where tokens pass around the current world state having undertaken a number of transitions where at each transition the inputs and conditions for a given situation were satisfied and the

outputs and effects have been recorded. More information on this process can be found in Narayanan and McIlraith (2003); Miao et al. (2008); Brogi et al. (2009).

5.1.3 Main Control Loop

```
1  private void processDescriptions() {
2          while(descriptions.next()){
3                  try{
4                          descriptions.Current.score = processModel(descriptions.Current.model);
5                  }catch(UnRecoverable e){
6                          descriptions.Current.unsuitable = true;
7                  }
8          }
9  }
10
11 private int processModel(ProcessNet model) {
12         return model.processPlace();
13 }
```

Figure 5.5: Main Description Processing Control Loop in Pseudocode

The NPE-M main control loop initialises processing for each individual service description. The list of descriptions (in the form of NPE-R documents) are setup prior to the instantiation of the main control loop. processDescription() is essentially a simple loop through the list of descriptions. Any unrecoverable exceptions triggered by non-recoverable violations of norms are caught and recorded at this point. The majority of the processing code is handled within the processModel() method which is detailed in the sections below.

5.1.4 Model Processing

The majority of normative processing during simulation is handled by the norm processor within processActivity(Description). This function forms the heart of the NPE-Engine and is run for every step on the Petri-based model associated with the input description. The function features three distinct operations:

- Norm Checking (Deactivation);
- Norm Matching (Searching);
- Norm Activation.

These three steps are detailed below. In addition to these operations, there is an ancillary task relating to the storing and maintaining of an internal world state for the simulation; updating the current tokens state (Section 5.1.6). The order of execution for steps during a function is: Load, Deactivate, Search, Activate, Update. Figure 5.6 illustrates the basic

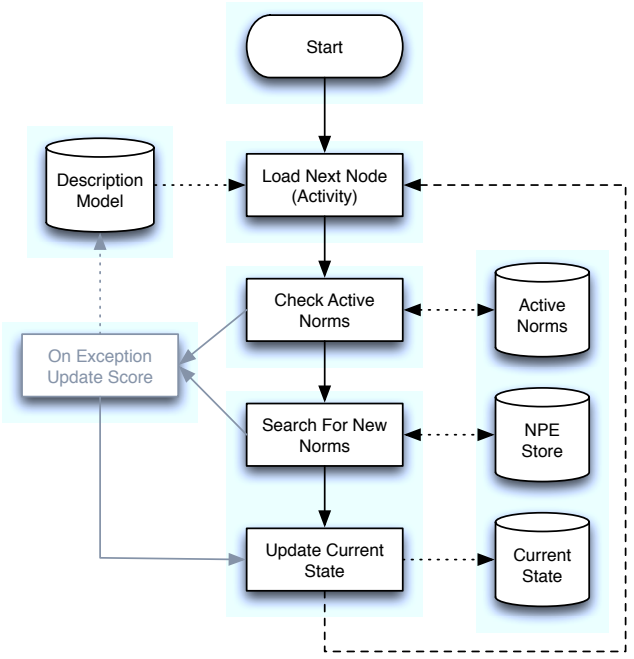

Figure 5.6: NPE Norm Control Loop Flow

flow of the norm processing function. A Java based pseudocode example of the structure of the whole processActivity() method can be found in Figure 5.10. It is worth noting that the activation and deactivation sections are both independent loops as they need to process newly found lists and active norms respectively.

The processing loop runs for every token passing every transition on the Petri net to the last without stopping. Processing will only stop if an unrecoverable violation is triggered and thus an exception is thrown.

To utilise the function processActivity() the NPE system must traverse the Petri net representation of a given process model description. Processing of the Petri net is handled using a depth-first approach leading place to place. Pseudocode of the function processPlace() can be found in Figure 5.11. This handles both traversing all reachable transitions in the tree and maintaining the correct number of tokens mapped to places (the marking).

5.1.5 Norm Handling

As has been discussed above, during model processing norms are handled in two steps: activation and deactivation. At all times, norms are described in NPE-L (Section 4.2) with the description described as in Section 5.1.1.

Norm deactivation (Figure 5.7) reasons over the validity of any norms the agent believes are active. This process looks at the deactivation condition of each currently active norm

and calculates if it is true or false. If the deactivation condition is currently true then the norm is deactivated (an outline of the condition matching processes can be found in Section 5.1.5.1). Norms usually deactivate silently, unless they are of typeOf() obligation and the task associated with the norm has not been undertaken (see below). In this case, a violation will be triggered. For all norms (not triggering an unrecoverable exception) deactivation is a simple case of being removed from the active norms store (and a note being made in the past norms store). The choice was made to have deactivation as the first step as it is important to remove any non-relevant norms from the active norms store before adding new norms (in case of inconsistencies) and to ensure that only the norms required are inspected during activation.

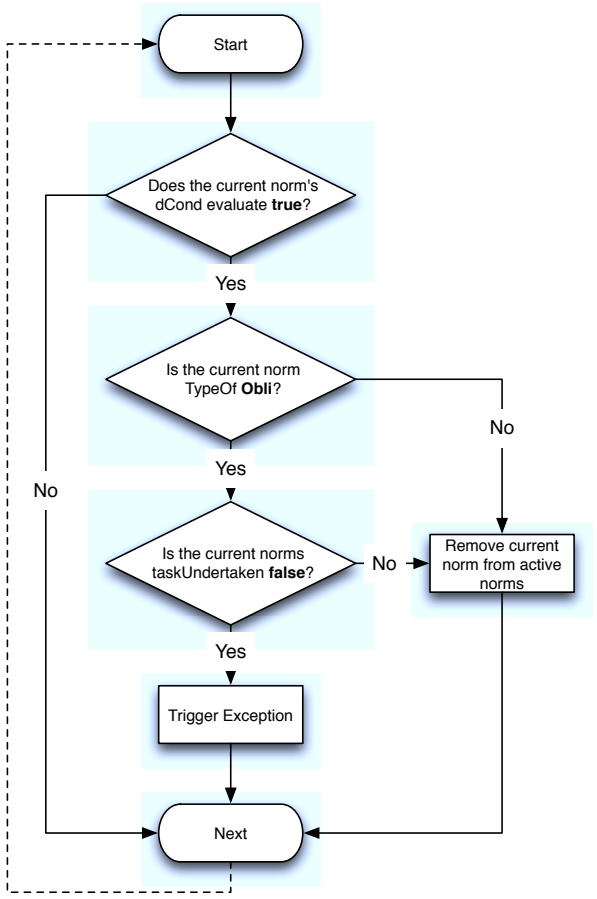

Figure 5.7: NPE Deactivation Flow

Norm activation (Figure 5.8) is somewhat of a misleading name. It could be argued that the actual activation of new norms is carried out once they have been matched, on their insertion into the active norms base. Activation is used as a label for this particular process as it is the first hurdle for new norms freshly inserted into the active norm base. The activate-norms loop then iterates over all currently active norms inspecting them and looking for violations. All the norms in the active norms base are taken to have met an in-condition at some point and have had no positive deactivation condition at any

point during the time they have been in the active norms store. The test carried out is to check if the task represented by the current step occurs in any of the active norms. Besides setting the value of "taskUnderTaken" to true for permitted and obligated norms, no major action is taken. If the matched norm is typeOf() forbidden then the norm is said to have been violated and a violation is triggered.

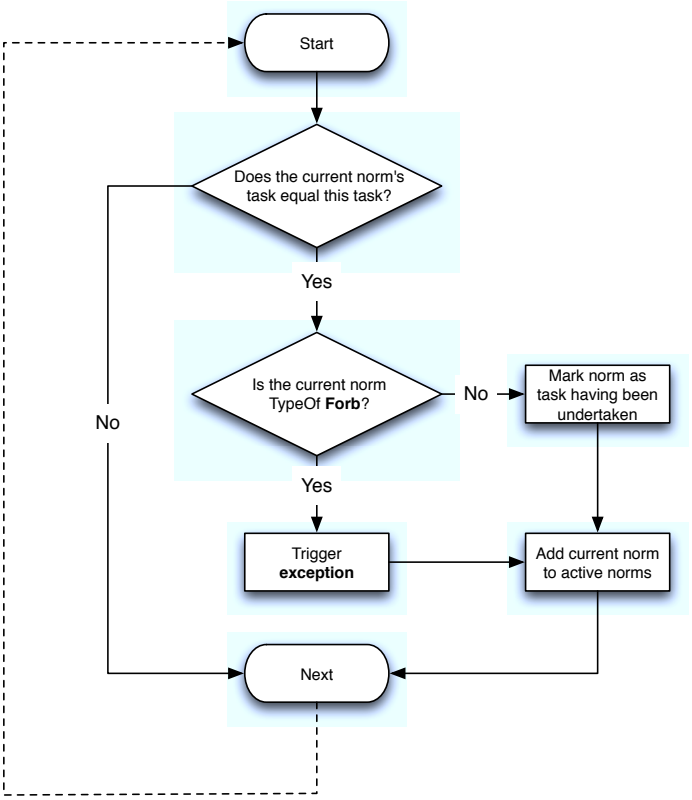

Figure 5.8: NPE Activation Flow

5.1.5.1 Norm Matching

Norm matching is simply a search over the agents' repository of norms (or potentially an external source/NPE-L document). This method is based primarily on the inspection of a norms' in-condition. If the in-condition resolves to true for the given current world state, then it is a candidate for activation. Once it is a candidate, the norms' deactivation condition is checked against the current world state. If the deactivation condition is also found to be true, the match is then cancelled (as the norm would be deactivated as soon as it was activated thus being cancelled out). If the in-condition is true and the deactivation condition is false, the norm has been matched and is added to the list $MATCHED$ ready for returning.

Condition processing has been outlined in Section 5.3. Currently, the majority of the reasoning is done over the current state using the key-words outlined below. The

conditions are expressed in SWRL (Section 4.2.2.4). However, the set of SWRL available for use is currently limited (more can be found on this in Section 5.4.1).

Supported Keywords Keywords are predicates encoded as SWRL with specific meaning to the NPE-M engine. These predicates are extracted during condition processing and passed to the engine for evaluation. Typically these predicates have a property name (extracted and treated as a keyword) and one or more variables. NPE keywords often refer to places or transitions occurring in the world around the simulated description. More can be found on keyword processing in Section 5.3 and a list of all the keywords supported by NPE-L can be found in Section 4.2.2.4. The current version of NPE-M does not support the full set of NPE-L keywords. Only the following may be used: In, Once, Active, Deactivate and Violate.

5.1.6 Storing The Current State

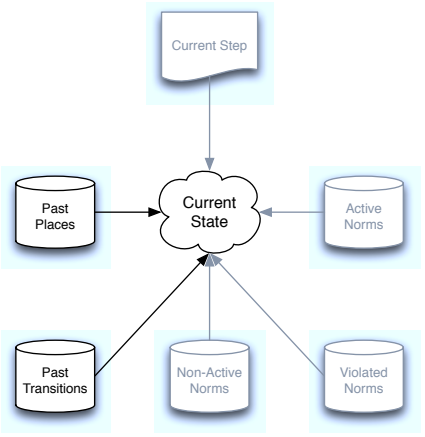

Figure 5.9: Update State

An NPE processing run is essentially a walk-through simulation of a Petri net based model of a set of OWL-S processes. For effective norm processing, the current state of the world simulated within NPE is required. This state is essential for the evaluation of activation (in) and deactivation (out) conditions for norms. For a norm to be activated the system must be able to reason over the value of any provided activation conditions. These conditions when included as part of a norm described in NPE-L can (and probably will) contain keyword conditions which work specifically with the current and past state records. More information on norm activation can be found in Section 5.1.5.

A number of different pieces of information are bound together to form the current state (and a history of past states). These have been summarised in Figure 5.9. It should be noted that the final step of each processing step is an update action on the current state. This action consists of storing the current step by putting the place (and any

associated transitions) in the appropriate store. As well as past places and transitions, the current state is also composed of a combination of the current active norms, all violated norms, previously (and now deactivated) norms and the current step. These elements are updated at various times during a single process (not necessarily during the update method).

For every initialisation place (state) on the generated Petri net, one token is produced. In the model, tokens hold the world state and execution history for a single execution path. Thus as each token is moved through the Petri net it essentially holds a unique isolated set of beliefs with relation to the world. Where complex conditionals or constructs are found, tokens may be copied or cloned (there is no difference between the two operations, however, they hold different meanings when joining paths). Copied tokens share the same worldview until they are changed through a transition. At this point, they become separate and changes to one occur in isolation.

5.2 Service Scoring

A service description's score serves as the most basic unit of output from the NPE module. From this score, an implementing system can decide to what degree they believe a process proposed by a service is "normal". Scoring is carried out after process and norm matching as part of the norm checking sequence. Here if a norm is violated an exception is triggered. Every exception contains a penalty: a numeric value of how serious a norm violation is. These penalties are set by the designer who initially defines the exception; allowing developers creating different catalogues of norms to set different penalties for very similar violations occurring in different contexts or at different levels of service. All services start with an initial score of zero. This means that an end score of zero signifies a service which does not violate any norms (or which successfully negates any violations, see next paragraph). A higher final penalty score signifies that a service definition either violated a number of norms or that any norms violated were understood to be more important. A high final penalty score would signify that a service is expecting participating actors to undertake tasks which violate expected norms.

For complex paths in process models, multiple final scores may be possible (one for each token present in the Petri net marking after processing). Only the highest score is returned following execution. This discussion can easily be evaluated in the future (to discover if alternative paths might be worth evaluating), for now though the "worst case scenario" is used to rate processes.

An extension has been added to the scoring system to allow the use of negative scores. Any score less than zero is interpreted as a reward, this allows the definition of potentially beneficial norms, which could be defined in negation thus enabling their adherence (or from the point of view of the system violation) to trigger a reward.

5.2.1 Exceptions

As has been outlined previously, exceptions result from the violation of an active norm. For a norm to be valid it must have an exception to trigger. There is no support in NPE for norms which when violated trigger no exception (although such norms could be used for logging). Exceptions are by default recoverable. That is to say when an exception is triggered the enclosed penalty is added to the total score for the current service description and then processing continues. Exceptions may also be non-recoverable. Triggering a non-recoverable exception causes the service description to be recorded as unsuitable for use and all processing to stop. This scenario should be left for only the most serious of violations.

5.3 Conditions, Current State and Reasoning

Section 4.2.2.4 outlines the representation of conditions in NPE-L. State is crucial to the processing of conditions in NPE. At all times, the NPE module maintains an accurate representation of the state of any simulated process (Section 5.1.6). Conditions are functions which evaluate to either true or false. In NPE-L conditions are used to activate and deactivate norms. A condition which evaluates to true can be referred to as having "triggered" the activation or deactivation of the associated norm. Thus, conditions form the basis for the entire reasoning process in NPE and must be treated accordingly.

The NPE-L ontology caters for the inclusion of conditions in any rule language. However, the current version of NPE-Engine only supports SWRL as a base language. In addition, SWRL conditions can then employ NPE keywords. The current version of the NPE-Engine only supports a small set of the available NPE-L keywords (to reduce the complexity of the NPE module). From the point of view of the NPE-Engine, the majority of the condition processing occurs during the keyword evaluation process.

To enable greater flexibility NPE extends SWRL using SWRL-FOL (Patel-Schneider, 2005) to add unary/binary first-order logic. The predicates AND and OR are currently supported. Processing for SWRL-FOL has been added to the NPE processing loop as an extension to enable reasoning over complex conditions and first-order logic expressions.

The SWRL processing loop is a relatively straight forward. The goal of this loop is to have all of the keyword state processing handled by the NPE-Engine and to have the SWRL processing handled by the SWRL processing engine. Thus, SWRL processing starts with the parsing of a string representation of the condition and the extraction of any keyword predicates. These predicates are then passed to the NPE-Engine where they can be evaluated. The NPE-Engine returns a true or false for each of these predicates. These results are then inserted back into the SWRL condition for final evaluation.

Keyword predicate evaluation within the NPE-Engine relies on the ability of NPE to store the current state and reason over this state. Keywords act to evaluate various parts of the current state (Table 4.1). Evaluation simply requires that NPE-E matches the keyword to an evaluation technique and then carries out the required evaluation with the predicates variable(s) providing the input (state name/process/input/output). The evaluation will normally be a search through the correct store to look for an element which matches the input.

5.4 Module Implementation

NPE-M has been designed to be implemented entirely in Java. The decision was taken at an early stage of the research development process to implement any proposed service selection architecture using a modular design paradigm. As such, the NPE module has been implemented as a single Java package exposing its capabilities through well-formed (and documented) interfaces. The NPE module has no ties to any one agent architecture or selection process and it provides the basic capabilities outlined by the use cases in Section 5.1.

A number of external Java libraries and tools are imported to provide advanced functionality and to cut down on the amount of coding required. Parsing and processing for OWL-S and NPE-L documents is handled by the OWLS-API and Protégé OWL-API respectively. These libraries enable the rapid inclusion of document I/O and reasoning without the need for a new parser. The Jena framework has been employed initially for the storage and reasoning of norms. Jena has been chosen as it is one of the most mature Java triple-stores. However, it is envisaged that concerns about the speed of Jena queries (McBride, 2002) will lead to it being replaced by Sesame (Broekstra et al., 2002).

5.4.1 Limitations

The current version of NPE-M aims to provide a concrete implementation of the NPE model and support a testable set of the capabilities of NPE-L. However, to ensure that this work is both realistic and attainable, some of the features of NPE-M have had to be limited. The following is a list of the current limitations:

- Service Definitions Limited to OWL-S;
  - One of the core design decisions surrounding this initial implementation of the NPE framework (model and languages) has been to only allow service descriptions in OWL-S. The decision as regards to the implementation of NPE-M was relatively straightforward as it has been dictated by the design of the NPE-L language. NPE-L activities match to OWL-S processes and

```
1  private int processActivity(Activity activity, Token token) throws Unrecoverable {
2          int penalty = 0;
3
4          if (!activity) {
5              // Currently we do not support reasoning over control constructs
6              return;
7          }
8
9          // Check all active norms
10         while (token.worldState.activeNorms.next()) {
11             Norm current = token.worldState.activeNorms.Current;
12             if (current.dCondition.Eval(activity, token.worldState.currentState)) {
13                 if (current.TypeOf(Obli)) {
14                     if (!current.taskUndertaken) {
15                         penalty += current.triggerException();
16                     } else {
17                         token.worldState.activeNorms.Remove(current);
18                     }
19                 } else {
20                     token.worldState.activeNorms.Remove(current);
21                 }
22             }
23             if (current.task.Equals(a)) {
24                 if (current.TypeOf(Forb)) {
25                     penalty += current.triggerException();
26                 } else {
27                     current.taskUndertaken = true;
28                 }
29             }
30         }
31
32         // Search for any new norms and add them
33         LinkedList<Norm>matchedNorms = this.normBase.Search(activity, token.worldState.actor,
       token.worldState.currentState);
34
35         token.worldState.activeNorms.AddAll(matchedNorms);
36
37         // Activate any eligible norms
38         while (token.worldState.activeNorms.next()) {
39             Norm current = token.worldState.activeNorms.Current;
40             if (current.task.Equals(a)) {
41                 if (current.TypeOf(Forb)) {
42                     penalty += current.triggerException();
43                 } else {
44                     current.taskUnderTaken = true;
45                 }
46             }
47         }
48         token.worldState.currentState.Update(activity);
49     }
50 }
```

Figure 5.10: Activity Processing Pseudocode

have been heavily based on OWL-S. Thus the decision to use OWL-S only descriptions was taken to make this implementation as simple as possible. This limitation only exists in the current version of NPE-M. It is envisaged that future implementations could include model generation facilities for alternative description languages. It may even be possible that other process (and worflow) description languages could be considered at a later date. Changes and updates should be possible due to the modular nature of NPE-L as any change would

only require a new model generation module (this would typically only require new matching filters).

- Limited Support for Complex Process Descriptions;
  - OWL-S processes support almost an unlimited array of flow contructs and behaviours. Modelling every permutation of every possible combination of control structures would be almost un-computable. To simplify model generation at this early stage of development, a number of limitations have been placed on the handling of OWL-S processes during model generation. Most notably: Repeat-While and Repeat-Until are limited by a lack of processing for termination/choice conditions and Split-Join's are unsupported. More details about these restrictions can be found in Section 5.1.1.
- Process Matching Limits;
  - It was envisaged during the inception of NPE-L that composite OWL-S processes could be matched to NPE-L activities using their constituent processes as a signature. This technique could then be augmented with the filtering approach currently used to produce an improved matching system. Unfortunately, this process is too complex to model within the scope of this thesis. A new "look forward" simulator capable of examining not only the current process but also any subordinate processes, would have to be employed. It was decided that this could be too complex for the current release, so a simpler filter only approach was applied.
- SWRL Only Conditions;
  - NPE-L supports the definition of activation and deactivation conditions for norms in a number of languages. However, the default language suggested in Chapter 4 is SWRL. Due to this and to make the creation of NPE-M simpler, the decision has been made to only allow conditions described in SWRL (extended with SWRL-FOL where needed). This is not a major problem as SWRL provides all of the features NPE-L requires in a rule language.
- Only A Subset of the NPE-L Keywords Supported;
  - The NPE-L specification defines numerous keywords which may be used during the construction of activation and deactivation conditions for norms (Section 4.2.2.4). These keywords aim to aid the interaction between potential norms and the current world state. To simplify the construction and testing of NPE-M, only a sub-set of these keywords have been implemented (Section 5.1.5.1). These keywords have been chosen so as to enable as complete testing of the NPE framework as is possible without bloating NPE-M.
- No Runtime Editing of Norms;

  - The use cases for NPE-M (specifically the administration use case Figure 5.1) specifies that norms will be added and removed during the set-up of the NPE module. No allowance has been made for the modification of any stores of norms by NPE-M during runtime. This limitation was put in place for two reasons: the first being that any store could be held separate from NPE-M and thus may be changed without accessing NPE-M, the second is that any changes to the norm store are likely to occur before or after any interactions and thus fall outside of the scope of this thesis.

- Limited Use of Roles;

  - The NPE-L specification allows for the defining of roles for participating actors. These roles can be used to aid catalog and norm selection and also during condition processing. Little work has been done to investigate potential methods for processing roles in an implementation of NPE and, therefore, roles have been mostly ignored in this version. It is envisaged that future releases will include more role guided processing options.

5.5 Conclusion

This chapter has detailed the design of a service selection module based on the NPE-L ontology and model proposed in Chapter 3 and Chapter 4.2. This module fulfils the second research subquestion in that it presents a mechanism for the reasoning of norms with regards to expected processes which enables norm based service selection to be performed. It is the opinion of this research that this technique will be an improvement over those existing methods identified in Chapter 2, in situations where norms are available and interaction with complex services is required.

This module builds on the research conducted in this thesis thus far and aims to provide a platform on which the concepts behind NPE can be tested. In particular, Chapter 6 will focus on building a system capable of utilising the NPE-M module outlined here. The system is to be designed and built as part of the testing of NPE and NPE-L and as such will encompass an NPE-M module with all of the tools and logic required for NPE to function in a production ready setting.

```
1  private void processPlace(Place place) throws Exception {
2         List<Token> tokens = marking.get(place);
3         if (place.endPlace) {
4             // We have finished
5             for (Token token : tokens) {
6                 try {
7                     // Test for outstanding obligations
8                     processObligations(token);
9                 } catch (Unrecoverable e) {
10                    if (settings.strict) {
11                        throw e;
12                    }
13                }
14            }
15            return;
16        }
17
18        List<Transition> transitions = getTransitions(place);
19        if (transitions.size() == 0) {
20            // Not currently used as relates to Join
21            return;
22        }
23
24        // depth first traversal
25        for (Transition transition : transitions) {
26            // Copy tokens for each transition
27            newTokens = tokens.copy();
28
29            // Join specific, not implemented
30            // if(transition.isJoin()){
31            //  newTokens = transition.mergeTokens(newTokens);
32            // }
33
34            // Process tokens
35            for (Token token : newTokens) {
36                try {
37                    processActivity(transition.activity, token);
38                } catch (Unrecoverable e) {
39                    if (settings.strict) {
40                        throw e;
41                    }
42                }
43            }
44
45            List<Place> reachablePlaces = getNextPlaces(transition);
46            for (Place reachablePlace : reachablePlaces) {
47                if (reachablePlaces.length > 1) {
48                    // Clone tokens (track this as parent)
49                    // Mark tokens to new place
50                    marking.get(reachablePlace).pushAll(newTokens.clone());
51                } else {
52                    // Copy tokens (keep parent)
53                    // Mark tokens to new place
54                    marking.get(reachablePlace).pushAll(newTokens);
55                }
56
57                processPlace(reachablePlace);
58            }
59
60            // If it is the last transition, remove tokens from this place
61            marking.get(place).clear();
62        }
63    }
```

Figure 5.11: Petri net Place Processing Pseudocode

Chapter 6

Evaluation Environment Set-up

This research has so far aimed to identify whether there is a research gap surrounding the use of norms in service selection. In particular, this thesis has focussed on the areas of normative agents and the evaluation of process models as a method to achieve greater accuracy in service selection. Previous chapters have identified that there is a research gap surrounding the use of norms for services selection, identified areas of research which are lacking for such a system to be produced and have proposed a solution in the form of an ontology for norms with regards to processes and a module for the reasoning over processes with regards to norms.

Beyond the creation of a framework for normative process evaluation, the focus of this thesis is on the investigation as to the limits of any such systems. In particular, attention is to be paid to the balance between adding value using a norm-based approach, versus occurring extra processing or resource costs. The following chapters form the core of this research effort, building on the work carried out previously and forming a fully functional test system on which to conduct experiments to answer the questions posed in Section 1.4.

To answer the questions posed in the hypothesis from which this thesis arises, a degree of experimentation is required. This will necessitate the production of a system with the ability to model an expected semantic framework into which a module containing an NPE based selection engine can be placed. The purpose of structure will be to facilitate the testing of the limits and bounds of the NPE system, in particular with regards to the governing research question:

> *"How can an actor with a set of known normative beliefs use these beliefs to aid service selection where IOPE matching typically falls short?"*

6.1 Methodologies

In this chapter the tests and evaluations to be applied to the architecture proposed in the previous chapters are detailed. A review will be conducted of the solution proposed using the following methodologies:

Baseline Measurement Testing to demonstrate that the proposed system can function to select services and that any selection activities are rational and expected.

Multi-Agent Simulation Simulated use within an agent environment. Utilising agents as an efficient mechanism through which the limits of normative service selection can be explored.

6.2 Motivation

Presently this research has only focussed on abstract, theoretical, or partial solutions for the need for normative based approaches to service selection. The primary motivations of this thesis as a whole however have been to identify whether service selection might be improved through the introduction of normative reasoning and to identify whether such an approach provides any measurable performance benefits over traditional (profile model) based approaches.

The purpose of the remainder of this thesis is to take the system produced in Chapter 5 and use it to answer the outstanding research questions posed in Section 1.4. The previous chapters have helped to set out a well-formed base of research and have succeeded in answering the first two questions posed in Section 1.4.

By using the methodologies previously proposed, this research can demonstrate that a system which can answer the remaining research questions has been produced. It is hypothesised that normative service selection can be used when traditional input/output approaches, which rely on semantic matching of profile models, fail to identify a single relevent service. The tests and simulations carried out will attempt to build an unbiased result set to demonstrate this hypothesis and answer the research questions posed in Section 1.4. In particular, this chapter aims to fulfil the following solution requirements as specified in Section 1.4.2:

- Operate so as to enable the collection of data to answer the governing research questions posed in Section 1.4.
- Identify appropriate methodologies to test any solution against the research questions posed in Section 1.4.

As well as answering some key questions, the previous chapters have naturally also led to the evolution of many new questions surrounding the use of norms in environments where service selection is being performed. This includes questions of how norms are to be exchanged, the creation of norms and complex reasoning over the effects of norm violations on the long-term utility of an actor's actions. These questions and any other topics not directly linked to the research questions outlined in Section 1.4 during the following investigations have been discounted, as they are deemed not necessary for testing of NPE (answering the research question or any of the sub-questions).

6.2.1 Evaluation Criteria

To reiterate, the governing research question of this thesis is:

> *"How can an actor with a set of known normative beliefs use these beliefs to aid service selection where IOPE matching typically falls short?"*

To answer this research question three sub-questions are outlined in Section 1.4. The final sub-question is as follows:

- What costs or bounds might a normative approach place on actors undertaking service selection?

To answer the final sub-question, a criteria for the evaluation of the costs or bounds placed on an actor undertaking a service selection will be required. Based on existing research such as Klusch (2012) and Küster and König-Ries (2010), and lessons learned from building NPE, it has been decided the the criteria for evaluation will be; a score based on the selected service description, and the time taken for selection to occur. These criteria will then be used to attempt to answer the final research sub-question.

6.3 System Design

Before the hypothesis laid out in Section 1.4 can be tested, the NPE-M module for normative process model grading must be introduced to a system where its outputs can be used to influence service selection.

When considering the overall design of any NPE based system, it has been decided that of utmost importance is the ability to add NPE to an existing or new system with minimal changes to any existing code base. Neither NPE nor the host system should have to be altered beyond the implantation of any interface requirements. Additionally,

communication between an NPE based system and any utilising host should be carried out through a single well-defined interface. As such the decision has been made to create a self-contained NPE based module, contained within a separate package and distributed as a standalone entity. This can then be implemented by any other system utilising the same language.

The NPE system as described in this chapter is essentially a testing harness for the NPE-M module as designed in Chapter 5. This module provides basic NPE processing capabilities with set inputs and outputs. The testing module will have to provide a setting in which the NPE module can receive the correct inputs and which can then use the generated output to determine how best to proceed (in the case of our testing , which service to select).

Discussions were carried out as to the best way to implement any NPE system for testing purposes and also how this module might differ from one deployed commercially. Numerous options have been considered during this research. Among the most notable was a plan to create a standalone NPE server based on Node.js (Dahl, 2011). This would have exposed the capabilities of NPE through a single SOAP interface. Such an implementation could be a good solution to the question: "how can we provide an easy to implement module which can be updated at will by the original developers and requires minimal set-up by implementers". However, the cost of transferring every query would have outweighed any potential benefit. Because of this deficit, this idea has been deemed as being out of scope of this thesis. Further research into this field can be found in Section 11.5.

The design principles behind the NPE testing module, used during this thesis are formed around the notion of a single "black box" module. The construction of which; is focused on enabling the widest number of uses without requiring any changes on the part of implementing actors. This modularity will be reflected in the design of the module itself. Where possible the implementation should reuse existing software components. This reuse is important to both the short-term creation of an NPE test module and also to any long-term plans for further deployment. Using existing components should cut the time required to created any module, will provide a more stable starting point and should enable the collection of results which more closely match those which might be expected in a production grade system. The tests conducted as part of this thesis are required to discover whether the ideas behind NPE might be suitable for deployment on a semantically aware platform. The suitability of any one implementation or set of tools used to deploy NPE on a potentially commercial scale is outside of the scope of this research. Thus, the prototyping and testing described in this and the following chapter will rely on a module built with stability and speed of development in mind. Less attention will be paid to future deployment or customisation, so long as all tests share any apparent costs. A general overview of the NPE module can be found in Figure 6.1.

The test module should follow the ethos of this whole thesis and maintain the use of open and freely available standards. Where possible to ensure that the finished module will produce test results which are an accurate depiction of use in currently deployed systems, common standards and techniques should be practised. These will include many of the standards described in Chapter 2 (more research towards the reuse of existing open standard derived toolkits can be found in Section 11.5).

The rest of this chapter will analyse the research and design process carried out to produce the software module used to test NPE. As this process focuses on modular design and reuse, this research will take the form of a review of whether the existing modules available are suitable for use, followed by the designing of new modules to fulfil NPE specific tasks.

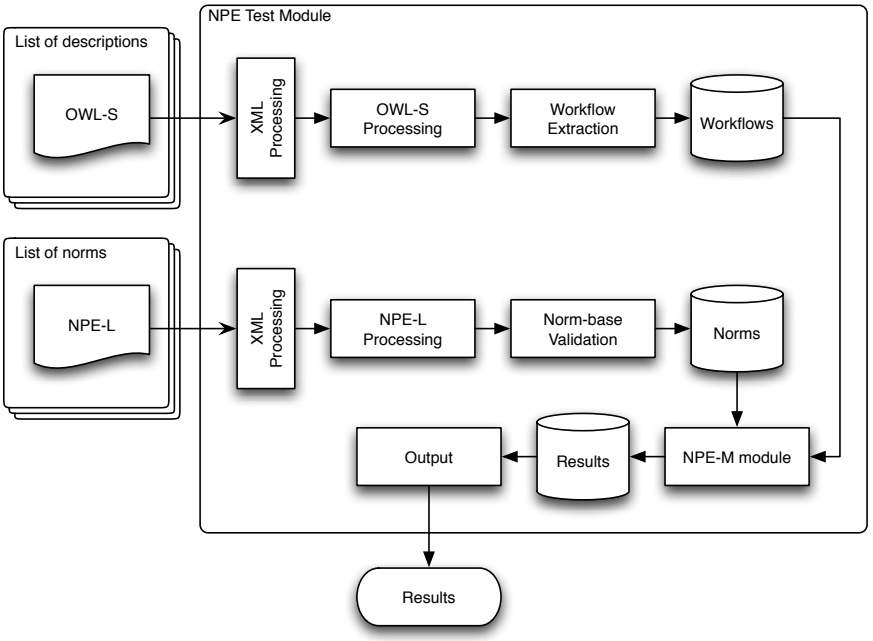

Figure 6.1: NPE Module Design

6.3.1 Existing Modules

A modular design pattern will be employed for the NPE module. As has been explained in the preceding sections, where possible, existing software modules should be employed to provide as many of the capabilities required as possible. This will reduce the time to code and improve the reliability of the final module. A search will be conducted to identify the most suitable modules for implementation. The primary focus for this search will be on the location of modules which are already used in the spheres of service based, web based and semantic web based computing. Satisfactory modules will be considered for use in the NPE testing module. Any gaps between modules will have to be filled by

modules produced specifically for this thesis. It is acknowledged that a number of these new modules will have to be produced. And as such it is not critical to find existing modules which fulfil all requirements. The search will focus on those which have common uses and can be employed to provide substantial time savings in development.

Figure 6.1 shows the initial high-level design proposed for the NPE test module. Essentially this design breaks down into four distinct areas: an input phase, a matching phase, a processing phase and an output phase. Requirements for modules within each phase will differ as they each have very different responsibilities within the system. Table 6.1 outlines the basic modules which are expected to be required and the prerequisites for use. The search conducted for existing modules will centre around fulfilling the requirements within Table 6.1. It is expected that the modules discovered will not exactly fill the requirements stated. In these cases, a preference for larger modules offering overlapping capabilities will be favoured over the use of numerous smaller modules. This decision is primarily intended to simplify the design and integration process as well as making long-term servicing simpler.

Module	Requirements
XML input	Process OWL-S, NPE-L and settings XML files. Validate XML and handle errors gracefully.
XML output	Convert results to XML for output.
OWL-S processing	Process input OWL-S into objects or other internal representation. Prepare documents for model extraction.
NPE-L processing	Process input NPE-L into objects or other internal representation.
Model extraction	Extract process models from OWL-S.
Norm-base validation	Validate NPE norms. Ensure validity of existing norm base while adding new norms.
Model storage	Store extracted models.
Norm storage	Store validated norms.
NPE processing	Reason over input service process definitions using normative facts.
Results storage	Store NPE-R results.
Output	Choose a service based on results set.

Table 6.1: Module Capability Requirements

Chapter 5 outlines the design and production of the module NPE-M. This module fulfils the role of instantiating the rules and beliefs behind NPE. It contains the majority of the logic code required to process workflows, handle comparisons between process activities and norms, as well as logic to handle norm life cycles. This module has already been created and is part of the NPE package. Essentially this test phase centres around the creation of a fully featured module (the like of which could be used easily

by any implementing system), wrapping the NPE-M module and adding functionality surrounding importing, exporting and storage of XML formatted data. This NPE-M module is likely to require some tweaking during this prototyping phase and so it should be treated as an existing module, but one which can also be altered when needed. Additionally, the module required to handle NPE-L documents is already partially created. During Chapter 8 when norms were created as part of an automated test case generation task, these norms were exported as XML formatted NPE-L. To carry out this exporting the NPEL-API module was created to handle NPE-L documents. Although the exporting of norms to NPE-L documents will not be required as part of this or any future testing process, a large quantity of this module has already been authored. Further details about the implementation of NPE-L importing can be found in Section 6.3.2.1.

As well as handling NPE-L documents, the NPE module also needs to be able to read documents defining how parterns will collaborate. These could potentially take any form, however, for the purposes of this thesis they take the form of OWL-S process models. In Chapter 5 the NPE processing module is described as having a built in OWL-S processor. This description needs to be clarified for testing as it is ambiguous in terms of to how much processing of NPE is to be carried out by the core NPE-M module and how much is carried out externally. In the case of testing, initial test runs will rely on using a separate OWL-S processing module to produce an in-memory OWL-S module which can be stored or fed directly to the NPE-M module. One of the aims of this chapter is to explore whether it is better to process input documents this way or internally to the core module.

An investigation was carried out as to whether it might be better to separate the XML processing from data processing and use a JAXP processor, for example. This idea was rejected as the XML processing built into the OWLS-API and latterly the NPEL-API, offer comparable performance with greater ease of deployment.

Storage of normative facts, process activities and associated structures are required to enable the buffering of input data. Numerous options were considered from pure in-memory solutions such as storing information as objects, through to heavy solutions such as the use of relational databases such as MySQL. After much research (some of which contributed to sections in Chapter 2), it was decided that as the OWLS-API and NPEL-API are both backed by the Java-based Jena semantic web framework, Jena would be a suitable processing and storage engine. Jena offers in-memory and persistent storage of RDF data as well as support for SPARQL queries and reasoning backed by the Pellet OWL reasoner. Jena and Pellet have also been successfully used together during this thesis, for reasoning over the creation of test cases in both Chapter 7 and Chapter 8. Both have been demonstrated to be more than adequate for the requirements of the NPE system.

6.3.2 Required Modules

The previous section details the packages and APIs selected for reuse during the creation of the NPE test module. As can be seen by comparing the modules selected in Section 6.3.1 with the requirements outlined in Table 6.1 and Figure 6.1, the majority of the functionality required is already provided by these modules. Very little extra code is required for a working NPE based system to be produced. This is as a result of the ethos of reuse carried through this thesis and underlines theory behind the design decisions taken earlier in this chapter.

Of the remaining modules still outstanding, the most significant are the output (or choice) module and the import section of the NPEL-API. The option choice module is straight forward and will receive most tuning and attention in the next chapter. Responsible for the selection of a final valid Service (process model), the output module could be seen as optional. However, as the purpose of this thesis is to produce a system capable of selecting a single service for interaction, this module is seen to be important for testing and validation. In this chapter the output module is limited to the selection of a single Service based on the lowest score passed by NPE.

6.3.2.1 The NPEL-API

The NPEL-API has been produced during investigations into the creation of the NPE ontology. For the full version of NPE this API must have the ability to not only export norms, but also import and store them. The design of the NPEL-API is similar to that of the OWLS-API and OWL-API. At the heart of the module is a storage system backed by Jena. This along with the OWL-API enables the loading of the NPE ontology generated in Chapter 4.2 and the processing of XML-based NPE-L to Java objects. The design of this API loosely follows an MVC pattern, with specific NPE model classes and structures representing NPE concepts in memory, a set of NPE specific controllers wrapping the data store and a single view exposing querying functions to external implementers.

6.3.3 Development and Operational Considerations

The finished NPE testing prototype was written in Java as this is the language most common among the modules discovered during research. Java is also a language which is familiar to the authors and has a background as one of a few often used in agent and semantic Web technologies. Thus a Java based NPE testing module will be easily adapted during initial testing and suitable for use for full experimentation in Chapter 10. The design and inception of the final NPE testing module was straightforward owing to the modular approach taken during design. The majority of the features required to test NPE have been sourced from external modules or have been created separately. The

research carried out to find or design sub-modules can be found in the previous sections. The only major section of code which had to be written at this point was the linking and control logic joining separate modules together. This logic mainly controls concurrency and the flow of data around the testing structure.

Two diagrams are included to help explain the make-up of the NPE test module. Figure 6.2 describes the deployed module set as is correct before the first round of testing. Of note in this diagram is the inclusion of flows of information back to any logging service used at runtime. It should also be noted that information is passed back from four points (NPEL-API, OWLS-API, NPE-M and the chooser), allowing easy debugging of problems and monitoring of progress. Figure 6.3 shows the basic activity plan for a run of the NPE test module. This does not include any underlying architecture. It is presumed that one run of this activity diagram represents a single NPE test using a single test case. It should also be noted that this diagram only includes a subset of the activities occurring as part of the NPE-M core module. Full information about the runtime of NPE-M can be found in Chapter 5. It has been omitted to save space. Within the diagram, decisions 1,2,3 and 4 form loops which loop over the previous code while there is still an input to process. The decision labels have again been omitted to save space. Finally, it should be noted that although the test factory does not execute tests simultaneously, the processing of OWL-S and NPE-L documents within the test module is achieved concurrently. This decision was made for the purpose of improving the performance of the test module. Currently, the execution of NPE-M is not handled concurrently so that each service is processed separately.

6.3.4 Input and Outputs

Before an NPE based system can be fully implemented or tested, a set of test cases must be created in order to analyse the performance or the system. Test cases are essentially "a set of test inputs, execution conditions and expected results developed for a particular objective, such as to exercise a particular program path or to verify compliance with a specific requirement" [IEEE-STD-610]. In this case, the test cases are created in the form of OWL-S documents describing services and NPE-L documents describing normative facts. As has been previously explained, OWL-S has been chosen for the default service description language for NPE based systems. Languages other than OWL-S could be used, but the scope of this thesis is limited to services providing descriptions in OWL-S only.

A set of norms that will be included as an input for the NPE module during testing along with a test case from the descriptions test cases (Chapter 7). The combination of a service description and a norm catalog go together to form a full NPE test case.

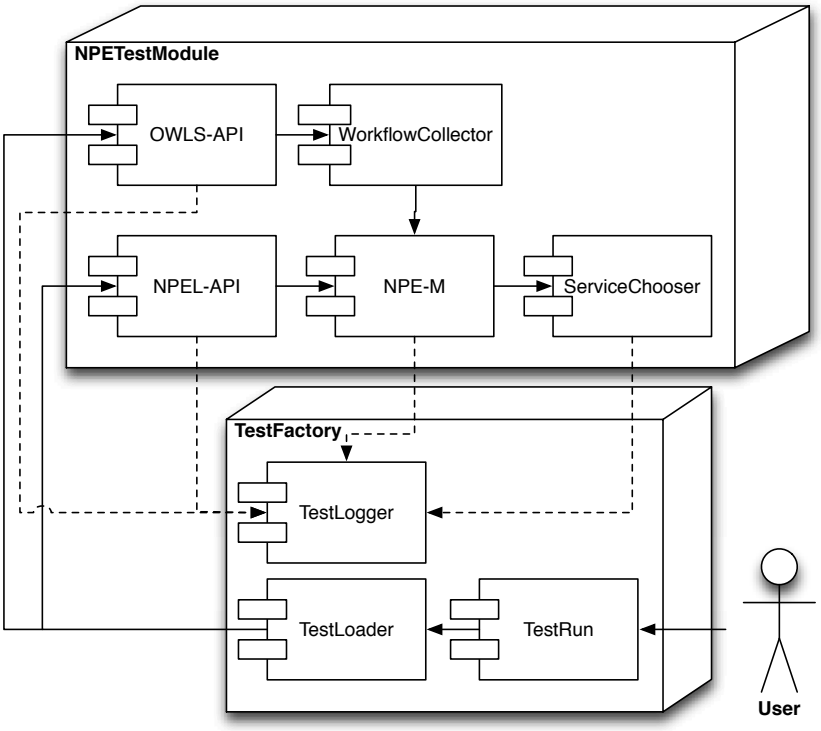

Figure 6.2: Testing Structure Deployment Diagram

The research in Chapter 7 and Chapter 8 presents a complication which has been uncovered during this thesis; that as both the field of normative autonomous computing using semantically aware actors and the field of service selection using complex process models are relatively young and niche, there is almost a complete lack of well rounded test cases for the evaluation of systems. Chapter 7 and Chapter 8 shows that for each a body of research exists (for the area of OWL-S test cases this body is extensive). However, neither offers a solution of a high enough level of complexity to warrant using NPE (over existing profile and input/output matching techniques).

As a result of this deficit the question of what inputs are be used to assess a normative service selection approach must be answered before testing of the original hypnosis and assumptions can be started. For this reason, Chapter 7 will outline the production of a set of service based test cases and Chapter 8 will outline the production of a set of norms which relate to the service test cases.

In generating these test cases, it will be critical that, where possible, bias is eradicated from the production process. Injecting bias at this point will be easy as would damage any final results, making answering the research questions as posed in Section 1.4 almost impossible. Further discussion of bias can be found in Chapter 7, Chapter 8 and at the end of this thesis in Chapter 11.

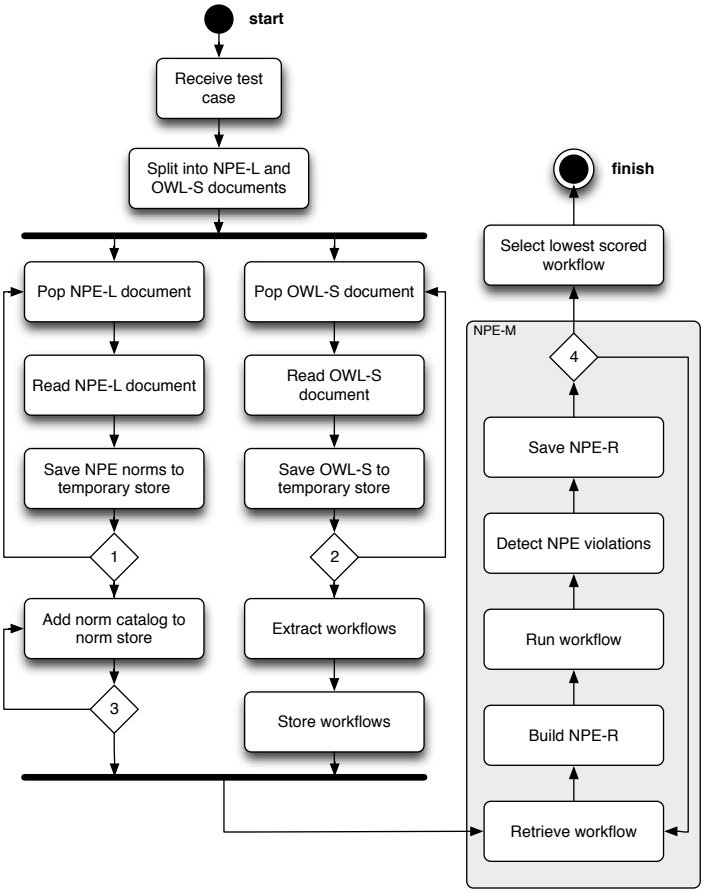

Figure 6.3: NPE Test Module Activity Diagram

6.3.4.1 Internal Norms

In addition to building a set of input test case norms to be imported at runtime, Chapter 8 will also build an additional set of norms to be held internally within the NPE system. These consist of a small set of globally important *super* norms which are held in memory and test not only the effect of the presence of a number of consistent norms, but also the potential cost of importing any norms at runtime (when only the internal catalogue is used).

6.4 Service Selection as a Dynamic Process

This research treats service selection as part of a wider dynamic process involving actors interacting with services (or potentially other actors). There is an existing well-formed body of research into service selection in dynamic environments, much of which has been identified and highlighted in Chapter 2. This research leans heavily on existing research for answers to questions, including how best to test any final normative selection module.

More traditional semantic based service selection initiatives have relied on fairly simple tests involving the repeated execution of a proposed algorithm in a static or limited environment (Sycara et al., 1999; Kim et al., 2007; Klusch et al., 2008; Barakat et al., 2012b). This poses numerous problems for the approach outlined in this thesis, most notably that norms are of most use when they are acting to regulate autonomous actors in an environment. In addition to the extra complexity brought to this approach by reasoning over the processes rather than the profile of a potential actor this has led the authors towards carrying out verifications using an environment of higher dynamism, rather than a static test rig.

6.4.1 Agents as Actors

This approach links the areas of semantic based service selection with that or normative agent research. Borrowing from both will allow a norm-based system to better distinguish between similar services.

Although the module proposed in Chapter 5 was created with agent use in mind (and as has been proposed in the previous section will be tested in a dynamic agent-based environment). The NPE-M module could potentially be tested in any setting. As has been mentioned in Section 6.3, one of the aims when developing a testing environment for the module is to enable its use in virtually any environment.

The choice of agents to be actors within any system for the purpose of testing has been carefully weighed against the potential complexities it brings. It has been decided that for final verification of any answers to research questions (as laid out in Section 1.4) agents should be used as containers for NPE based selection modules. This research does acknowledge that this choice may be seen a strange by some who have experience with existing service selection research where tests are often conducted in closed environments without the frameworks of a multi agent system. This choice was motivated both by the close ties this work has to normative agent research (as agents are likely to be actors acting within the constraints of a formal normative environment), as well as a wish to find a setting capable of offering the best tool set with which to evaluate any solution. An agent-based system has been chosen so simulations can observe how a selection in such an environment might be effected by utilising a normative approach. For these simulations agents are used as "dumb" holders for the NPE system, stripping them of any advanced social or proactive traits (which might otherwise, be seen as key for their agency). However, this deficiency is accepted and countered by suggesting that this thesis is only interested in focusing on one small step of a larger agent reasoning cycle and that for this step the suggested system is extracting all the social notions needed (namely normative reasoning, communication with a single authority and rational decision making).

6.4.2 Agent System Design

The general plan for the investigation of the NPE system is that the NPE-M module will be reused in conjunction with a new agent based testing structure. The primary goal of this new structure will be to simulate as accurately as possible a scenario when service selection might be aided by an NPE based approach. For this to occur the test module produced previously will be extracted from the test harness, wrapped in a well-defined interface to simplify implementation and hide the internal workings from the new implementation and inserted into a new agent-based simulation. This agent system will simulate a potential use case for an NPE based service selection actor and will enable comparisons to be drawn between configurations of NPE and non-NPE systems.

The final NPE module as exposed to any host implementer is outlined in Figure 6.4. There will be no further discussion of the internal workings of this module during this chapter as for the purpose of these simulations, the NPE system is a tool for the production of service ratings and the central focus of this investigation is the interplay between the agent containers, service descriptions and NPE catalogs.

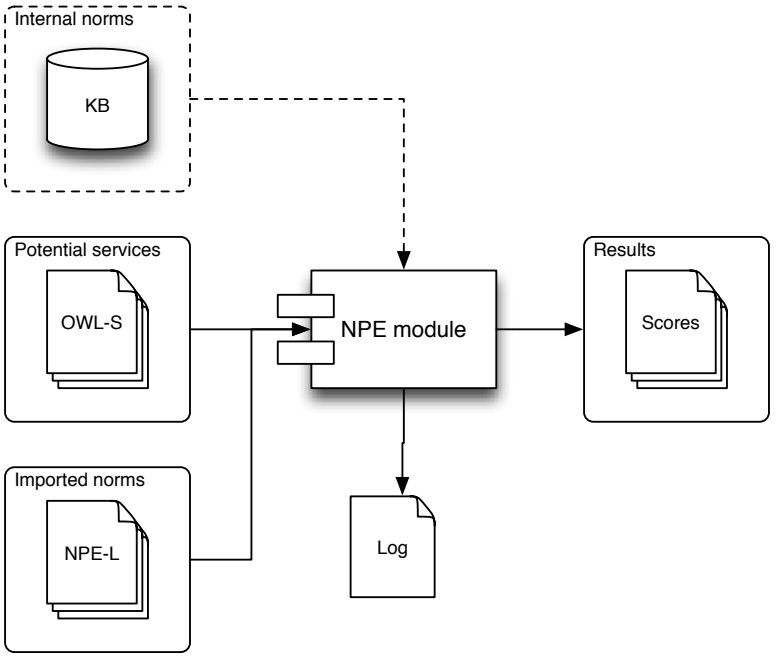

Figure 6.4: NPE Module

6.4.2.1 Agent Frameworks

During the course of this thesis, it has been foreseen that the primary use for a workflow grading (service selection) system, such as the NPE-M core module, would be in the sphere of agent-based computing. Specifically, the capabilities offered by a norm-based

selection technique would enhance the automated processing of potential resources in an autonomous agent operating without human interference. Thus, it has been decided that an accurate simulation of the potential use for NPE should involve an agent-based environment. The final round of full tests is to take place as a set of simulations utilising numerous agents reasoning over norms and potential services. To make the creation of a test ecosystem and the incorporation of the NPE module simpler and closer to standard practices, an agent framework is to be utilised. Frameworks have been discussed previously in Section 2.2.2. In this case, a multi-agent framework is to be deployed to act as a host and container for a number of test agents.

Numerous agent frameworks have previously been explored during research carried out for Chapter 2.2.2. Several attempts have been made to produce an example simulation environment in research carried out externally to this thesis. These have primarily focused on two frameworks: the Jason agent environment and the Jade agent framework. At the start of this research Jade was considered the suitable choice for simulation environment. Its status as the widest used framework backed by a solid body of existing research made it an obvious candidate. Initial tests using Jade have confirmed that the inclusion of a simple resource selection module (as an alternative to the NPE module, used as a trial) is possible and produced positive results. As part of a number of pre-thesis tests the Jason agent environment was also subjected to a set of trial simulations. A number of agents were created in a Jason container equipped with the OWL-S API and a profile model based selector. The initial Jason tests produced results similar to those of the Jade platform. The choice for final simulation was informed by these initial findings as well as research gained throughout this work. The Jason agent environment was chosen over Jade for its tighter environmental controls and full BDI implementation. Information on BDI and Jason can be found in Section 2.2.2.

6.4.2.2 Agent Environment

For the purposes of this thesis an environment in which to simulate one or more agents reasoning over a service selection problem is required. This will necessitate the deploying of a container in which to run tests, as well as structures to support basic agent to agent or agent to service communication. Any environment should be quick to deploy and rely on standards commonly used in the field of agent-based computing. Essentially any container shall provide a high-grade base on which reliable unbiased results can be collected. This thesis is interested in the operation and performance of NPE and not of any underlying container. For these reasons and those detailed above, the Jason agent system has been chosen and deployed to aid in these simulations. These simulations will take place either from within a hosted Jason instance on a remote server, or from within a Jason instance operating out of the Eclipse IDE utilised previously.

Jason in relation to these simulations will be required to be able to host numerous agents in a reliable manner. These agents should be capable of both autonomous action and the ability to communicate with each other when needed. The underlying Jason framework will provide the container in which these agents can operate, along with structures to aid design, processing and communications.

As has been mentioned previously, Jason is a multi-agent platform centred around a generalised BDI agent architecture. Agents participating in a Jason based interaction operate within a strict environment regulated by the Jason container. Written in Java, this container provides a familiar structure in which agents can be designed and tested. Jason removes much of the effort needed for agents to operate, including handling inter-agent communication and internal reasoning cycles. Agents cooperating within the Jason environment are designed and programmed using the AgentSpeak BDI agent programming language. Jason operates as an interpreter for an extended version of the AgentSpeak language. For these simulations AgentSpeak acts as a reactive planning vocabulary; a mechanism for the production of plans as an input for an agents BDI cycle.

Structure The structure of the agent community to be used for the simulation of NPE use will loosely model that of numerous agents competing to find and access services. Each agent equipped with NPE will be a separate entity unaware of other competing agents. This will all take place in a single Jason container using agents scripted in AgentSpeak. At the heart of this simulation is a single administration agent responsible for the provisioning of test cases as well as the logging of results. A single administrator was selected over a previous idea surrounding three separate administrators (one for norms, one for services and one for logging) due to its simplicity in deployment. Having only one administrator also helps to keep the overheads associated with execution to a minimum.

Administration The administration agent is the centre of the simulated environment in which NPE is to be tested. As has been stated in the previous section, a single agent has been selected to administer all areas of the simulation. This agent acts as the gateway through which test agents can access sets of NPE-L catalogues and OWL-S documents. Test cases constructed of NPE-L and OWL-S documents are aggregated into specific groups as detailed in a predefined test protocol (see Section 10.1.3). Each agent is registered with the administrator before runtime, registration maps the agent to an agent object in the test plan. When an agent requests a set of service options from the administrator it is provided with a predetermined collection of service descriptions and a norm catalogue based on the current test run number and its own identity. Results sent back to the administrator are stored along with a brief performance profile and sparse logging data. On all of the test agents either finishing their allotted tests, or timing

out, the administrator agent collects all of the results and returns them to the user. A high-level overview of the administration agent can be found in Figure 6.5.

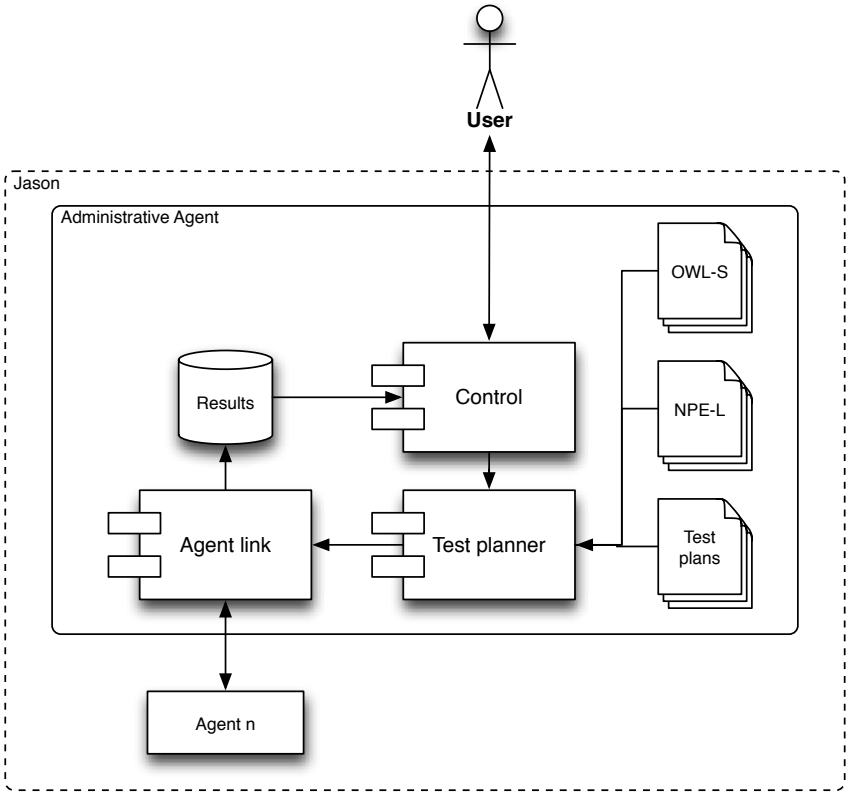

Figure 6.5: Administration Agent Workflow

Reporting The administration agent is the primary source of runtime information and results. To reduce excess loading during simulation very few messages will be returned to the user in real-time. The majority of information will be passed back as a set of formatted results following completion of the simulation. These results include the ratings given to each OWL-S document (in the form of NPE-R documents), the final choice of service, performance information and error rates. Logging from individual agents is reduced and furthermore the small amount of logging still utilised is to be returned to the administration agent on the completion of processing. It is possible to run Jason from within the Eclipse IDE. As such a similar profiling technique as used for the original five test runs can be used to further investigate the performance of individual parts of the simulation. As required, this service was also used to examine potential performance issues as and when they might arise.

6.4.3 Simulations

Simulations, follow on from testing, take the system formed and apply lessons learned from test runs. The set of simulations undertaken can be viewed as the experimental part of this thesis. They take the form of a number of multi-agent based simulation runs intended to depict a realistic use for an NPE based system. It is through these simulations that NPE has been benchmarked against alternative systems. All of the tests undertaken during this chapter are automated with minimal human input required. Protocols which are to be used for iteration control and any risks associated with implementing simulations will be detailed in the following sections. All of the inputs used here rely on the service test cases created in Chapter 7 and the norm test cases created in Chapter 8. No new test cases or other input data (other than simple configuration information) will be added to the simulation protocol during this chapter.

Simulations are the imitation or enactment of a proposed "real world" task for the purpose of testing or proving of a system. To test NPE a number of simulations have been proposed, all of which will utilise the structure outlined in Section 6.4.2.2 and the testing workflow shown in Figure 6.6. The basic simulation structure is that of a Jason agent container, acting as a closed world simulation framework. This container has the ability to host a number of agents, a simple repository for test cases (OWL-S documents), a simple repository for holding NPE-L catalogues and the ability to log results (the final three are provided by a single administration agent). This container will host six agents: a control agent and five NPE based agents. For the purpose of this thesis, the agent not utilising NPE will use a basic IO matching algorithm based on the OWLS-API to choose a service.

Of the five NPE based agents, all will utilise the NPE system to rank provided service descriptions. Table 6.2 details each of the agents utilised while testing NPE. Agent 3 can be viewed as the baseline NPE agent as it is set up to rate all services (process models) and exit when an unrecoverable exception is raised (through the violation of an unrecoverable norm). Agent 2 tests the effects of NPE when run without exiting on unrecoverable violations (if all service process models are processed then the results are processed). Agent 4 tests the effect that over harsh use of norms will have on service description recall, this agent not only exits processing on unrecoverable violations, but also when the total cost of the sum of violations exceeds a low threshold. Finally, Agent 1 utilises the same NPE system as Agent 2 (processes whole models), except the input descriptions, are limited by first passing them through the IO matching module used by Agent 0. This simulates using NPE to help an agent using a traditional matching approach who then wants to improve their choice by using a normative approach. For all of the agents involved in testing and simulation, when multiple descriptions are ranked or selected together a single description will be selected at random.

In a single simulation run, each agent is served a selection of OWL-S descriptions divorced temporarily from their test cases (see Section 10.1.3 for more information about test case groupings). Simulated agents may also be passed a single NPE-L catalogue. Both these variables are served from repositories within the simulation container. Responsibility for choosing the correct combination of description and catalogue is delegated to the administrator agent within the container. The mappings for each run follow the sets laid out below in Section 10.1.3.1. Agents then reason over the combination and generate a ranked list of services and a preferable service. At the end of the run, the lists are sent to the log file together with data about performance and runtime loads.

The goal is to produce tests which simulate as many combinations of descriptions and norms as possible for a minimal run of simulations. Each simulation will initially run ten times (this testing process is similar to that used in Maximilien and Singh (2004) and many of the papers listed in the below sections). Results will be passed to a log file where they can be evaluated. During these simulations, this research will be aiming to test the overall performance of NPE equipped agents, as well as the reliability of NPE based systems.

Agent ID	Matching technique
Agent 0	Profile model based IO matching only
Agent 1	IO matching followed by NPE (catching exceptions)
Agent 2	NPE (catching exceptions)
Agent 3	NPE
Agent 4	Strict NPE (with an upper limit on cost)

Table 6.2: Testing Agents

6.5 Evaluating performance

Evaluating the performance of a norm-based service selection module (with regard to existing alternatives) is likely to be difficult due to the lack test cases for both norms and services there being no current best practice for testing the selection of services which expose complex process models. The Service Selection Contest (Klusch, 2012) and work such as Küster and König-Ries (2010) have looked in detail into the grading of service selection (often matchmaking) with large sets of atomic processes, and this thesis recognises that work. Additionaly, there is a wish to not expand the scope of this research into the wider reasoning motives of agents. All of these issues have been raised previously and solutions have been proposed. The testing and validation system proffered in this chapter aim to provide a fair and unbiased platform on which to evaluate the performance of NPE-M module and underlying normative model.

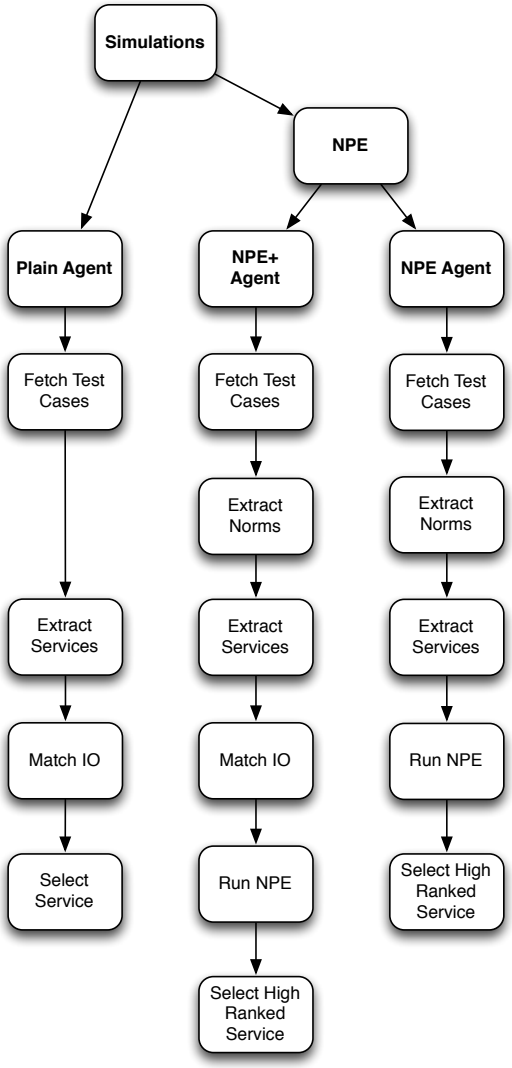

Figure 6.6: Proposed Simulation Workflow

The choice of metrics with which to evaluate the performance of a system utilising an NPE based approach also have to be carefully considered. This research borrows primarily from research in the field of service selection when looking for reasonable metrics with which to evaluate the impact of NPE. This thesis has chosen to utilise two metrics in particular: time to run and precision. Time to run is a simple method for deciding the impact of an addition to an already functioning system. Used to evaluate numerous existing systems (Yu et al., 2007; Bonatti and Festa, 2005; Barakat et al., 2012b; Padovitz et al., 2003), this approach requires that the runtime of a single selection process is measured so that the impact of adding additional processing can be evaluated. This approach is a sensible starting point for the evaluation of a selection system for use in an open agent-based environment where multiple agents may be competing and time is a precious resource. If the addition of an NPE based reasoner to the process causes an agent to be adversely affected in terms of the number of services he/she can utilise in a

given time period, any advantage gained by increasing the precision of service selection (see the next paragraph) will be lost.

The second metric to be used to evaluate the performance of any system utilising an NPE based approach is that of precision. Precision is a well-used term within service selection (and computing as a whole). This research takes its lead from (Klusch et al., 2005) in measuring the precision and recall with which an actor retrieves services selected services. This approach relies on the calculation of a weighted average of the precision and recall for an actor deriving a score from 0 to 1 (Rijsbergen, 1979). The general calculation of the precision, recall and derived score (evenly weighted harmonic mean) for a single test run (multiple service selections) can be found in brief in Figure 6.7. For a fuller explanation of precision and recall Klusch et al. (2005) should be referenced. Alongside time to complete, this will serve to give context to any utility gained by an actor implementing an NPE based selection technique (over that of a similar existing technique).

$$Correct\ selections = TP \tag{6.1}$$
$$Returned\ selections = TP + FP \tag{6.2}$$
$$Possible\ selections = TP + FN \tag{6.3}$$
$$Precision = P = \sum \frac{TP}{TP + FP} \tag{6.4}$$
$$Recall = R = \sum \frac{TP}{TP + FN} \tag{6.5}$$
$$F1 - Score = F_1 = \frac{2PR}{P + R} \tag{6.6}$$

Figure 6.7: Calculation of the Precision, Recall and F1-Score

6.6 Conclusions

This chapter has proposed two linked methodologies for the testing and validation of the NPE-M module proposed in Chapter 5 and solving of the research questions laid out in Section 1.4. These aim to help improve the understanding of uses for norms in service section and to provide answers to the research questions. In designing, testing and simulation methodologies, this research has attempted to take into account the removal of bias and introduction of fair checks and reasoning where possible. It is acknowledged that any simulation process will not be perfect and have laid down a case for the chosen agent based approach. By utilising agents, this thesis is able to lean on numerous well researched tool-kits to provide additional support as well as demonstrating for any potential future user the benefit of normative service selection in potentially autonomous environments.

At this point in this thesis all of the solution requirements as stated in Section 1.4.2 have been fulfilled. To reiterate, they are as follows:

- Provide a machine readable mechanism for the representation of norms to be used in reasoning over process expectations.
- Have ties to existing Semantic Web technologies so as to fit within the existing Web service stack (Section 2.1).
- Provide the ability to reason over both complex and simple sets of service based process model inputs.
- Consume over both complex and simple sets of normative inputs.
- Operate so as to enable the collection of data to answer the governing research questions posed in Section 1.4.
- Provide a set of input norms for the testing of any proposed solution.
- Identify or create a set of Web service descriptions to act as test cases for a service selection procedure. These descriptions should describe services which are similar enough that exisiting IOPE solutions would fail to differentiate between them, and yet where some expose processes which are not too be expected for any given set of normative beliefs.
- Identify appropriate methodologies to test any solution against the research questions posed in Section 1.4.

The following chapters will enact the methodologies laid out here, including the creation of input test cases for use in further simulations. In brief, Chapter 7 and Chapter 8 will cover the generation of test cases for use in evaluating NPE, Chapter 9 will detail the initial static testing of the selection module and Chapter 10 will present the final simulations along with analysis and verification of the research questions as laid out in Section 1.4.

Chapter 7

Test Cases: Services

This chapter contributes a mechanism for the production of OWL-S test cases. 24 new test cases have been produced using this mechanism, these test cases will be open sourced following completion of this thesis.

Before an NPE based system can be fully tested, a set of test cases must be created in order to analyse the performance or the system. Future chapters will outline the evaluation procedures which will be used to determine whether NPE has fulfilled the hypothesis of this work. This chapter focuses on the creation of a set of test cases for descriptions of services over which NPE might be asked to reason. These test cases have been produced to fulfill the solution requirement outlined in Section 1.4.2 that a set of input service descriptions will be required to help answer the research questions posed in Section 1.4.

There is an ever increasing number of Web services available on the Web. Although compared to the number of sites proving HTML-based data and services over HTTP the number of Web services is still small, even this small number is large and diverse enough to warrant an equally thorough and diverse set of test cases. As such a set of descriptions must be created which can adequately represent both the current uses for Web services and the most likely future uses. This work has identified three key areas in which Web services might be used; information gathering, service provision and purchasing. These three areas have been chosen as the three categories around which any test cases will be based. Each category has been chosen as it represents both an area of business which currently uses Web services (or where example test cases are available, ie. in the OWL-S TC Klusch et al. 2008) and a type of services which is different from those in other categories (services in each category provide a different type of service to the end user).

Test cases will also have to be created carefully to eliminate any bias which may be inadvertently introduced. Bias is especially important to identify in test cases which are wholly or partially created by the system developer. In creating these test cases every

effort must be put into creating a set of tests which will provide a fair representation of the information the NPE will be expected to process. A test set which is biased towards proving an initial hypothesis will only serve to undermine the validity of any results. Later in this chapter more information is given about how bias will be eliminated in this test case set.

This work acknowledges that the three categories of services chosen are only a small subset of all the possible applications for services on the Web. These three provide a good representation of the uses for services and test NPE against a wide range of differing tasks.

Within these three categories will be five similar service descriptions. These descriptions will be differentiated by the metric of perceived "reliability" derived from the behaviours they expose in their process models. Each service will be ranked according to its perceived reliability so that at runtime an astute actor should always choose a higher ranked service. In the testing of NPE, this ranking score will be hidden and the NPE module will have to provide its own ranking based on the process model.

7.1 Test Case Quality

The preceding paragraph introduced the notion that some services can be reliable and some can be unreliable. This work treats reliability as a degrading scale rather than a black and white issue. Of the five services within a single category one will be rated as most reliable and the other four will have progressively lower reliability scores. An actor working purely on the idea that the only service to interact with is the most reliable will obviously only choose the most reliable. However, other actors may be able to sacrifice reliability for other metrics such as lower cost or ease of use. Actors may also sacrifice reliability to achieve a goal which may otherwise not be possible.

It is the introduction of unreliability to the test cases which presents the greatest threat for adding unwanted bias. The five services all have to be relatively similar so that only an NPE-based system could differentiate between them. At the same time, the differences must be introduced in such a way as to challenge NPE. Differences must not be introduced to match requirements set by NPE or any catalogue of norms.

To generate five services of differing reliability, a starting point is needed. Two starting points can be found: it is possible to either start with an original service and degrade it, or to start with an imperfect services and improve it. There are few test cases of OWL-S services available which have been published. None of which have been published demonstrate imperfect (or degraded) services. For this reason and because it was deemed to be a simpler task to degraded a service, the set of test cases will start with an original service description. Each of the three original services (one from each category) will be

taken from the OWL-S TC test case library. This is one of the few published OWL-S test case repositories and although relatively poorly documented, it does provide a large number of pre-verified service descriptions.

From a starting point of one description at runtime, four other descriptions will be produced. This procedure is carried out once per run by the testing service discovery repository. Mutations will be carried out based on a set of rules held by the repository. Each mutation has a cost which will be appended to the test case as its reliability score. For each of the four unreliable service descriptions, one will receive one mutation, the next two, the next three and so on. Mutations will be selected pseudo-randomly in an effort to remove as much bias as possible. Once the mutations have all taken place, the test will start and the descriptions distributed.

7.2 Base Test Case Generation

The test case generation procedure outlined in this chapter relies on the initial input of a number of original service descriptions. These descriptions could be generated by hand as part of this research. However, it has been decided that in an attempt to reduce bias and improve the quality of any test cases these descriptions should come at least in part from third party sources. In the field of OWL-S service descriptions, these sources exist as repositories and small collections of documents published on the world wide web. As this section will explore, one of the main obstacles to the creation of test cases and testing regimes based around OWL-S scenarios is the lack of available OWL-S descriptions. For the purposes of this thesis, only service descriptions which have been verified before release and published in reputable locations will be of sufficient.

7.2.1 Existing Repositories

As with many of the requirements for this thesis, the ability to import existing information for adaptation or reuse can both reduce the time taken and improve the quality of any results. There is little point in recreating existing information if a wealth of good quality examples already exists. To do so would be a waste of effort and most likely lead to results which could be easily challenged. For this thesis, this is especially true of the test cases used for evaluation of NPE. It is vital that any test cases used are as free from bias as possible, as well as being well-formed and taken from as wide a range of applicable uses as possible. The research needed to produce a comprehensive set of OWL-S test cases would take up a second whole Ph.D. As the aim of this thesis is to investigate norms and NPE, rather than to create a set of test cases for testing OWL-S service selection techniques, the author determined that it would be wise to use as many existing resources as possible.

The best source for existing OWL-S examples is on the Web. However, supply is limited. For example, there are no large-scale commercial repositories or directories as there are for WDSL. Industry is simply choosing not to describe their services using OWL-S and when they are any descriptions are either used internally or provided as extra information with the service documentation. This section will not look into why currently there are no large commercial repositories of OWL-S documents; any such discussion is contained in Section 11.5.2.3. It does, however, acknowledge the problem.

There are a few research-based initiatives providing OWL-S descriptions. The most relevant of which are listed in Table 7.1. These sites provide example OWL-S descriptions to form the basis for an OWL-S testing set. They vary in size from large test case sets to smaller example sets. On the whole, the larger sets are more useful to this thesis as they provide large numbers of verified service descriptions, whereas the smaller sets are more useful for one-off demonstrations as they tend to provide singular more complex examples (quite often with more documentation). Two of the largest repositories are outlined in the following subsections. These two repositories are important as they both offer a wide range of descriptions which may be freely used for research. However, as will be shown, they are both flawed in many areas.

Included in Table 7.1 is a mention of 48 atomic processes created by this work. This set will be described in more detail in Section 7.3.3. This is not a set of complete OWL-S descriptions. Instead, it is a set of activities which have been extracted from other sources. These may be composed to create complete process models which can then be used in OWL-S descriptions. More information on the uses of this set can be found in Section 7.2.2.

Location	Count	Notes
OWLS TC	1083	Atomic processes only
OPOSSum	2851	Collection of SWS descriptions in a variety of languages
Mindswap OWL-S Examples	11	Collection of examples
OWL-S API Examples	13	Example OWL-S documents
SWS-TC	240	Slightly outdated
NPE OWL-S Processes	48 (Processes)	A set of OWL-S atomic processes

Table 7.1: OWL-S Sources

7.2.1.1 OPOSSum

The Online Portal for Semantic services is an initiative dedicated to the creation of a Semantic Web service test case set. Created by Dr Ulrich Kster at the Friedrich-Schiller-University Jena, OPOSSum is essentially a database of some 2851 semantic Web service descriptions in a variety of languages. Descriptions are added by individuals or organisations and community participation is encouraged. Access to this collection is provided by a web interface as well as via SQL to the underlying MySQL database (read-only). Initial queries can be easily handled by the provided web interface, with results available in web-based form or as a zip file containing all returned service descriptions. This collection is an admirable start to a large repository of OWL-S descriptions. However, it does have a number of limitations: it is small in size, nearly half of the descriptions included come from one collection (the OWLS-TC outlined in Section 7.2.1.2). Only a handful of the descriptions listed in the OPOSSum database actually have any concrete grounding (describe implemented services). The lack of a SPARQL endpoint also would appear to be a glaring omission by the designers for a resource which is supposed to enable the testing of semantic services.

The authors have collected numerous services from the OPOSSum database. However, it recognises the limitations of this resource as a source for semantic Web service test cases and as such has made it one of a number of sources.

7.2.1.2 OWLS-TC

The OWL-S Service Retrieval Text Collection is a set of OWL-S service description and request queries, created by Dr. Matthias Klusch and his colleagues working at the German Research Center for Artificial Intelligence, Saarbrcken. This collection represents the single largest and most mature set of OWL-S services. Currently at version 3.0 revision 1, the OWLS-TC represents a concerted effort to form a set of OWL-S descriptions to support the evaluation of OWL-S service matchmaking algorithms. The collection comprises 1007 services written in OWL-S 1.1 and divided into seven domains (see Table 7.2). Using the OWLS-TC is straightforward for anyone with a local web server installed and all of the contained descriptions have been thoroughly checked and tested. This collection is a good starting point from which to create a set of test cases due not only to its size and maturity but also because it has already been used in numerous peer-reviewed research papers as the basis for test scenarios (Klusch et al., 2008).

The OWLS-TC is suitable for many research applications especially those looking to test applications working with OWL-S. However, this repository is limited. Even in version 3.0 many of the 1007 services are almost identical, differing only by name, or minor variation in parameter type. This is of minor concern compared to the issue of service complexity: the OWLS-TC is limited to providing atomic OWL-S processes. There is

no support for complex interactions or processes. Service descriptions are essentially limited to the provision of a single ("black box") step, showing initial inputs and final outputs but no intermediate steps. This essentially renders many of the features of OWL-S useless, making service descriptions small (but still well formed) and potentially unrepresentative of commercial needs.

This limitation is the greatest hindrance to the use the OWLS-TC (in its current form) as a sole source of test cases. Instead, it has been decided that the OWLS-TC will be added to a number of other sources of descriptions (Table 7.1) and augmented with more complex process models. An outline of the proposed techniques can be found in the following section.

Category	Count
Education	286
Medical Care	73
Food	34
Travel	197
Communication	59
Economy	395
Weapon	40

Table 7.2: OWL-S TC Categories

7.2.2 Automated Process Model Generation

As has been described in the proceeding sections, there is a limited supply of comprehensive OWL-S descriptions. Less than 1300 have been chosen as suitable for use in this thesis and of those, only 24 present process descriptions with complex actions. The majority of the OWL-S documents currently available only expose atomic processes in their definitions. Of additional concern is that only a small percentage of the descriptions found on the web are grounded in working services. These issues pose serious concerns as to how NPE is to be tested. Without a comprehensive and unbiased test case set the full and fair testing of NPE will be almost impossible. NPE has the ability to reason over complex processes, not just atomic ones. Its power lies in the capability that come with being able to handle even the most complex situations as easily as the most simple. Complex OWL-S processes will be inevitably needed for testing. Without this set of norms the solution requirement for input service test cases defined in Section 1.4.2 will not be met and the quesitons governing this thesis cannot be answered.

There are a few examples of complex OWL-S processes existing to date in the public domain. The vast majority of published OWL-S descriptions contain atomic processes. As has been stated before in this thesis, there is no wish to recreate or duplicate information if it is already available to use. To overcome the lack of complex process definitions, all

that is needed is a set of published processes and a method with which to compose these atomic processes into new complex process models. Each atomic process attached to a definition can be seen as a process on its own ready to use. If the pool of test cases available online is seen as instead a pool of available OWL-S processes for composition, then it is possible to compose new valid complex test cases from this pool. An automated process model generation technique has been proposed and will be implemented. This system takes as input descriptions from the sources in Table 7.1 and outputs new OWL-S process models.

The proposed system operates in the manner indicated in Figures 7.1, 7.2 and 7.3. It relies on an ability to load multiple OWL-S descriptions, isolate any process models and then extract any processes (atomic or otherwise). Once loaded, every process from every description can be compared. Processes are matched by I/O type and label. The output of each process is matched to the input of every other process in an attempt to discover which processes can serve as sources of information for others. Initially, this process will form pairs of processes which are matched as having compatible output (for the first) and input (for the second) types or labels. Matching is based both on semantic and textual data. Five matching filters have been proposed with varying degrees of importance (loosely equivalent to those outlined in Section 5.1.1.1). These filters are listed in Table 7.3. The semantic filter exact is the most strict. Exact requires that there is a one-to-one mapping between the type of the output and that of the input. This would enable any data created by one process to be sent without change to a second. Plug-in filters relax this constraint in that they allow the output type to be more specific that the input type (and thus require minimal conversion for the two processes to be linked). The subsume filter again relaxes this requirement in that it allows the input value to be more specific that the output. This reversal makes it harder to accurately link data types between processes but does enable more matches to be found. Text based matching does not rely on any semantic data at all, instead it looks for similar labels. Labels are compared using basic textual similarity filters (such as does one label contain the other), in an attempt to look for similar types of data. This metric is very much relaxed compared to the semantic filters above and it is likely that any matches would have to be applied manually to overcome the lack of semantic similarities. However, in an attempt to capture the intent of the original developer as to the uses for a process. Textual matching is also applied along with the strictest semantic filter "exact". This combination gives an extra level of matching which attempts to capture both semantic data information and the intention of the original developers. These techniques are an evolution of existing ones used in the OWL-S semantic matchmaker OWLS-MX (Klusch et al., 2008) and draw on research conducted while designing the NPE implementing module NPE-M in Chapter 5.

Running the matching algorithm produces a set of pairs of OWL-S processes. These can then be strung together to form flows of multiple processes. The task of appending

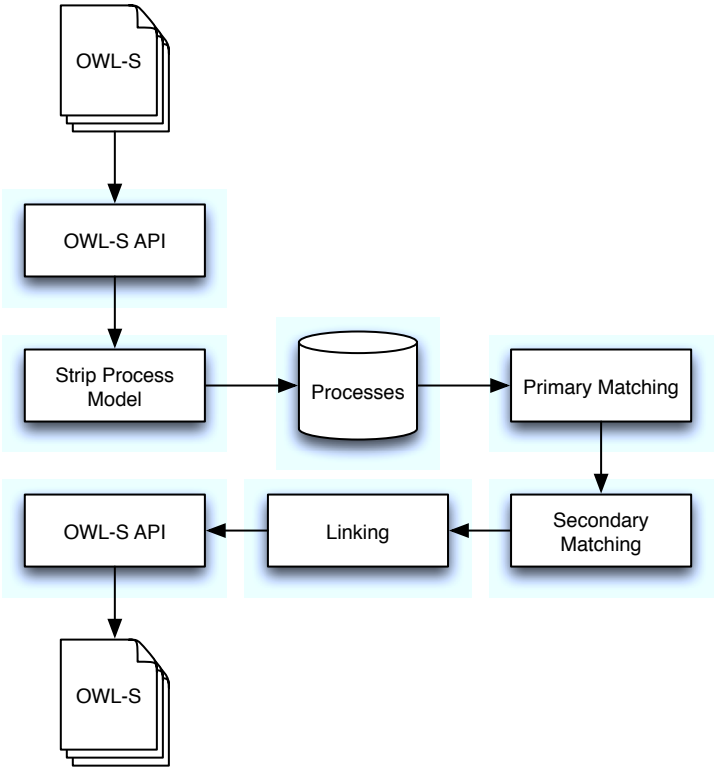

Figure 7.1: Automated Service Generation System Overview

Category	Importance
Exact + Text	5
Exact	4
Plug-In	3
Subsumes	2
Text	1

Table 7.3: Matching Categories

paired processes starts by linking pairs of processes where the output process is the same as the input process of another as described in Figure 7.3. This is a computationally trivial phase. Following this, the 48 processes defined as part of the NPE process library (Table 7.1) are added to the set of processes. Any complex flows are compressed to processes (with only the external I/O shown) and the matching phase is run again. A final joining phase is completed as the process models are all expended. This produces a list of potential models, some with relaxed or missing constraints (I/O parameters). As this is only a quick example system for use in creating test cases (and not the focus of this thesis), the final steps of process model creation will be carried out by hand. These steps are to take the recommendations produced by the system and create valid OWL-S descriptions from them. This process entails the creation of appropriate service and profile models, as well as adapting any processes to fit the complete model. This may

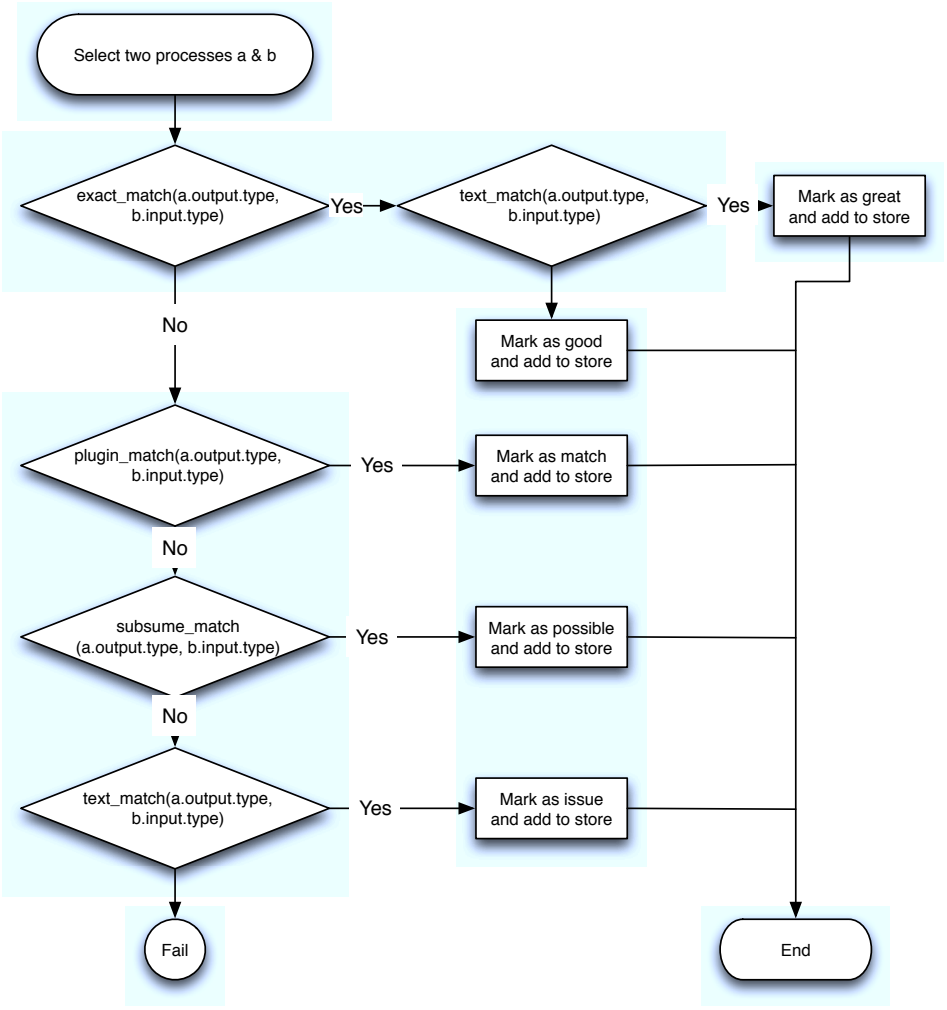

Figure 7.2: Automated Service Generation System Matching Overview

also include the addition of more complex control constructs where needed. It has been decided that these final steps will be carried out manually as this would be quicker than augmenting any previous algorithms. It is the intention that this will create a set of 10 to 30 complete process models which will be used to produce 15 to 21 complete OWL-S descriptions.

The technical design of the described process model generation system is outlined in Figure 7.1. It is essentially a straight forward OWL-S description loader with a filter based matching system. The system has been written to be uncomplicated and light. The intention was to create a system that would deliver the process models (and descriptions of these models in OWl-S) needed with minimal input from the developer. The system has been implemented in Java and is designed and executed in the Eclipse environment. Basic performance monitoring is handled by the Eclipse Memory Analyser and all results are passed to a basic log file. Support for OWL-S is handled by the OWL-S API (of Basel, 2010). This enables reading and writing of OWL-S descriptions as well knowledge base

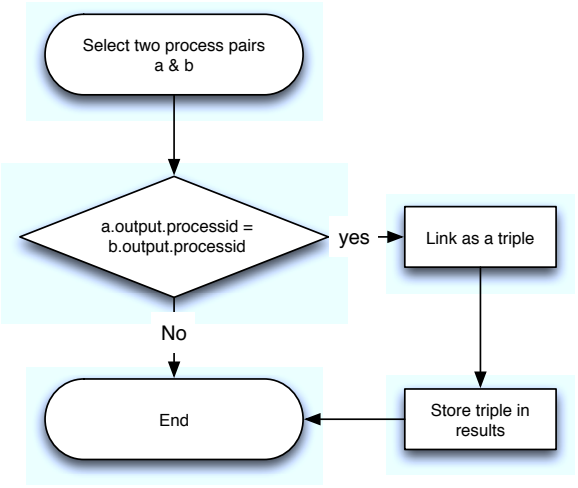

Figure 7.3: Automated Service Generation Linking Process

handling. Through the OWL-S API, Pellet is used as a reasoner and queries to the in memory knowledge base are handled using SPARQL.

7.2.2.1 Test Case Generation Results

The program outlined above was executed four times. Three times using sets of descriptions from a single domain and a final time using all available OWL-S files. The domains were taken from those of the OWLS-TC as per Table 7.2, augmented with descriptions from other sources (sourced manually into domains). Of the seven domains available, three were chosen for single runs. These were: education, economy and travel. Chosen because of their large size and they were a broad representation of the kind of scenarios which NPE might be expected to have to handle. The final execution evaluated over all the available OWL-S descriptions.

Each execution run was carried out on a dual core machine with 4Gb ram and a large amount of free disk space. Initial runs failed due to the Java virtual machine running out of memory (out of heap space). This was traced to an old version of the OWL-S API which was upgraded to the latest version from CVS (version 3.1). Following this upgrade, a number of small performance, tweaks were applied to the underlying code and the Java virtual machine was given access to a larger amount of memory. Subsequent executions completed without fail, although run-times were in excess of four hours.

After an initial run 731 matching pairs of processes were found. Divided by domains, this results in: 184 for education, 302 for economy and 84 for travel. The linked processes were related by function not just I/O types. A sample of the pairs created along with a manual decision on suitability can be found in Table 7.4. Upon running the second round of mapping, a number of more complex chains were produced. This was added to by

the inclusion of 48 new processes from the NPE process library. At this point, some 242 complex flows were present in the results set. This initial set included process models with matches which relied on subsumed and textual matching filters. Also included were process models which were semantically valid in terms of I/O (Exact or Plug-in matches) but invalid in terms of functionality.

To solve these problems the list of matches and constructed flows was inspected manually. A short list of 86 process models was created, from which 12 were selected. These were altered in minor ways to make them valid (altering types to make a subsumed match into an exact, for example). Finally, six were selected and split into the three categories outlined in Section 7 with two in each category.

In a straight forward step these six models were converted into OWL-S process models. These process models were then added to OWL-S descriptions. Each OWL-S description was created by hand with a new service and profile model written for each. To make this process quicker, information was used from the OWL-S descriptions of the atomic processes which composed the final process model (type declarations where the I/O types hadn't changed for example). A skeleton OWL-S template was also used to provide the initial imports and structure for each document. Each of the six finished descriptions was then checked using Protege and Pellet. As a further test description documents were loaded into the OWL-S API to test parsing (loading was carried out using the above model generator with error reporting on high and the matching cycles switched off). All six passed the tests and were assumed to be valid.

From	To	Suitable
SignInData	AcctID	Information Gathering
Genre	Film	Information Gathering
Location	Map	Service Provision
AcctID	Itinerary	Service Provision
SearchTerm	Item	Purchasing
PaymentDetails	Receipt	Purchasing

Table 7.4: Potential Mappings

7.2.3 Base Test Cases

For the three base categories, six base descriptions were initially been chosen; three primary descriptions (one for each) and three secondary. Six rather than three were initially chosen so that if the degradation of one was deemed to be insufficient, or be producing skewed outcomes, any results could be checked against a second set. The initial base cases can be found in Table 7.5. Full listings can be found in Appendix A.5. An explanation of the relevance and objectives of each base test case can be found in Section 7.4.

Name	Category
FindExpert	Information Gathering
FilmLocator	Information Gathering
BookMedicalAppointment	Service Provision
GenerateReportFromItinery	Service Provision
BuyGenericItem	Purchasing
BuyHouse	Purchasing

Table 7.5: Base Test Cases

7.3 Full Test Case Generation

At the start of this chapter, it was explained that the testing procedure for NPE required the presence of a set of original test cases. Six such test cases were produced using the procedures outlined above. These six descriptions represent the basis of the NPE testing system. A great deal of time and effort was put into ensuring that these were able to test NPE in a fair and unbiased manner. It was important to ensure that the base descriptions were as correct as possible as all the other test cases used are derived from them. From these initial descriptions, a set of degraded descriptions was created. For each of the six base test cases four derived test cases, were produced, with increasing levels of reliability. The following section outlines the techniques used to create these additional test cases.

At the very heart of the testing strategy outlined in Section 9.3 was the need to demonstrate to what extent NPE can differentiate between similar services based on the process model's exposed in any OWL-S description. For this aim to be met, sets of similar test cases needed to be produced. Six base cases were produced describing services which if chosen would be the "best" choice. For each of these cases, four derived descriptions needed to be created. To produce these derivations, it was decided that an automated approach with some degree of random choice should be applied. This was primarily to avoid adding undue bias to any derived descriptions (by unconsciously introducing errors which could be easily identified by NPE), but also to speed up the creation of any descriptions (as 24 need to be made). The technique used to created these derived descriptions was derived from mutation testing. It was envisaged that by utilising a static base of mutations and applying them randomly to the base test case descriptions, new descriptions could be quickly produced. The processes behind mutation testing and a description of how they were used in the context of producing new test cases follows.

7.3.1 Mutation Testing

The technique used to produce test cases outlined in this section was based on the idea of mutation testing. Mutation testing techniques originated in the 1970's as one of a

number of fault-bases testing routines. Its primary use is for evaluation of test adequacy (DeMillo et al., 1978; Acree et al., 1979). Original mutation testing was carried out using "white box" techniques. Based on the notion that no programmer is 100 percent accurate, mutation testing altered the original program code to generate mutant copies which could then be compiled and tested. To speed up testing the process was carried out automatically, even so it was seen as a long and costly process which took up large amounts of time and resources. The quality of a test is based on its ability to detect faults. Mutations will not always lead to valid program executions; those that cause the program to fail or give different outputs are said to have been "killed". The mutation score of the test is the ratio between killed and non-equivalent mutants. One of the biggest issues with mutant testing is the detection of equivalent mutants. These are mutants where-by the mutations applied to the original code produce mathematically (or logically) equivalent test results and thus are unable to be killed. Detecting equivalent mutants is difficult even in small programs and only adds to the cost of running a test.

The high cost of white box source code based mutation testing has led to a recent interest into black box techniques. This, coupled with a rapid growth of the use of non-procedural languages such as XML has led to the development of mutation testing in the form of interface (Delamaro et al., 2001) and specification mutation tests (Wuzhi et al., 2005; Murnane and Reed, 2001). Tests have also been carried out into mutating syntactic software descriptions (Offutt et al., 2006) and service descriptions (Wang et al., 2009; Lee et al., 2008b). The latter are of especial interest to this work as they look at the possibility that mutation testing techniques could be used to test systems relying on OWL-S descriptions.

7.3.2 Mutating OWL-S Processes

Mutation techniques based on those found above were used to create new OWL-S descriptions by degrading base test cases detailed in Section 7.2.3. Using mutation techniques, one description can be mutated multiple times to form multiple unique test cases. This enabled all of the relevant service test cases for this thesis to be created much faster than manual methods.

The mutation process is relatively simple. A single base test case is loaded into the mutation system. Once processed (and providing it is valid), mutations are randomly applied to the descriptions process model. Initially, mutations are based on swapping individual or groups of processes. To try and cut down the number of killed mutants created (the intention of using mutation techniques is not to test OWL-S but to create new descriptions, thus this work is not particularly interested in the number of killed mutants) processes were swapped in a manner that ensured the I/O of the process model would not change and the internal flows of data would always be secured. Mutations were based on a set of OWL-S processes stored within a database. Mutations could be

applied in one of five ways: a one to one swap of processes (with the same I/O), with the addition of a new process (or processes), removing a collection of processes and replacing with a single one, by removing single processes, or removing groups of processes.

Following mutation, the new process model was then checked for validity before being inserted back into the OWL-S description. This was then loaded into an execution engine and executed for validation. If the description passed this test then it was checked against the original test case. If the new description failed either of the first two tests it was deemed to be invalid or killed and is discarded (and recorded as killed). If the last test is failed then the description is discarded but not killed (recorded as similar).

The mutation process was rerun several times on the same base test case. A pool of mutated descriptions was produced. These are "first level" mutations. These were then loaded back into the mutation system (one by one). The results of this second round of mutations are "second level" mutations. This process was carried out a further two times to form four levels of mutations. One description was then manually chosen from each level to form part of the NPE test case set.

7.3.2.1 Scoring

The NPE testing system required that test cases be grouped into sets of five related services: a single original service description followed by four closely related services or declining "reliability". To make this easier, each mutation was tagged with a score. The more unwanted mutations the higher the score. These scores were stored in a separate database which was used to associate scores with descriptions during testing. It has been the intention of this thesis to discover if a system equipped with NPE can differentiate between services based on differences in process models. If NPE was working as hypothesised then it should be able to choose descriptions with lower scores (without knowing those scores).

7.3.3 Proposed Mutations

For mutation testing to be possible a set of mutations had to created. For the NPE testing routine, this set of mutations was static and described at design time. Mutations stored in this set were essentially a tuple consisting of an OWL-S process (or processes) and some keywords. These keywords come as a set of none, one or many and were used to identify the types of processes this mutation would effect. As has been described in the preceding sections, mutant descriptions were created by substituting OWL-S processes in the original for ones provided by the mutation system.

The mutations proposed by this thesis as being suitable are mainly interested in altering the flow of data between participants. This may include one partner attempting to

extract data from another, or data being sent in an incorrect order. This can also include the altering of process models in a manner which might suggest to an implementing actor that the associated service might not be able to provide the results expected.

The mutation system stores a list of possible mutations as tuples in a database. This database was created to enable the easy sorting of mutations by both keywords and I/O requirements. This is essential as the first test a mutation must meet if it is to be applied is that the I/O requirements of the mutation match those of the process in the original description. Once any possible mutation had been filtered by I/O then they can be filtered by keyword. This filter was not so strong and created a number of killed mutants. A database of 64 mutations was created, as a selection of which can be seen in Table 7.6.

I/O	Keywords
[orderwithcard, storecard]	[order, cardno, expiry, accno]
[shippingdetails, billingdetails]	[address, street, name]
[searchforex, searchforexarchive]	[currency, amount]
[searchgoogle, search]	[searchstring]
[login, register, storetext]	[username, name, password]
[locateaddress, billingaddress]	[address, street, name]

Table 7.6: Proposed Mutations

7.3.4 The Mutation System

The mutation system was designed to be straight forward. As with the process model generation system in Section 7.2.2 this process relies on the OWL-S API to provide handling of OWL-S descriptions. This simplifies the initial steps of the mutation process as well as the final output phases. Allowing for more time to be spent on getting the central mutation algorithms correct. Reasoning is handled by Pellet, the majority of which centres around discerning type matches in the I/O of process. The mutation database holds potential sets of mutations using a MySQL relational database model.

The basic mutation system employed is outlined in Figure 7.4. The initial operations were similar to those of the process model generator. A description was loaded using the OWL-S API and the process model was extracted. The processes from the process model were extracted and stored as separates and also as groups (ordered sub-parts of the original model). A matching algorithm derived from that used in the process model generation system was employed to match processes to mutations. The matching process used all of the semantic I/O approaches listed in Table 7.3 but avoid the textual matching criteria. This was to avoid any mutations which might mean that the I/O of the substituted process would not match those of the preceding and proceeding processes

in the original model. Mutations occur one at a time and then are recorded. Many mutations occurred on one run. Results were buffered until all mutations were complete. Once this had been achieved then each of the mutants was checked for validity.

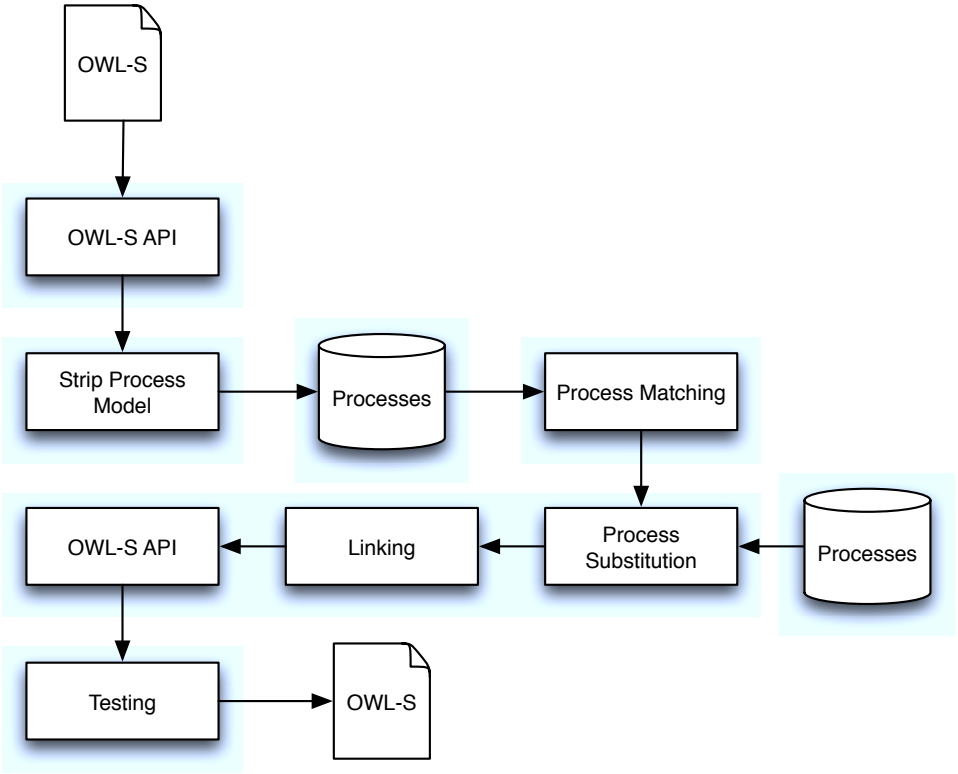

Figure 7.4: Automated System Generation Mutation System

Checking occurs after the mutation phase and starts with the mutant process being reinserted into the original OWL-S description. This was carried out programmatically through insertion into the knowledge base created at the start by the OWL-S API. Any changes which were required in the profile model were undertaken at this point as well. The description was then validated and executed using the standard OWL-S methods. If a mutant failed either of these tests it was discarded and the error message along with a record of the mutants details was logged. Any descriptions which passed these tests were exported as XML based OWL-S for manual selection.

7.3.4.1 Error Rates

As with any mutation testing based approach, a large number of errors are detected at runtime. These manifest themselves as killed mutants, in this example; invalid OWL-S documents which are either incorrectly linked or expose infeasible process flows. These errors are recorded as plain text in logs for analysis after runtime.

This thesis is not interested in the analysis of any error logs. Errors were captured during runtime for bug checking purposes and to ensure a full and fair testing process. However, the contents of these logs was not important when compared to the resulting mutants. The results of these logs were parsed and are displayed in Table 7.7. The errors displayed are those of mutants killed and duplicate mutants.

The mutation system was created not to test OWL-S but to generate as many valid mutant descriptions from one input description as possible. As such the error rates are skewed compared to those usually associated with mutation testing. Filters limiting which mutations can be applied have reduced the number of killed mutants significantly. Of more interest to the authors was the number of duplicate mutants. Duplicate mutants are of more concern as the testing process was intended to produce multiple descriptions different from the original. If the output descriptions are duplicates of the originals then obviously the process had failed. The rates of duplicate mutants displayed are within the range expected. The output descriptions were found to be of adequate quality to be of use after the verification phase. The error rates shown here demonstrate that the design of the system was of a high enough standard to produce minimal killed mutants and mostly useful descriptions.

Error	Count	Notes
Pass	136	Passes are mutants where the flow of data within the process model is valid.
Duplicates	97	Duplicates have the same flows of data within created process models.
Fail	283	A service fails if the flow of data within the process model is invalid.

Table 7.7: Mutant Error Rates

7.3.4.2 Results

The result of the mutation process was a set of eight new degraded descriptions for each initial base description. The process took slightly longer than initially expected. However, when the caching of new descriptions (during the mutation phase) was switched off the time to run increased significantly. The final architecture produced results in an expected and timely manner and was of acceptable quality. Initial attempts to speed up processing by controlling the mutations to cut down on errors was also successful as is shown by the relatively high ratio of killed mutants to valid descriptions.

It was intended that it may be possible to have a fully automated process which could produce four new descriptions of declining quality having been given one initial description.

However, this was shown to be impossible due to the efforts needed to sort through the many valid mutants produced after a single run. This process was far from trivial and so for speed was carried out manually. The manual aspect of final test case creation made the automated production of test cases at runtime (during the NPE testing cycle) impossible. However, it was possible to produce a large test set of descriptions using the above methods which were then used at runtime in a random fashion.

The final result of the mutation process was four new descriptions for every base description. A well as this an extra 12 descriptions were produced for every base description to facilitate the automated selection of random test case sets during testing runtime.

7.4 NPE Test Cases

A full set of test cases was produced. This stands as an initial set of 30 with 72 more which could have been used as necessary. The following sections outline the relevance of each initial base test case, focussing on the relation each category has to testing NPE and how different descriptions tested different areas. Following on from this, a look at the most common mutations aimed to demonstrate the ability of NPE's to detect these variations.

Out of numerous potential areas of service use identified by this work, three key domains were chosen. The reasons for the selection of these domains has been covered by this thesis in Section 7. Categorising services into defined domains is potentially difficult as any one service may possess characteristics which span several domains. This thesis has defined three categorise and placed test cases into each based on no metrics other than apparent best fit. The following sections look at these categories in turn, evaluating the test cases generated for each and provides context and facilitate an understanding of each test case. Domains and test cases are evaluated in the following three subsections in no specific order.

7.4.1 Information Gathering

This thesis defines information gathering as the process by which a requestor can obtain information from an information provider. The only limitations on this domain are that information is provided to the requestor, no interest is paid to transactions or payments for information and there are no requirements on security.

Information gathering was the first of the potential domains identified by this work to be chosen as suitable for use in testing NPE. The choice was primarily motivated by the ready availability of information gathering web services in use on the Internet today. Examples of which include price and description lookup services such as those provide by

Amazon.com and eBay. Services currently in use in this domain are not limited to those provided by retailers looking to allow product data to be integrated into third-party applications. There is also a growing number of information providers which are using web services to enable data reuse by others. These applications include weather data, geographical data (including postcode and address locating), currency data and financial information. Of all the domains identified by research, information gathering would seem to be the most prosperous; most probably because rather than requiring new data to be produced, services within this domain instead tend to place new views on existing data.

Services which fall under the domain of information gathering are typically characterised by having larger amounts of outgoing data than input data (although this is not always true). Data obtained from an information provider tends to be highly structured so as to reduce its footprint and/or ensure that clarity is preserved. In many instances, the input data for an information gathering exchange is either unstructured (such as a free text search term) or comprises of a very short identifier (for example, an area code or geographic location). Often very little filtering is carried out by the provider, thereby enabling the requesting actor to obtain the maximum amount of information.

Many of the services within this domain fall under the sub-categories of search and look up. That is to say that they are characterised by minimal steps (only one or two) where a requester sends a request to a provider along with a number of limiting filters and receives information in response. Such services include those offered by Google for web search[1] and Amazon for[2] product search. These services are very straightforward, requiring only WSDL groupings as they contain no complex process models to enable completion. Services such as these have been discounted from this work. Although they make up a significant proportion of the services currently available, the authors believed that their simplicity would not aid in the testing of NPE. Determining the suitability of a service providing a process model with two or fewer processes is important, however, it was likely that existing profile based techniques (Section 2.3) would provide just as good results as NPE.

For the purposes of testing NPE longer, more complex services were chosen for the information gathering domain. Here rather than just gathering a potentially large set of information from a provider, clients have the ability to sort through data and ask multiple requests to focus the end results. The information gathered may also be increasingly complex and answer multiple questions during the processing of the simulated process model.

To test NPE, two final service examples were chosen: FindExpert and FilmLocator. These two were chosen as they represented a good cross section of the whole domain. They also were chosen as they decompose into separate processes, many of which could

[1] http://http://code.google.com/apis/soapsearch/
[2] http://aws.amazon.com

be seen to be information gathering in their own right. Both are centred on search and aim to enable clients easier access to required information. These two were also chosen as they contain a wide set of OWL-S operations and so should offer numerous avenues to test NPE. What follows is a brief description and analyse of these two test cases.

7.4.1.1 NPE Test Case: FindExpert

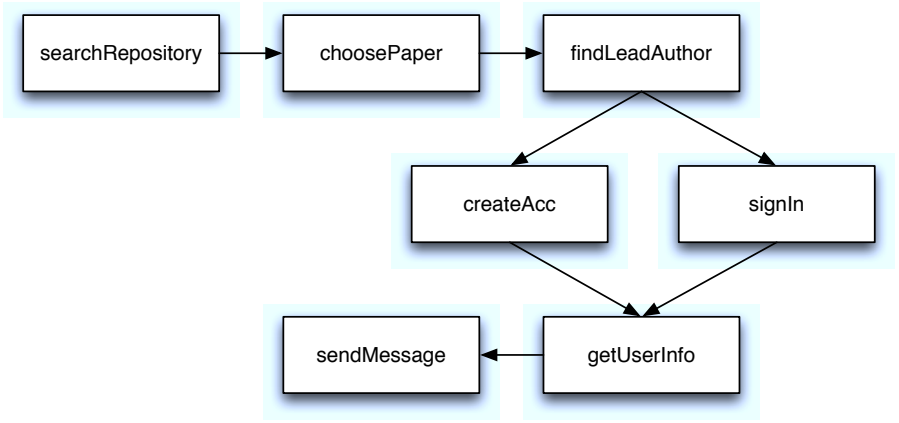

Figure 7.5: Process Model For FindExpert

FindExpert is based on a search routine (Figure 7.5). The initial test case definition was to simulate the searching of a repository of academic papers. This is a straightforward example but to which the author is accustomed. It is worth knowing that this example can apply to the searching any static repository of information. A repository within the academic domain was only chosen for ease of use.

The service is enacted with a single term providing the input for a search process. This single first step search would be enough to form a search process which on its own would be a useful tool. However, this example provides a multiple step search, where numerous searches are provided along with other processes to enable the formation of complex process models. Advanced control constructs are provided by OWL-S including choices. This enabled both the testing of NPE's handling of complex controls and numerous OWL-S constructs to be tested.

To best test any NPE based system, Find Author was been produced with several "real world" elements. At the most basic level, this is represented by the inclusion of an initial search taking a tree text input and returning a list of responses. Additionally a number of more complex constructs were added including security constraints based on a login or sign-up process and the ability to send out data to a third party at the end of the iteration. This last feature does not technically fall within the domain of information gathering, however, it was allowed having searched for a user (and it was often produced as a matched pair of processes during test case production).

The OWL-S document detailing FindExpert can be found in Appendix A.5.1.

OWL-S Use Case FindExpert enables a requester (client) to find a relevant expert in a field and message them. This service is based on searching a repository of academic papers to find experts. Fields of interest are initially specified by the requester by using text-based search terms. An expert is defined as the lead author on the paper, which has the highest citation count out of the papers returned following a keyword search. Having obtained an authors details, a requester is then invited to login or created an account before being able to message the author.

The following use case was defined:

1. FindExpert starts with a free text search of the repository, this returns a list of papers with matching keyword terms.
2. The client then sends a list of papers to ChoosePaper which uses the number of times each paper has been cited to rank the list. This then returns the highest ranking paper.
3. To obtain the lead author for a paper the client then sends the chosen paper to FindLeadAuthor which returns an author.

4a. If the client has an account they now send the account details to SignIn and receives an account id following a successful login.

4b. If the client does not have an account they may now send some account details to CreateAcc and will receive an account ID in return.

5. The client may now request the account ID for the previously obtained author.
6. Finally, the client can send a message to the author using the provided account ID. A confirm will received on sending of the message.

No alternative or invalid flows were included for this use case, as NPE does not currently fully support OWL-S conditions and effects.

7.4.1.2 NPE Test Case: FilmLocator

FilmLocator is a shorter and comparatively simpler test case than FindExpert (Figure 7.6). There are both fewer steps and simpler control constructs. This was chosen purposefully to act concurrently with FindExpert as a simpler test case as has been discussed previously in this chapter, to enable the fullest possible testing of NPE, pairs of similar test cases with differing complexities are required for each domain.

Process	Type	Inputs	Outputs
FindExpert	Composite	#SearchTerm #AcademicPaperList #AcademicPaper #AcctInfo #SignInData #Author #Message	#AcctID #MsgSendConfirmation
FindExpertFull-SearchRepository	Composite	#SearchTerm	#Author
SignInAlternatives	Composite	#AcctInfo #SignInData	#AcctID
SignInSequence	Composite	#SignInData	#AcctID
CreateAcctSequence	Composite	#AcctInfo	#AcctID
SearchRepository	Atomic	#SearchTerm	#AcademicPaperList
ChoosePaper	Atomic	#AcademicPaperList	#AcademicPaper
FindLeadAuthor	Atomic	#AcademicPaper	#Author
SignIn	Atomic	#SignInData	
CreateAcct	Atomic	#AcctInfo	#AcctID
LoadUserProfile	Atomic		
GetUserInfo	Atomic	#Author	#AcctID
SendMessage	Atomic	#Message #AcctID	#MsgSendConfirmation

Table 7.8: FindExpert Processes

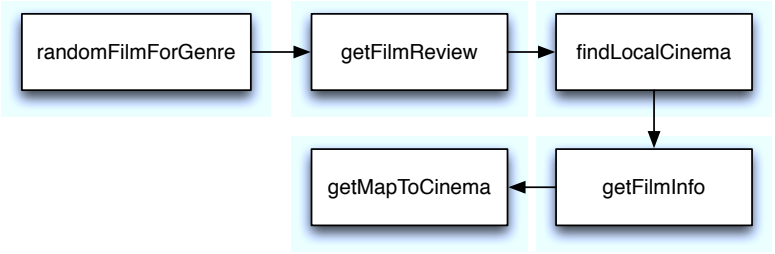

Figure 7.6: Process Model For FilmLocator

As with FindExpert, FilmLocator is again heavily search focused. Having two search focused examples in the information gathering domain would seem to be counter-productive. However, this situation was chosen as search is heavily favoured in information gathering and a large part of many business workflows. This example also differs from the previous in that it only accepts heavily structured inputs. For the initial search term, this comes in the form of a genre, as defined by the associated concepts ontology (Appendix A.5.7). There are fewer filters than FindExpert has, however, these filters are well structured and tend to take smaller amounts of data. To further test NPE's handling of different situations, a number of different examples of service use are included. This includes the use of geo-location data when looking for a nearby business. FilmLocator unlike

Process	Use Case
FindExpert	The containing composite process encompassing all other processes.
FindExpertFull-SearchRepository	A composite process encompassing: SearchRepository, ChoosePaper and FindLeadAuthor.
SignInAlternatives	A composite process encompassing: SignInSequence and CreateAcctSequence.
SignInSequence	A composite process encompassing: SignIn and LoadUser-Profile
CreateAcctSequence	A composite process encompassing: CreateAcct and Load-UserProfile
SearchRepository	Enables a client to search through the repository for papers which match any of the provided search terms. Returns a list of matched papers.
ChoosePaper	Given a list of papers, returns the one which is cited the most.
FindLeadAuthor	Returns the lead author for a given paper.
SignIn	Signs a user in using the provided credentials and returns an account ID.
CreateAcct	Creates an account based on the provided credentials. Returns an account ID.
LoadUserProfile	A stub for loading a users profile.
GetUserInfo	Returns the account ID for a given author.
SendMessage	Sends a message to the given account ID. Returns a confirmation of sending.

Table 7.9: FindExpert Use Cases

FindExpert (which deviates from the information gathering domain in its final step) is a pure information gathering service. For a relatively small amount of highly structured input data, a relatively large amount or structured data is returned.

The OWL-S document detailing FilmLocator can be found in Appendix A.5.2.

OWL-S Use Case FilmLocator enables a requester (client) to search for a film and cinema. Initial film searches return a single random film for a given genre. Clients can then obtain a review for the film. To simplify this process model, there are no iterations. As such the client cannot request a new random film and review after the first. Once a film has been obtained, the client can pass some location data to the service and find the nearest cinema. Finally, information about showtimes and a map to the cinema are returned. As this is an information gathering service, no options are given to purchase or reserve tickets.

The following use case was defined:

1. FilmLocator starts with the client sending a genre to the service. The service then returns a random film within this genre. It is to be accepted that the service chooses from a list of films on general release.

2. Having been suggested a random film the client can then obtain a review for the film.

3. Next the client provides the service with location data. The service searches for the nearest cinema to the given location and returns information about it.

4. Having found a cinema and film, the client next looks up showtimes and other film information.

5. Finally, the client passes the cinema to the service and is returned a map for the cinema.

No alternative or invalid flows were included for this use case, as NPE does not currently fully support OWL-S conditions and effects.

Process	Type	Inputs	Outputs
FilmLocator	Composite	#Genre #Film #Location #Cinema	#FilmReview #FilmInfo #Map
FilmFinder	Composite	#Genre	#FilmReview
RandomFilmForGenre	Atomic	#Genre	#Film
GetFilmReview	Atomic	#Film	#FilmReview
FindLocalCinema	Atomic	#Location	#Cinema
GetFilmInformation	Atomic	#Cinema #Film	#FilmInfo
GetCinemaInfo	Atomic	#Cinema	#Map

Table 7.10: FilmLocator Processes

Process	Use Case
FilmLocator	A composite process encompassing all other processes.
FilmFinder	A composite process encompassing; RandomFilmForGenre and GetFilmReview
RandomFilmForGenre	Returns a random film with a given genre from a list of current showing films.
GetFilmReview	Returns a film review for a given film.
FindLocalCinema	Returns the cinema nearest the given location.
GetFilmInformation	Returns showtimes (and other information) for the given film in the given cinema.
GetCinemaInfo	Returns a map to the given cinema.

Table 7.11: FilmLocator Use Cases

7.4.2 Service Provision

Service provision services aid actors by providing basic or complex abilities. It should be noted that a first glance the steps involved are very similar to those present in the previous information gathering examples. The primary difference between an information gathering example and a service provision example is intent. That is to say that the intention of an information gathering service is to primarily provide information in response to a request, whereas the purpose of a service provision service is to effect change based on a received response. This change may involve altering or adding to the data received, or it may involve elements outside of the service. It is these minute differences that NPE picks up on and which agents can use to differentiate services.

Service provision as a domain covers a great number of potential uses. As has been touched on above, service users could even class information gathering as a form of service provision. Indeed many service provision process models start with search phases identical used in information gathering. However, as has been mention in the previous paragraph, the differences between information gathering and service provision often lay in the intent and final outcomes of any interaction. The two test cases presented below demonstrate service provision from the view of searching for and reserving an appointment and aggregating and verifying data. These are just two of the many uses for service provision, which may include among others: information manipulation, storage, verification and processing.

7.4.2.1 NPE Test Case: BookMedicalAppointment

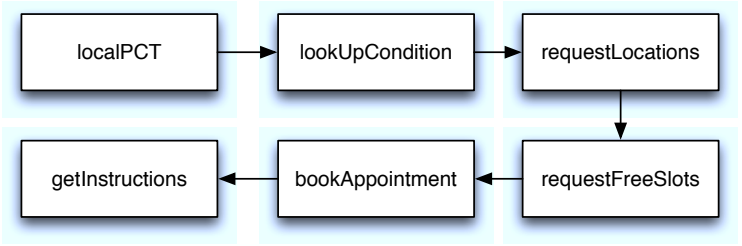

Figure 7.7: Process Model For BookMedicalAppointment

BookMedicalAppointment highlights the characteristics of a service provision example (Figure 7.7). This case shares many similarities with those in the information gathering domain. It is essentially a six stage search, enabling a requester to find information about medical appointments based on a number of searches. This domain crossover is highlighted due to the similar control structure, based on six processes and no complex OWL-S constructs, as mostly shared with FilmLocator. One small difference is the

relatively balanced amount of input and output data shared between actors in comparison to the large amount of output data received from an information gathering service.

One notable issue with this example is how the service copes with the presence of potentially sensitive data. Almost all of the data transferred here could be seen as personally sensitive and should be treated with care. The order in which data is collected and processed will be of extra interest to an NPE based system involved in evaluating the suitability of any service handling sensitive information. Processing sensitive data in an unusual order, or asking for information other than that which might usually be expected, could be a good indication that a service is malevolent or proving an unreliable service.

The main difference separating this test case from those in the information gathering domain is that of the final effect of the service. In this case, BookMedicalAppointment does provide a large amount of information in the form of instructions for the medical appointment. However, it is set apart from the information gathering domain by the effect it has on the wider world. Following the correct execution of the BookMedicalAppointment process model not only will the client have obtained information about attending an appointment, they will have also booked a medical appointment. This step to book an appointment sets this service apart from those in the information gathering domain.

The OWL-S document detailing BookMedicalAppointment can be found in Appendix A.5.3.

OWL-S Use Case BookMedicalAppointment enables a requester (client) to search for an appropriate medical location and book an appointment. This is a relatively straightforward process which allows the client to first search for a Primary Care Trust (PCT) near a provided location. The client next searches for a medical condition based on a set of search terms (keywords). This, along with information about a PCT, is used to find the nearest medical service location (department) suitable for treating the ailment. Once a department has been found a query for free appointment slots in submitted before the client selects the medical service location and makes a booking. It is worth noting that as this is a basic process model, no provisions were made for the use of blocks or flags to prevent free appointments being booked whilst a client is deciding. Finally appointment information, such as how to get to the location, are requested by the client.

The following use case was defined:

1. BookMedicalAppointment starts with a query using location data for the nearest PCT.

2. A query is then submitted looking up the clients ailment using any number of keywords as search terms.

3. The client then requests from the PCT a medical department (within a medical location) at which treatment for the potential condition can be sought.
4. Having found a medical department, the client requests all free appointments.
5. The client selects an appointment slot and books it with the medical service provider.
6. Once the appointment has been made the client requests appointment information from the hospital department.

No alternative or invalid flows were included for this use case, as NPE does not currently fully support OWL-S conditions and effects.

Process	Type	Inputs	Outputs
BookMedical-Appointment	Composite	#Location #SearchTerm #AppointmentTime	#AppointmentTimeList #MedicalAppointment #AppointmentGuide
FindCareSequence	Composite	#SearchTerm #PCT	#CareOrganisation
LocalPCT	Atomic	#Location	#PCT
LookUpCondition	Atomic	#SearchTerm	#MedicalCondition
RequestLocations	Atomic	#PCT #MedicalCondition	#CareOrganisation
RequestFreeSlots	Atomic	#HospitalDepartment	#AppointmentTimeList
BookAppointment	Atomic	#AppointmentTime #HospitalDepartment	#MedicalAppointment
GetInstructions	Atomic	#CareOrganization	#AppointmentGuide

Table 7.12: BookMedicalAppointment Processes

7.4.2.2 NPE Test Case: GenerateReportFromItinery

As a second more complex example of a service provision process model, GenerateReportForItinery provides for the testing of NPE's handling of some of the more complex OWL-S control constructs (Figure 7.8). However, the service it provides is still a relatively straightforward one. The aim of GenerateReportForItinery was to enable the creation of an itinerary, which can then be verified and a report generated as a summary. This test case builds on the perennial "travel agent" example which is so often used to test agent and Semantic web-based systems. In this case, the service provided aims to replicate the verification of any travel plans (perhaps to ensure that no clashes in timing occur) and the production of a summary. For an example of such a service think Trip-It[3].

This test case was purposefully more complex than that of BookMedicalAppointment. It is also focussed more on service provision rather than information gathering. It has both

[3]http://www.tripit.com

Process	Use Case
BookMedical-Appointment	A composite process encompassing all other processes.
FindCareSequence	A composite process encompassing; LookUpCondition and RequestFreeSlots
LocalPCT	Given a location, returns the PCT which governs the medical centres near that location.
LookUpCondition	Given a set of symptoms this service returns a possible condition. This is done on a best effort basis and is only meant to be as a guide to enable the selection of a specialist department. This is not intended as a diagnosis.
RequestLocations	Returns a medial location (department, hospital, clinic) suitable for the treatment of the given medical condition. Only locations within the jurisdiction of the given PCT are selected.
RequestFreeSlots	Returns a list of free appointment slots for a given medical location.
BookAppointment	Books an appointment slot at the selected department.
GetInstructions	Returns instructions for appointments at the selected department.

Table 7.13: BookMedicalAppointment Use Cases

more processes and a more complex flow between processes. This can be seen from the start as the first processes force the user to create a login account or use an existing one. Security constructs such as user accounts are common within these test cases as well as on many other workflows outside of this thesis. They force the user to provide a level of identification which both protects the user and the service provider. Having logged in and created an itinerary the client is then asked to place travel items such as flights and hotels into the itinerary. This step tests some of the most complex control constructs OWL-S has by linking a choice with an unlimited iteration. This enables the client to continue adding as many or a few holiday items as is needed. Following this step the rest of the model is relatively simple, only made more complex by two rather than one major output phases (allowing for the result of verification to be returned then the user separately from the end report).

The OWL-S document detailing GenerateReportFroItinery can be found in Appendix A.5.4.

OWL-S Use Case GenerateReportFromItinery provides a set of tools to create an itinerary and then validate the contents and produce a report. The use case for this process is one for a requester (client) who has already booked a holiday and now wishes to produce a personalised holiday itinerary. This itinerary can then be validated and retrieved in the form of a report. Verification aims to ensure that the itinerary is valid in

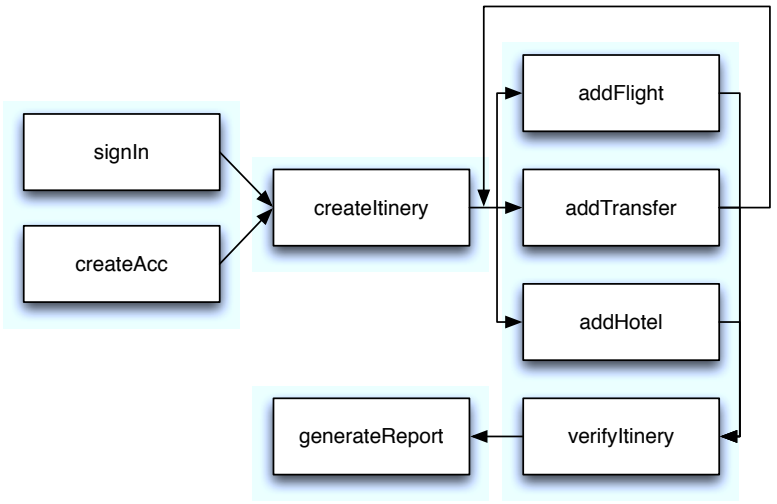

Figure 7.8: Process Model For GenerateReportFromItinery

the sense that no two elements overlap and that the client is in the correct location for each item. An itinerary is produced by repeatedly adding holiday items, which can be in the form of: flights, transfers, or hotels. To ensure that any personal details are kept secure clients must first login or create an account.

The following use case was defined:

1a. If the client has an account they must first send the account details to SignIn. The client receives an account ID following a successful login.

1b. If the client does not have an account they first send some account details to CreateAcc and will receive an account ID in return.

2. The client then requests the creation of a new itinerary.

3a. The client may now add a flight to the itinerary.

3b. The client may now add a transfer to the itinerary.

3c. The client may now add a hotel to the itinerary.

4. The client keeps on returning to step 4 until all available travel items have been added.

5. The client next applies for the itinerary to be validated.

6. Finally, a report based on the itinerary is generated by the service and returned to the user.

No alternative or invalid flows were included for this use case, as NPE does not currently fully support OWL-S conditions and effects.

Process	Type	Inputs	Outputs
GenerateReport-ForItinery	Composite	#AcctInfo #SignInData #Flight #Transfer #Hotel	#AcctID #ItineraryReport
TravelItemIteration	Composite	#TravelItinery #Flight #Transfer #Hotel	#TravelItinery
AddTravelItem	Composite	#TravelItinery #Flight #Transfer #Hotel	#TravelItinery
SignInAlternatives	Composite	#AcctInfo #SignInData	#AcctID
SignInSequence	Composite	#SignInData	#AcctID
CreateAcctSequence	Composite	#AcctInfo	#AcctID
CreateAcct	Atomic	#AcctInfo	#AcctID
SignIn	Atomic	#SignInData	
GenerateReport	Atomic	#VerificationReport	#VerificationReport
LoadUserProfile	Atomic		
VerifyItinery	Atomic	#TravelItinery	#VerificationReport
AddHotel	Atomic	#Hotel	#TravelItinery
AddTransfer	Atomic	#Transfer	#TravelItinery
AddFlight	Atomic	#Flight	#TravelItinery
CreateItinery	Atomic	#AcctID	#TravelItinery

Table 7.14: GenerateReportFromItinery Processes

7.4.3 Purchasing

After information gathering, in the opinion of the authors, purchasing is, the second most visible domain of services. Computer-based retail is booming on the Web, so it was only sensible that services were produced to enable clients to buy goods through data-centric views. The two test cases outlined here enabled the purchasing of two very different items in two different ways. This was undertaken to enable the fuller testing of NPE.

Many purchasing flows operate in very similar patterns. Usually, a transaction is conducted using three steps: finding a product, entering payment details and receiving a receipt. Intermediary steps may change and steps such as a search for a product may be added, but central to the core of many purchasing workflows are these three steps. The ability to differentiate between purchasing flows which have only a few variations is one of the key requirements of NPE. As such it was expected that more test cases would be added during testing.

Process	Use Case
GenerateReport-ForItinery	A composite process encompassing all other processes.
TravelItemIteration	A composite process encompassing AddTravelItem and adding an iterative loop.
AddTravelItem	A composite process encompassing: AddHotel, AddTransfer and AddFlight.
SignInAlternatives	A composite process encompassing: SignInSequence and CreateAcctSequence.
SignInSequence	A composite process encompassing: SignIn and LoadUserProfile
CreateAcctSequence	A composite process encompassing: CreateAcct and LoadUserProfile
CreateAcct	Creates an account based on the provided credentials. Returns an account ID.
SignIn	Signs a user in using the provided credentials and returns an account ID.
GenerateReport	Adds a full report for the itinerary associated with the given verification report.
LoadUserProfile	A stub for loading a users profile.
VerifyItinery	Verifies the provided itinerary and returns a verification report.
AddHotel	Adds the provided hotel item to the given itinerary.
AddTransfer	Adds the provided transfer item to the given itinerary.
AddFlight	Adds the provided flight item to the given itinerary.
CreateItinery	Generates a new blank itinerary and associates it to the provided account ID.

Table 7.15: GenerateReportFromItinery Use Cases

The two processes detailed below both accept data from the client enabling the identification of an item for sale. What is more, they also both accept payment details from the client and return a validation for payment. As has been mentioned, these steps were key. Also important is the ability to handle lists of items if searching for an individual. Critical to the safety of an interaction involving any exchange of sensitive data (such as payment information) is security and privacy. The following test cases handled these issues in different ways, ensuring they must always be addressed.

7.4.3.1 NPE Test Case: BuyGenericItem

BuyGenericExample is an example based on purchasing from an online shop (Figure 7.9). The processes involved are loosely representational of many online outlets. As has been discussed above, there are a number of key processes that the majority of purchasing flows posses. This example features all three. It is worth noting that this example is based

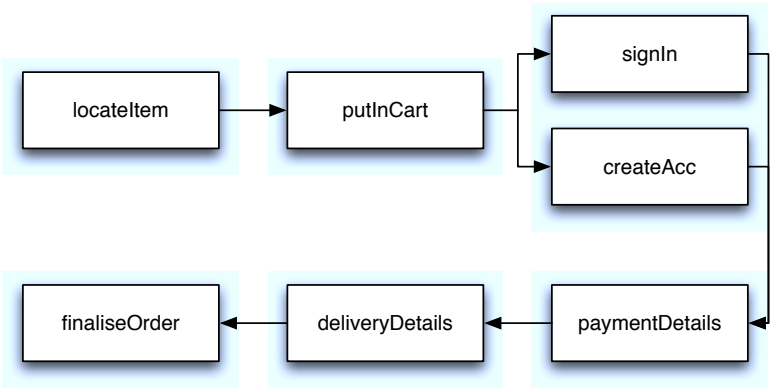

Figure 7.9: Process Model For BuyGenericItem

around the purchasing of a generic item. When the test case was initially created the item was not generic; the item was an automobile. The decision was taken to generalise the subject of any sale so that in theory this test case could be used to test a more generic shop model.

In this example, it is likely that the first step of the process model (locateItem) would also be provided as a standalone service for the retrieval of product codes and other information by the service provider. This is perceived to be common practice such that purchasing services can find benefit in offering a client the ability to search product lines without the requirement to buy later in a process.

There is one key limitation of the BuyGenericItem test case; there is no iteration over locateItem and putInCart. What this means is that in this example a client can search for an item once only and can only add one item to their shopping cart. In reality, it is likely that a client would like to search multiple times (a use for the standalone locateItem process described above). It is also likely that the client would want to add multiple items to a shopping cart so that many purchases can be made at the same time. The decision to have a process with no iteration was guided partially by the initial input process model resulting from the automated generation phase and partially by a wish to prevent the process model from being overly complex. It was felt that this simpler example was already accurate enough to test many areas of NPE. For example, security constraint handling is included with a login or create account choice and a payment process is included to test handling of sensitive data. Further constraints were placed on this example in the form of the removal of constraints and processes triggered by failures at any point of the flow. These were removed as NPE does not currently support OWL-S conditions or effects and so they are not needed for testing.

As with the majority of purchasing services, during the BuyGenericItem process there is a transfer of funds between the two parties. This if followed by a transfer of a proof of purchase. The security of these steps and the ability to monitor where they occur within

a process model are key abilities of NPE. As such this example is key to the testing of NPE in regards to validating its use in a consumer based situation.

The OWL-S document detailing BuyGenericItem can be found in Appendix A.5.5.

OWL-S Use Case BuyGenericItem enables a requester (client) to purchase a generic item from a service using a straight forward and common flow. The item being purchased is first searched for using a set of free search terms, before being added to a shopping cart. Once in a cart the client can the pay for the item and arrange delivery. This process model requires that the client is logged into an account before the purchase can be complete, it also requires that payment details and provided. No allowances are made for failed payments, or out of stock items. What is more, this use case features no loops or other iterations so only one item may be added to the cart and purchased, at any one time.

The following use case was defined:

1. The client first searches the store for an item using a free text search term.
2. The item is then placed in a shopping cart.

3a. If the client has an account they must send the account details to SignIn and received an account ID following a successful login.

3b. If the client does not have an account they now send some account details to CreateAcc and will receive an account ID in return.

4. The client then specifies payment details by sending credit card information.
5. The client then specifies delivery details.
6. Finally, the client authorises the sale and receives a receipt in return.

No alternative or invalid flows were included for this use case, as NPE does not currently fully support OWL-S conditions and effects.

7.4.3.2 NPE Test Case: BuyHouse

The final test case centres around a simplified house buying process (Figure 7.10). The process of buying a house is usually quite a complex one, involving numerous iterations over several processes before a suitable house can be either found, or a price settled on. In this test case no such complexities exist. A six step process was generated to simulate the purchasing of a house where the house is already known (this example would be another use for the separate singular search process as described in Section 7.4.3.1) and only one

Process	Type	Inputs	Outputs
BuyGenericItem	Composite	#SearchTerm #AcctInfo #SignInData #CreditCard #PropertyAddress #PackagingType #DeliveryType	#Shipment #AcctID
BuyItem	Composite	#Item #AcctInfo #SignInData #CreditCard #PropertyAddress #PackagingType #DeliveryType	#Shipment #AcctID
BuySequence	Composite	#SearchTerm #AcctInfo #SignInData #CreditCard	#AcctID
SignInAlternatives	Composite	#AcctInfo #SignInData	#AcctID
SignInSequence	Composite	#SignInData	#AcctID
CreateAcctSequence	Composite	#AcctInfo	#AcctID
LocateItem	Atomic	#SearchTerm	#Item
PutInCart	Atomic	#Item	
SignIn	Atomic	#SignInData	
CreateAcct	Atomic	#AcctInfo	#AcctID
LoadUserProfile	Atomic		
SpecifyPaymentMethod	Atomic	#CreditCard	
SpecifyDeliveryDetails	Atomic	#PropertyAddress #PackagingType #DeliveryType	

Table 7.16: BuyGenericItem Processes

offer is needed to be made to purchase the house. The reason for the over-simplistic nature of this test case was to satisfy a requirement to test NPE's handling of the order in which legal requirements must be fulfilled. There is no interest in any further testing of NPE's handling of potentially complex control constructs. Buying a house involves numerous legal requirements which must be fulfilled in order for any sale to be valid. In the wrong order, missing steps, or with additional steps, a malicious (or unorthodox) service could easily cause a sale to fail or harm either participant. It is against these kinds of errors that NPE aims to protect.

The OWL-S document detailing BuyHouse can be found in Appendix A.5.6.

Process	Use Case
BuyGenericItem	A composite process encompassing all other processes.
BuyItem	A composite process encompassing: BuySequence, SpecifyDeliveryDetails and FinaliseOrder.
BuySequence	A composite process encompassing: PutInCart, SignInAlternatives and SpecifyPaymentMethod.
SignInAlternatives	A composite process encompassing: SignInSequence and CreateAcctSequence.
SignInSequence	A composite process encompassing: SignIn and LoadUserProfile
CreateAcctSequence	A composite process encompassing: CreateAcct and LoadUserProfile
LocateItem	Given a number of free text search terms, returns an associated item from the shop.
PutInCart	Puts the provided item in a new shopping cart.
SignIn	Signs a user in using the provided credentials and returns an account ID.
CreateAcct	Creates an account based on the provided credentials. Returns an account ID.
LoadUserProfile	A stub for loading a users profile.
SpecifyPaymentMethod	Enables the client to provide payment details in the form of credit card details.
SpecifyDeliveryDetails	Enables the client to provide packing and delivery details.

Table 7.17: BuyGenericItem Use Cases

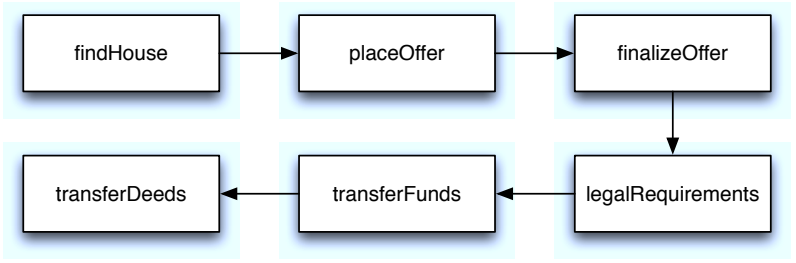

Figure 7.10: Process Model For BuyHouse

OWL-S Use Case BuyHouse proposes a process for the purchasing of a house. It enables a requester (client) to find a house within a property database, place an offer on that house, finalise said offer and then fulfil a number of legal requirements to facilitate the transfer of deeds. This process gives the client and seller no chance to exchange multiple offers, nor does it handle any sort of failure.

The following use case was defined:

1. The client first searches the house he/she wishes to purchase using location data.
2. Next the client sends an offer for the house. In return an acknowledging offer is received.
3. The offer is said to have been accepted by the client this acceptance is returned to the service where it is finalised and the final offer amount is returned.
4. The client then sends the finalised offer to the service where it is added to a number of documents in a set of legal reminders and returned.
5. Based on the final offer and data from the legal requirements, the client transfers funds and is returned a receipt.
6. Finally, the deeds can be requested and are returned finalising the sale.

No alternative or invalid flows were included for this use case as NPE does not currently fully support OWL-S conditions and effects.

Process	Type	Inputs	Outputs
BuyHouse	Composite	#SearchTerm #PropertyOffer #FundingInfo	#PropertyListing #Receipt #Deeds
OfferSequence	Composite	#PropertyOffer	#PropertyOffer
FindHouse	Atomic	#SearchTerm	#PropertyListing
PlaceOffer	Atomic	#PropertyOffer	#PropertyOffer
FinaliseOffer	Atomic	#PropertyOffer	#PropertyOffer
LegalRequirements	Atomic	#PropertyOffer #FundingInfo	#Agreement
TransferFunds	Atomic	#Agreement	#Receipt
TransferDeeds	Atomic		#Deeds

Table 7.18: BuyHouse Processes

Process	Use Case
BuyHouse	A composite process encompassing all other processes.
OfferSequence	A composite process encompassing; PlaceOffer and FinaliseOffer
FindHouse	Searches a repository of houses for sale looking for details of the house at the given location. Returns a property listing.
PlaceOffer	Places an offer for a property.
FinaliseOffer	Allows a client to submit a final offer. The result is an offer updated with all changes and taxes.
LegalRequirements	Converts the given offer into a set of legally required documents.
TransferFunds	Allows a client to send a set of legal documents (including funding information) and initialise a transfer of funds.
TransferDeeds	Returns the deeds for a house.

Table 7.19: BuyHouse Use Cases

Chapter 8

Test Cases: Norms

The methodology laid out in Chapter 6 required not only the service based test cases produced in Chapter 7, but also a set of well-formed norms for reasoning over. This requirement echos that of the initial solution requirement stated in Section 1.4.2 that a set of norms will be required to answer the govering reseach quetion posed in Section 1.4. Research conducted as part of Chapter 2 uncovered a large amount of research surrounding norms and computing. However, the body of research into offline norm generation is significantly smaller than that of service based computing and the Semantic Web (on which the service based test cases are based). As a result, there was an even smaller body of existing example norms for use as a basis for testing. Due to this, the majority of the norms used for testing and validation had to be created by this research. The following chapter outlines how norms were created and suggests a method of categorisation which enables richer testing of the NPE-M module produced in Chapter 5.

Equal consideration must go into creating norm test cases as has been put into creating service test cases. Any bias towards a particular hypothesis or outcome undermines the testing procedure and research. However, in the context of NPE, bias in the norm catalogue test cases was of less concern than it was for service test cases, because the NPE norm catalogues were created by humans for use in a variety of domains. It was not so important to test a subset of all of the norms available in the world, rather it was of more use that a set of norms applicable to the scenarios provided in Chapter 7 were chosen. These were then augmented with a number of norms outlining the need for a global context and globally applicable norms. A harder challenge than removing bias in the creation of norms was finding an accurate NPE-L based representation of norms in use in the "real world".

Unlike the test cases for service descriptions, test cases for norms were static. Four main test case sets were created, falling into two categories which were rated by severity of norm: hard and soft. Hard norm catalogues contained stricter norms which dictated what might be forbidden and were far more likely to halt the processing of a description due

to infringements. Soft norm catalogues contained norms which could have far reaching effects and scope, but would still enable the processing of a description despite violations. The aim in having these two categories was to look at the effects of differing norms on the processing. Within these two categories two separate sub-types of catalogue: long and short. Long catalogues contained numerous norms, many of which could be interlinked (triggering one norm could lead to another being triggered). Short catalogues contained only a few norms. The intention between having these four types was to test NPE's handling of a wide set of structures of norm catalogues and types of norms rather than test the handling of semantically different types of norms.

A separate fifth catalogue of norms was be shared by all actors. This last type contained a set of globally applicable internal norms. These were Global norms (Section 8.3.1) which are vital to business and universally applicable that it would be reasonable to expect every ethical actor would have to take them into account. These were internal to each actor, loaded at start-up and offered the ability to test NPE's processing of norms held in memory. More information is be provided about these norms in Section 8.3.

Creating norms manually introduces numerous unwanted side effects. As has been mentioned previously in this section, the issue of bias is of particular importance in a testing situation. This also holds true for the creation of norms for use in wider business contexts. The introduction of bias by a developer or investigating group at the design phase can radically alter the outcome of any results. This may include altering the selection of business partners, decisions on which intentions to pursue, or influencing the outcomes of any future negotiations. For these reasons, bias must be closely measured during both testing and final implementation phases. However, not all bias is bad. It could be argued that cultural behaviours can present themselves as bias. A developer from one culture may inadvertently bias a set of norms in favour of what he/she perceives to be "normal". Many of the norms introduced will be created due to actions prevalent in the culture surrounding the implementing business. Bias of this nature was accepted within this research so long as the synthesised norms were used within the current culture.

To overcome many of the problems created by the manual specification of norms, a number of automated approaches were proposed by the authors of this thesis. This chapter does not specifically cover the implementation of these systems. However, this chapter leverages some of the ideas created during these proposals to aid in the production of norms. This follows on from the automated process model generation techniques used in Chapter 7. Automated techniques were used to aid the creation of norms for testing.

8.1 Aims

It was the aim of the norm test cases to provide a broad range of norms against which service test cases could be evaluated. It was intended that a set of test cases would be

formed which would allow the analysis of NPE on many levels with the fewest number of runs. These test cases were produced to test functional, specification and stress handling abilities. Two main categories of catalogue were differentiated: hard and soft. These aimed to test the limitations of NPE in controlling behaviour, both functionally in the behaviours which are produced by NPE and also as a comparison to the initial specification. The research intended to test how far norms could be taken before NPE rejected too many descriptions (making a goal impossible). Within these two categories, there were long and short catalogues. The intention was that this would allow the testing of the speed at which NPE worked, allowing a measure of whether it was worth running NPE and potentially loosing time and resources. Alongside this ran a single static internal norm store which was the same for every actor. This aimed to provide a base of norms which was constantly available and to enable the testing of NPE when working with in-memory norms (norms which would not need to be parsed and checked).

On a broader note in relation to NPE's final goal of providing a method for the regulation of actions based on globally dissimilar norms, the aim of the following cases was to provide a set of norms which could be related to the differences found when comparing distinct cultures. Differences included those related to communication specifications, business practices and customs surrounding the exchange of goods and services. As such the norms prescribed below aimed to provide a base on which to test NPE allowing for both basic and more complex global norms to be analysed.

Whilst evaluating the research aims it was important to remember that only a small subset of possible tests could be completed. In designing test cases and a testing protocol it was important to take into account this limitation and build a set which best represents the aim of testing. As such, the test cases created in the following chapter were created on a best effort basis; the aim being to cover the widest range of testing criteria whilst ensuring that input test cases were focused enough to give adequate results.

It was not the place of this research to expressly evaluate any produced test cases against the above aims. This research focused on the suitability of NPE, not on the suitability of any NPE-L based test cases. However, the authors accepted that it was likely that an ad-hoc evaluation of the test cases would emerge during the NPE testing process. It was expected that this would manifest its self as bias or a partial failure of areas of the testing process.

8.2 Norm Catalogues

Five norm catalogues were prescribed. These provided normative NPE rules which governed general interactions and business relationships both generally and more specifically in relation to the expected behaviours involved in the three categories of services defined in Chapter 7. As has been mentioned above each catalogue had a different aim. This

ID	Category	Length	Aims	Notes
1	N/A	N/A	Test performance hit from in memory norms.	Global norms.
2	Hard	Short	Test handling of harder norms with blocks.	Less specific blocking norms.
3	Hard	Long	Test "worst case".	Complex but by nature specific, blocking norms.
4	Soft	Short	Test "simplest case".	Effectively suggestions.
5	Soft	Long	Test handling of a large number of norms and the performance hit taken when implementing.	

Table 8.1: Norm Catalogues

ran parallel with the goal of making each of these catalogues as representative of the requirements of a "real world" situation as possible.

It was the intention of the testing protocols laid out in Chapter 6 that NPE be tested by using both internal and external sources of norms. To this end, four external catalogues and one internal catalogue were created. The single internal catalogue was a relatively small generic store; details of which can be found in following sections. The four external catalogues were spilt by two variables: length and category. Each variable had two possible values, giving four unique catalogues for testing. These catalogues have been outlined in Table 8.1 and are described in the proceeding paragraphs.

Each of the five catalogues held a small set of norms. These norms were prescribed in NPE-L and were verified against both syntactical and presentational errors. The normative contents of each catalogue had also been verified for validity both internally in relation to norms held in the same catalogue and also in relation to all other norms in the other test catalogues. Although the four external norm catalogues were not used together in testing (only combinations of a single external and single internal catalogue were used), the decision was made to verify all of the prescribed norms to try and reduce any bias. It was also decided that for these tests the "world" modelled by the test cases should be as "perfect" as possible. The intention of these tests was to examine the limits of NPE, rather than examine how NPE might counteract any imperfections in the surrounding world, or how inconsistent norm stores might be handled.

Internal Catalogue The internal catalogue had the ID 1 as prescribed in Table 8.1. All NPE tests shared one common internal store of norms. This internal store was used

during every test and formed the basis for all norm checking. The internal store of norms contained a small set of uncomplicated norms. However, these norms were seen to be so important as be applicable to every business situation. In this way, these norms are analogous to the super-norms described in Section 8.3.1.

Catalogue ID: 2 Catalogue 2 featured a small selection of norms which were considered to be relatively "tough", norms which restrict behaviour. Being short these norms were typically atomic, with no interlinking and basic triggers.

Catalogue ID: 3 Catalogue 3 was the strictest of the catalogues used during testing. The norms contained within all place strict restrictions on behaviours. Norms within this catalogue were also interlinked and triggered through complex preconditions. The aim of this catalogue was to try a "worst case" for NPE to handle, a set of norms requiring a large amount of extra processing; a set which is more likely to fail a proposed service process definition (workflow) than pass it.

Catalogue ID: 4 Catalogue 4 was the "easiest" collection of norms; the opposite of Catalogue 3. Soft norms very rarely cause processing to halt, nor do they often forbid actions. Soft norms were intended as placeholders or warnings. They allow processing to continue and the final decision about whether to allow a process definition (workflow) to continue to be left to the overseeing agent.

Catalogue ID: 5 Catalogue 5 took the norms outlined in Catalogue 4 and applied a larger number of them with more complex triggers. The aim of this catalogue was to test NPE's handling of a large number of norms. The results gained from tests with Catalogue 5 were compared to those of Catalogue 4 to gain an insight as to the ability of NPE to both process norms from a catalogue (the import stage) and to process multiple process definitions (workflows) without failing with varying numbers of norms.

8.3 Internal Norms

The intention of the test cases outlined in this chapter is to enable as far as was possible the unbiased testing of the NPE system. The NPE testing methodology is detailed in Chapter 6 and undertaken in Chapters 9 and 10. For every test, a set of NPE test cases was required (along with a set of service description test cases). Test cases of norms consisted of NPE-L based normative catalogues. Five catalogues were prescribed as outlined in the previous section. Among these was an internal store of norms intended as a baseline from which other catalogues could be compared. This set was simpler than the other test cases sets prescribed and was held internally in memory at runtime.

As its' name suggests, during testing the internal norm catalogue was stored internally within the NPE based system under test. There were several reasons for this. First, by storing a set of norms in memory within NPE, the import phase of any NPE run could be bypassed. Second it was likely that many implementing agents would want to have often used norms stored locally for rapid access. It was expected that the import phase would incur significant performance costs. As external NPE catalogues were either be imported from the web or read from files on disk, there were multiple extra costs associated. Seek times and download speeds posed the largest possible costs, with these being possible 100 or more times slower than accessing in-memory stores. Catalogues held internally are also already held as NPE objects. External sources were described using XML based NPE-L, requiring interpreting and converting to NPE objects before use. This step again added further costs to the NPE cycle which did not exist for internally stored norms. The use of an internal set of norms as part of the NPE testing protocol aimed to remove the extra costs associated with importing catalogues of norms. By doing so, this was intended that these extra costs could be calculated so that the semantic benefits of NPE could be weighed against any performance costs. This testing step was added as it was considered that a large number of the norms used by systems employing NPE were likely to be held internally as trusted set of in memory catalogues.

The proposed internal norm catalogue test case consisted of a relatively small number of norms. These norms were simple compared to more complex norms outlined in some of the external catalogue test cases (Section 8.7.2). Very few of the norms contained were blocking. That was to say that the aim of these facts is not to enforce norms but to offer warnings for or against actions. The intention was that these norms would cover a wide range of activities and be applicable to almost every group and culture. This was reflected in the design of the testing protocols as these norms were utilised during every NPE testing run. As these norms were intended to govern the majority of activities undertaken irrespective of the actors location or culture, they were analogous to the super-norms described in Section 8.3.1.

The testing methodology outlined in Chapter 6 and undertaken in Chapters 9 and 10, prescribed that the internal norm store case would be used for every testing cycle (with exception of the dummy agent, agent 0, which did not use NPE). As has been mentioned in Section 8.2, all of the norm catalogues prescribed in this chapter were checked for internal consistency. Attention was also be paid to ensuring that all of the external norm catalogues are consistent with the single internal store. This was important as the internal was used in conjunction with every other norm catalogue.

8.3.1 Super-norms

This research supports the conclusion that there are different levels of norms. In everyday speech, the notion of a norm actually represents a second tier norm. Norms can be

ordered in groups by a number of different factors including: range, severity, longevity and complexity. Grouping norms like this makes it easy to create a bottom-up hierarchy in which norms can be classified. If norms are grouped by scope then it is possible to place norms with a narrower scope at higher levels. Conversely, norms with a wider scope of influence will be placed lower in the hierarchy. The highest level norms are likely to not only be tightly scoped but also temporary in their existence. Norms which are stacked on top of one another inherit properties from their parents. However, it is possible to have gaps in normative hierarchies with norms which are higher level, but have characteristics distinct to a single group or activity. An example of a normative hierarchy can be seen in Figure 8.1. Norms in the bottom level have the widest scope and are likely to be universal. In this research, the recent trend in referring to these lowest level norms as super-norms is followed.

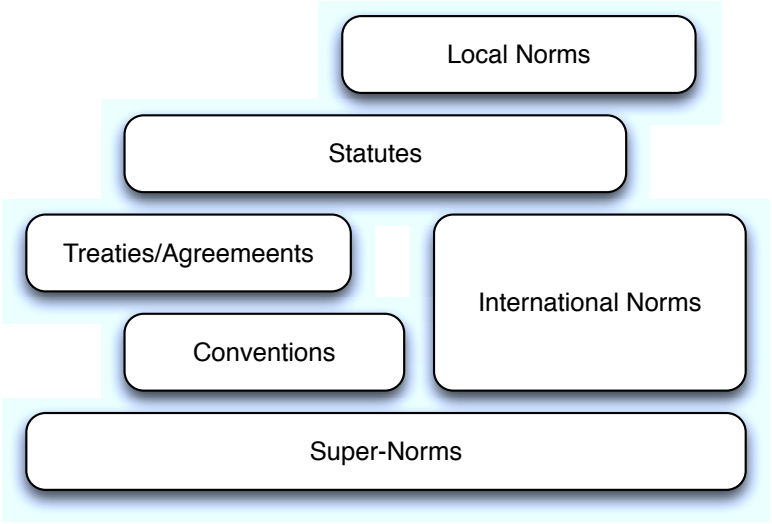

Figure 8.1: Normative Hierarchy

The above ideas introduce the notion of a super-norm; normative facts which operate at the lowest level of any normative hierarchy. Scoped across all domains, groups and cultures, super-norms are universal. Realistically in practice such norms may be scoped at an international level or by large multinational organisations, since universal scope is difficult to provide on a system as distributed as the internet. Super-norms characteristically consist of simple concepts which are so integral to society that they form the very basis of its fabric. Consider the few basic human normative facts which form the basis of any civilisation, such as not intending to harm others. Without these, civilisation would not exist. Super-norms can be seen as the basic building blocks on which reliable societies and civilisations can be built.

There is an existing body of research surrounding super-norms and their existence, classification and uses. Most notably the UN have produced a document called the UN Global Compact (King, 2004; Barnett et al., 2009). This can be viewed as a charter

businesses can adhere to, containing nine principles for activities that should be avoided during business. Intended to start a debate as to the nature of norms and standards in international business, the UN Global Compact forms the starting point for a true set of global super-norms.

The above documents were used to form a set of supernorms for inclusion in the internal norm test case. It was indented that by mixing "real world" examples with examples created just for the testing of NPE, bias could again be avoided whilst preserving the high quality of the test cases.

8.4 Norm Generation

The key to the success of NPE is the existence of a base of normative facts. These facts have to be represented in NPE-L and accessible to any NPE based system wanting to utilise them. Without the existence of any norms NPE if essentially redundant (as it would also be redundant is there were no input process models). Norms can either be produced as new NPE-L facts, or discovered in existing NPE-L repositories. As such, a key component to the success of the NPE system is the creation of normative NPE-L facts. Responsibility for the creation of norms generation techniques to enable the use of NPE falls initially to the creators of NPE, as without an initial set of norms, NPE cannot be tested or used to answer any questions posed.

In Chapter 7 the generation of service process test cases was discussed. Normative facts were generated using similar techniques. Many of the principles were the same, as both rely on the existence of domain and process knowledge about a specified field. There were two options for the creation of normative facts in NPE-L: automated and manual. These are analogous to the automated and manual techniques discussed in Section 7.2.2. Each has advantages and disadvantages. The rest of this section discusses these two approaches, before taking an in-depth look at the use of automated techniques for the rapid generation of norms for testing.

8.4.1 Manual Techniques

Manual normative fact generation techniques lend themselves to the generation of small numbers of norms. Norms generated manually tend to be more specialised and domain focussed. On the whole, conditions tend to be more focused and complex. Manual techniques require a developer to create every norm by hand. Automated tools may be used to speed up some of the creation process but, on the whole, the decisions and end result are the responsibility of a single (or group) developer. Creating normative facts requires an in-depth knowledge of the domain over which normative reasoning is needed. Additionally knowledge of how any created norms are to be used is useful. The manual

creation of norms is relatively slow and as such is only suitable for the creation of small numbers of norms. The creation of complex norms is straight forward and accurate. This can be made easier through the use of automated verification tools and action logics. One of the main problems with manual techniques, however, is the introduction of bias into any norms by the developer. This may be desirable where a particular behaviour is wanted, however, in general, bias is unwanted.

8.4.2 Automated Techniques

Automated techniques rely on specially created programmes to carry out a lot of the work. In regards to the creation of norms, automated techniques mainly rely on the ability to spot trends in or patterns in collections of workflows. By spotting trends within numerous workflows belonging to the same domain or culture, it is possible to extract norms. These norms are only inferred from perceived behaviour. They rely on an assumption that society as a whole is good and that most actors will act in a way that will maximise profits. Automated techniques are useful for creating larger collections of norms, or generating norms for domains about which no knowledge in known. However, norms created in this fashion are likely to be simplistic and relatively broad in their scope. A further look at automated techniques follows, along with a look at how they have been employed in this research.

Automated generation techniques allow the rapid creation of numerous normative facts. With automated techniques the production of "perfect" normative facts is secondary to speed and availability. Although norms generated automatically are likely to be structurally "perfect" (so long as the code creating any normative facts is correct), automatically generated norms aim to give a general feel of the expected actions and processes occurring within a given domain. It is most useful to use automated techniques when either little information is known about a new domain, or a large number of new norms is required.

Several techniques for the automated production of norms exist, most notably: synthesis, conversion and extraction. Synthesis relies on the production of norms from small component parts, as demonstrated with the synthesis of process models in Section 7.2.2. Cases produced using syntheses can be very accurate and complex. However, this approach requires a large number of norms to be produced (or parts of norms) and is of little use here. Conversion relies on the existence of stores of norms which can be easily converted straight into formalised normative facts. This would be useful if there were existing stores of norms, but there are no such stores. The most promising technique is that of extraction. Using extraction normative facts can be retrieved from sets of workflows or process models. This is the technique which this research focuses on and describes in this section.

Extraction works for the creation of norms because it relies on the existence of collections of workflows and process models. Such repositories already exist and are well populated. It is possible to look the future when even more repositories will exist and when the creation of normative facts in any formal language will be even more reliable. These repositories exist as both standalone business processes (exposed using languages such as BPEL) and as part of service discovery languages (such as OWL-S). As this research is interested in further uses for OWL-S, a link to the process model was useful and provides an increased incentive to use extraction. For extraction to work it has to be assumed that by grouping together service descriptions by domain or purpose and further by culture, region or ethnicity, sets of services with similar objectives and process models (workflows) will emerge. By identifying trends in these groups as processes to sequences of processes, normative behaviours can be extracted. The conversion to NPE-L is then straight forward (as the whole extraction system is set up to produce NPE-L type data). A difficulty exists in striking a balence between producing too stricter rule which would apply retrospectively to actions only perceived or actions extremely similar to those already perceived and producing rules with constraints which are too relaxed.

Using the above method, it is possible for an actor to build a catalogue of norms from a set of perceived or otherwise external process definitions (workflows) even before joining a society. This can both reduce development times and act as a guard against potentially being treated poorly due to being "new" in a society. However, this approach does rely on the majority of the actors in any one group to be acting as to maximise profits. Care should be taken as this method can be exploited by collaborating groups of agents forcing behaviour by agreeing to expose poor process definitions (workflows). This method is also potentially flawed in the long run as is stands in the way of process and innovation. Actions are decided on in relation to what activity most of the rest of the actors is a system are partaking in, not in relation to how the current actor can outperform the competition.

8.4.3 Automated Norm Extraction

For NPE to be tested effectively two forms of input data were required: a set of test cases for services and at least one catalogue of norms. The automated (and manual) production of service descriptions for test cases has been covered in Section 7.2.2. With these in place, the second requirement had to be fulfilled through the production of one or more catalogues of norms. To aid the generation of norms for the testing of NPE, automated techniques were employed. These were complementary to the service generation techniques found in Section 7.2.2.

The automated generation techniques used during this research were based on the extraction model outlined in the Section 7.2.2. Norms were extracted from sets of process models through the identification of trends within flows of processes and information.

In the case of the example outlined in this section, the definitions parsed originated from OWL-S process models. Repositories of OWL-S service descriptions identified in Section 7.2.1. Descriptions can be found in Chapter 7, with a list in Table 7.1. As with the service description creation process outlined in Chapter 7, the greatest problem in relation to the location of original OWL-S documents was the lack of largescale repositories of reliable examples. This was especially true for descriptions featuring complex process models.

The system used was designed to reuse as much of the code created in Chapter 7. The system design can be found in Figure 8.2. It used as many modules from the automated process model generation system as possible. Initial OWL-S importing relied on the OWL-S API based model created in Section 7.2.2. The main differences in the initial set-up and import phases were that the input service descriptions were sorted more thoroughly into domains and tasks. When loading repositories each domain and task set was kept separate so that trends were discovered within individual domains. The primary goal for this was that stronger trends would be found.

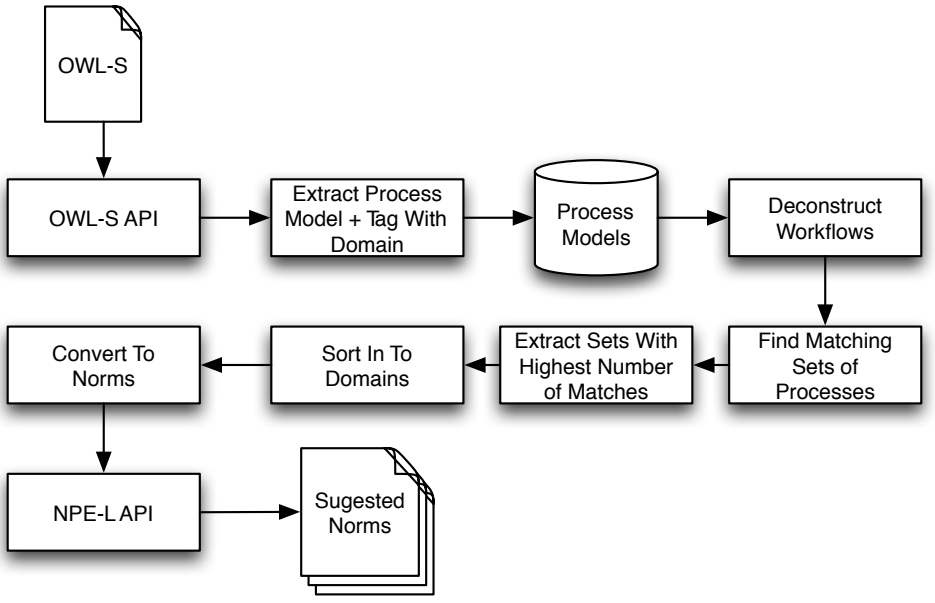

Figure 8.2: Automated Norm Generation System

Having loaded a repository into memory, processes were then extracted from the contained definitions. This happened in much the same way as in Section 7.2.2. Processes were extracted from the process models of parsed OWL-S files and stored before a matching algorithm attempted to identify trends within them. Trends were identified by examining single processes and groups of linked processes and comparing them to others. The majority of the processing occured over I/O and relied on similar matching metrics as those outlined in Section 7.2.2 and Table 7.3. Identified trends were given a "score" based

on their perceived significance and the top few were selected for conversion to NPE-L facts.

The conversion process was relatively straight forward but crucial to determining the scope and impact of each normative fact. A generalisation process occurred on every fact identified as a valid trend. This process started with a simplification of the associated role to one of a few types selected from Table 7.2. Following this, the action name was simplified using a modified potter stemming algorithm (Porter, 1980) to make it as generic as possible. Conditions were extracted from the surrounding processes. An amount of inference was used to attempt to produce accurate conditions for the widest range of situations. The result of these steps was the production of a modified OWL-S process with a set of associated conditions. To convert the generated process model to an NPE-L fact, a norm type had to be decided. This process determined the impact of the norm. For this Thesis, the decision surrounding which norm type to select was carried out manually as no single algorithm was deemed suitable. A proposed system was tried using the above significance scores along with a database of role and action types, but this produced unsatisfactory results. Having decided on a norm type, NPE-L facts could then be generated. For some norm types, this required an extra step of negating the action controlled by the norm (for example to forbid not performing an activity, rather than permitting).

The collection of process, conditions and normative types can then be exported using the new NPE-L API module. More information about this module could be found in Chapter 6. For each group of descriptions processed, a single normative catalogue was produced. These catalogues were then sorted manually and combined with norms created manually to form the five required catalogues.

8.4.3.1 Results

The system proposed above were created and utilised to help produce a set of NPE-L based normative catalogues. Execution of the system was carried out over groups of service descriptions taken from the repositories listed in Table 7.1. Execution was carried out on a dual core Apple Mac system using 4Gb of ram. Extensive lessons were learned from the automated production of service descriptions in Chapter 7. As such the initial Java heap size was initially set at 1Gb to prevent heap size errors. The code was also carefully written to eliminate as many loops as possible. As a further performance measure, the initial loading of service descriptions from XML was made a fast as possible so that the majority of the calculating could take place in memory. Despite all of this, each run was still relatively slow and took up a large about of resources.

In total, the initial set-up and loading phase produced a set of 1103 individual processes and 982 flows of multiple processes. These were extracted from seven initial domains.

From this an average of 200 suggested norms were produced with an average of 28 norms per domain finally making it through selection into the export phase (outputs were weighted towards the three domains used in the creation of service test cases). These norms were then collected into the five catalogues required for testing (being careful to not mix domains in such a way as to divide the norms any one single catalogue). These catalogues were then augmented with manually generated norms to form the final catalogues (summarised in Table 8.1 and described in Section 8.7).

8.5 Norm Types

For the purpose of this research, a method for grading norms was proposed. This method was informal and non-rigorous. It did not form a formal part of the NPE structure. However, it enabled the description of the differences between norms, which was useful for testing procedures. This section describes these grading terms and how they were used to sort norms into catalogues to aid the testing of NPE.

The catalogues created above were specially created containing norms in no particular order or grading. The description of the catalogues proposed in Section 8.2 required that norms presented in each catalogue were a specific size, complexity and strength. To achieve this, the previously generated catalogues had to be reorganised. The end result was five catalogues, with each containing sets of norms from different domains, but with similar characteristics. This reorganisation process was touched on in the last section. What follows is a further look at the processes involved and the reasons behind them.

The primary reason for this sorting process and associated notation techniques was to enable NPE to be tested to as fuller extent as possible within the time-scale of this research. Five unique catalogues were required for the NPE testing protocol outlined in Chapter 6 and Chapter 9. Whilst creating these catalogues it was important that the domains covered by the service description test cases (Chapter 7) were also covered by norms. Each catalogue contained a sample of norms from each domain. No domain was left out of any one catalogue. Additionally no catalogue (for testing purposes) contained norms applicable to a domain which is not part of the testing protocol. These decisions were taken to ensure that during every testing run the NPE system was able to process an unbiased set of norms and descriptions. The intended result was a set of runs not skewed by bias or the importing of unused extraneous data. The original domain based catalogues created in Section 8.4.3 was be split into five new catalogues based on the metrics outlined in the following subsections. This split was carried out manually, as the number of norms present was manageable by hand.

8.5.1 Categories of Norms

The primary mechanism for the grading of normative facts for use in the NPE testing process relied on placing norms into categories depending on their perceived severity. Norms are sorted by how rigorous their effects are. This sorting was carried out in a relatively straight forward fashion by initially ordering normative facts from most to least harsh. This metric was essentially a measure of the effects of each norm and may be compared to a measure of perceived instantaneous effectiveness of the norm (in influencing potential actions), whereby the most effective norms leverage the most control on any NPE based system.

For the purposes of this research only two classifications of categories of norms existed: hard and soft. Hard norms had more impact and can be said to have leveraged more (direct) control over the behaviour of systems implementing NPE. Hard norms were essentially defined as norms which aimed to block or change actions intended by the implementing system. These norms were likely to be coarser in their scope and were more likely to halt processing if broken (it could also be said that hard norms are more likely to be broken, although this was not always true). Conversely, soft norms are not as likely to block processing or deny access to activities. Soft norms were characterised as having much looser levels of control. Soft norms were likely to contain facts based around permit rather than forbid, making them much more akin to recommendations and suggestions than hard norms. At first glance, soft norms may seem irrelevant when compared to hard norms. However, the two cannot be compared directly; they must be seen as aiming to achieve different goals. For hard norms, the goal was to limit the number of possible actions available to an actor at any one time. Conversely, soft norms suggested actions to be taken which might be beneficial. Both had an important role to play in the planning and execution of potential strategies.

These two categories of norms complemented each other. Hard norms should not be seen as more important than soft norms. Rather they were seen as working together to aid task planning and service choice. One limited options, the other suggested routes to take (or avoid).

NPE catalogues created in Section 8.4.3 were sorted into hard and soft categories. These are then grouped again into domains before being loaded into the appropriate normative catalogue (See Table 8.1). The intention was that each catalogue would have a different bias towards a category of norms while at the same time all catalogues would have roughly the same number of norms from each domain. In creating these groups the indicators used were to look for attributes such as forbidden (hard) vs permitted (soft) and blocking (hard) vs non-blocking (soft). All of these final steps were carried out by hand due to the relatively small number of norms used.

8.5.2 Norm Complexity

The second metric by which norms are sorted was much simpler. A measure was created by which the normative facts created above were sorted by complexity. Complexity in regards to this metric was a measure of how complex the norm might be to process. No exact processing was undertaken to evaluate norms and discover figures for this metric. However, thought was introduced to highlighting the areas which could take up resources. It is worth noting that this research will also be useful in the creation and evaluation of the NPE system in Chapter 6. Areas which have been accounted for include: complex conditions, long sets of actions (norms can apply to sequences of actions as well as individual actions; Section 4.2.1.2) and interdependent chains of norms.

It is important to emphasise that complex does not have the same meaning as "long". Complexity in this context meant the potential resource costs caused by utilising the norm. These costs are mostly computational and come in many forms. Of most importance to this research were the extra costs incurred during processing of: conditions, action sequences and linked norms. Conditions were important as there were two to each norm and both required processing before a norm could be activated. Conditions reason over beliefs about the surrounding environment, internal information, previously executed norms and currently executing actions. The interpretation and evaluation of conditions is a non-trivial task and as such could potentially place a large strain on any implementing system. Allowing norms based on sequences of actions and not just one single action is useful, but the detection of sequences and not just single actions causes the resources required to increase significantly. Suddenly when allowed, this causes the initial process model import to be squared. Finally, the chaining of norms was handled by NPE as an internal consideration. Careful design of the chaining procedures should minimise any extra costs. However, it was still likely to cause a few extra costs to be incurred at runtime.

Sorting into complexity was carried out after the normative facts had been sorted into categories, although the initial grading of complexity was carried out on a copy of the norms created in Section 8.4.3 with the gradings transferred to the norms sorted above. The normative facts were then transferred to catalogues in accordance with Table 8.1. The aim of this test was that the more complex a normative fact was the larger its resource requirements would be (so any test run involving complex norms would exhibit slower speeds and increased demands for resources). It may be possible that this behaviour might have mapped to smaller, lighter norms being used more in systems where resources as sparse.

8.6 Conditions

The above techniques for the creation of norms rely on processing numerous Web service process models to detect patterns. Essentially this relies on notions that norms can loosely be defined as being the statistically most acceptable actions. This is an enormous assumption to make as often norms can have very little to do with "common practice". However, for the production of initial sets of norms for testing any concerns were waved in exchange for rapid creation. The sets produced feature numerous prescribed norms; the majority of which adhere to strict styles and typing. Typically the created norms feature single actions or chains of actions linked by compositional structures. The initial power behind these norms is derived from the triggering of exceptions by the action (or actions) addressed as the focus of the norm. In these initial catalogues the creation of conditions for the deactivation and activation of individual norms takes place second. Thus the conditions used are uncomplicated. Often triggers rely on single actions or processing directives such as the beginning or end of a process. Conditions derived from the above processing techniques are simpler than the controlled actions only because it has been found to be harder to effectively create complex conditions using basic automated processes.

Much of the power of NPE-L was derived from the open style in which conditions could be created and handled. Conditions within NPE-L are essentially just statements in one of any number of rule languages. By default these rules were defined using SWRL and no limits were placed on the definition of concepts by NPE-L itself. Rather, the limits were provided by the implementing engine. By using these methods, conditions within NPE-L could be used to represent and control virtually any concept or element accessible at runtime. This made activation and deactivation conditions very powerful. Much of the flexibility and power of NPE-L resides not in the actions controlled by individual norms, but in the conditions controlling the status of norms. As has been mentioned, the conditions created by automated techniques often lack focus and power. Therefore a large number of the conditions in the final norm catalogues were manipulated to increase the viability of the resulting norms for testing. Conditions were changed to broaden or tighten the scope of norms, altering the effects of any controls. In some cases norms had effectively been inverted to enable the conditions to control actions affected by the norm. In other cases norms had been created specifically around conditions, with effects stemming from activation. Such techniques were especially useful in empowering obligations and permissions where action based triggers might have been insufficient to capture complex attitudes or elements of scope.

As conditions can evaluate any elements or concepts visible at runtime, they were not limited to the processing of actions. This capability allowed for the creation of conditions which rely on external concepts. If supported by an implementing engine this would allow NPE-L norms to be triggered by activities within the underlying execution. For example,

norms can evaluate the current version of NPE in use for capabilities, or loaded modules enabling norms to be specified about the inspection process its self. Norms could also be created to be activated on the trigging or activation of other norms thereby enabling the rapid chaining of norms. Such techniques enable NPE to evaluate information separate from the semantic description of services. This could include further information about the service, including: labels, types, versions and other non-semantic data. Many of these conditions need to be created by hand or using specific automated tools. However, the mastering of these techniques will enable NPE-L to become increasingly powerful and open up new avenues for control.

8.7 Norm Catalogs

The processes described in this chapter were used to produce a set of five final catalogues of normative facts described in NPE-L. The intention was to enable the full testing of the NPE system. These catalogues were created with the NPE testing procedure outlined in Chapter 6 in mind. Every effort was made to produce a set of catalogues which best fitted the testing procedures, without introducing bias or influencing the end results of any simulations.

Removal of bias towards any one outcome was the most challenging part of the norm creation process. Unbiased test cases were essential for the full and fair testing of NPE. Five normative catalogues were produced based on the metrics outlined in Table 8.1. Norms have been created originating from the domains in Table 7.2. Norms from these domains were split so that every catalogue had roughly the same number of norms from each. As the testing protocol required each domain of service test cases to be matched up with one catalogue at runtime, this enabled all of the five catalogues to provide valid norms to every NPE testing run. The following is a short review of each of the five normative catalogues (Table 8.1) expanding on the descriptions in Section 8.2.

8.7.1 Global Norms

This research defines global norms as "norms with global scope and influence". Key to the concept of global or super norms is the idea that it is possible to define rules to limit society on a global scale. With regards to NPE-L facts, global norms have no special connotations or annotations other than their meaning provided at design time. During testing, global norms were unique as they were held in memory during runtime.

Five global norms were identified by the processing outlined in Section 8.4.3. These are detailed in part in Table 8.2 and in full in Appendix 11.6. These norms were extracted using automated processing techniques and some manual changes were made to improve condition and role handling. The global norm catalogue introduced a number of concepts

Identifier	Type	Role	Activity
LocationSpoofing	Forb	Role	Any1To1Activity
ExposePersonalDetails	Forb	Client	ReturnPersonalDetailsActivity
Login	Perm	Client	LoadUserProfileActivity
RequireOWLS	Forb	Role	Any1To1Activity
UpdateOtherUser	Forb	Client	UpdatePersonalDetails
ProvideRoot	Forb	Client	ProvideRootTokenActivity
LegalRequirements	Forb	Client	PaymentActivity

Table 8.2: Global Norm Catalogue

which were used throughout testing. Key to much of the power of NPE-L was that, where implemented, generalised activities could be used to match all other specialised varieties. Thus a very basic activity, with very general inputs and outputs, could be used to match all other more complex activities. NPE-L is limited by having to match activities with the same number of inputs and outputs. It is possible for an implementing engine to be produced which could allow for this limitation to be surpassed.

As the most basic building block of any potential process, an activity with a single activity with one input and one output is very useful for the modelling of generalised triggers for norms. In the set of NPE-L testing activities the activity identified as "Any1To1Activity" realised this generalised norm. Introduced by the global norm catalogue Any1To1Activity represented any activity with one input and one output.

Much of the focus on global norms with regards to online transactions and distributed processes is centred on the prevention of the misuse of sensitive data. For example, norms may be in place to prevent the revealing of personal details or private information to a third party, or may aim to secure users from unscrupulous business partners. As NPE-L is not limited to solely reasoning over Web service elements, the catalogue of global norms in this research also introduced the notion of a test on the reasoning system itself. This was achieved in the form of a norm which utilised the ability (where implemented) NPE had to report on any loaded modules. This tested to see if the OWLS-API had been loaded and thus if OWL-S could be used as an expected process deinition language.

The full set of global norms aimed to provide easily identifiable limits which could be placed on any testing system as a whole. On a global scale norms can often be useful to provide sensible guidelines for the use and reuse of networked technologies and connections. It was intended that this catalogue would provide a sample of some of these norms.

8.7.2 Hard Norms

Hard norms represent restrictions and guidelines which have severe consequences. Such norms typically protect against activities or situations which may be seriously damaging or

costly to a participant. Violation of hard norms often lead to a failure in any interaction, unnecessary costs to participants, or a collapse of trust between parties. As such a violation detected by NPE at runtime will often lead to the halting of processing and the service being marked as junk or dangerous. Steps introduced by hard norms often aim to safeguard the integrity of privacy constraints and data providence. Ordinarily these constraints are active throughout the life of any processed service description.

Identifier	Type	Role	Activity
OverGeneralisedSearch	Forb	Searcher	GeneralisedSearchActivity
RequireSecurity	Forb	Client	SecureActivity
ExtraPayment	Forb	Client	PaymentActivity
EmptyWorkflow	Obli	Role	AnyActivity
UnexpectedTermination	Obli	Role	FinalActivity
Fallback	Obli	Client	RecoverActivity
OverGeneralisedAddItem	Forb	Searcher	GeneralisedAddItemActivity
RequireShoppingSecurity	Forb	Client	SecureShoppingActivity
BookMedicalAppointment	Perm	Client	BookMedicalAppointment-Activity
RequireLoginForSecure	Forb	Client	TransferDeedsActivity
ReturnDeeds	Obli	Role	SecureActivity
RequireRecieptAfterExtra-Payment	Forb	Client	RecieptActivity
RequireEncryption	Forb	Client	EncryptedActivity
RequestFurtherInformation	Forb	Role	RequestFutherInfoActivity

Table 8.3: Hard Norm Catalogue

Twelve norms were identified using the automated techniques described previously in this chapter. These are detailed in part in Table 8.3 and in full in Appendix 11.6. The catalogue of hard norms was populated with norms initially created using automated techniques to identify trends. These norms were then amended manually to ensure that they provide a full test set for NPE. There were more norms in this catalogue than in the global catalogue as it was more likely that a norm would have local relevance than full global scope and also as this catalogue was eventually split into two separate catalogues for testing. However, many of the norms present in this catalogue and identified as hard could also be used in a global context.

Included in this catalogue were norms relating to the over-generalisation of processing and data handling where the data expected was of a more specific nature than the actual data provided. Introduced as part of this catalogue was a new role of “Searcher”, a sub-role of Client representing an actor taking part in some sort of searching activity. Also included was a norm relying on the NPE function “next()”. This function allowed for conditions to be based on look-ahead actions, where the interpreter could interrogate the current process model to assert what the next activity would be before fully loading it. Finally,

this catalogue employed both “FALSE” and “once(@end)” deactivation conditions to underline the small but important differences between the two.

8.7.3 Soft Norms

Soft norms are populated by activities which are not serious, but which would be preferable if they were not violated. Violations of soft norms do not represent a critical danger to the processing of any information. Unlike hard norms, if NPE detects a violation of a soft norm during processing, no halt command is issued. Rather, a score is kept enabling multiple soft norm violations to occur and for process definitions to be evaluated after having been fully processed.

Identifier	Type	Role	Activity
OverspecializedSearch	Obli	Client	OverspecializedSearch
FinalizeOrder	Obli	Client	FinalizeOrderActivity
CreditCardRequirement	Forb	Client	DebitCardActivity
ExtraSecurity	Forb	Client	SecurityActivity
StartWithRequest	Obli	Client	Any1To1Activity
UnsupportedControl	Forb	Role	AnyActivity
SpellingError	Forb	Role	AnyActivity
GeneralisedSearch	Forb	Searcher	GeneralisedSearchActivity
UnsupportedLanguage	Forb	Role	AnyActivity
OverspecializedProcess	Obli	Client	OverspecializedProcessActivity
FinalOffer	Perm	Client	FinaliseOfferActivity
FullLogin	Perm	Client	FullLoginActivity
RequestResponsePattern	Obli	Client	ResponseActivity
FinalizeAction	Obli	Client	FinalizeActionActivity
AlterAfterVerification	Forb	Client	AddActivity
FinalizeAction	Forb	Client	FinalizeActionActivity

Table 8.4: Soft Norm Catalogue

Twelve norms were identified using the same techniques as for the global and hard norm catalogues. These can be seen in part in Table 8.4 or in full in Appendix A.4. As with the catalogue of hard norms, this catalogue was larger so that it could be divided for testing. Many of these norms were amended manually to provide a greater variety of testing conditions for NPE.

Many of the soft norms included in this catalogue were aimed at improving efficiency and reducing unnecessary steps within processes. Others were used to test NPE itself in the form of module checks (as seen in the global catalogue). Introduced as part of this catalogue is the idea of custom functions. These are modules which may be implemented by an NPE based solution and can provide extra abilities. In this case, the custom function was a spelling function capable of checking the spelling of any input. This was used to test the spelling of names within a prospective process since incorrect spelling

could be an indication that the process definition had not been carefully constructed. More information on custom functions can be found in Section 6.3.2. Numerous examples within this catalogue also detected over specialisation, where inputs or outputs could be unnecessarily specific and thus may have limited choices or outcomes. Unlike the previous two catalogues, this catalogue was uninterested in preventing the halting of processing or the scrapping of services. Rather, soft norms were a method used for the scoring of potential services as a technique for differentiation.

8.7.4 Catalogues for Testing

The three above catalogues were split into five separate final catalogues for testing. This was carried out manually but in such a way as to balance all of the catalogues in terms of influence over domains of test cases (Chapter 7). Details of each of these catalogues can be found in Section 8.2. What follows is a brief description of each catalogue and a listing of norms contained within.

8.7.4.1 Internal Catalogue

The internal catalogue was held in memory during testing. Loading was only carried out once before testing commenced and the catalogue was never altered and only read once during every iteration. This catalogue consisted entirely of global (super) norms. It was relatively small and the norms contained within were easy to understand. It was anticipated that this catalogue would have far-reaching effects during runtime and would be used on numerous occasions.

Identifier	InCondition	dCondition
LocationSpoofing	once(#SigninActivity) and not(equals(@trigger. grounding.location, @current.grounding .location))	once(#SignOutActivity)
ExposePersonalDetails	once(@start) and once(#SignOutActivity)	once(#SigninActivity)
Login	once(#SigninActivity)	once(#SignOutActivity)
RequireOWLS	TRUE	loaded_function(OWLSAPI)
ProvideRoot	once(@start)	once(@end)

Table 8.5: Catalogue 1

8.7.4.2 Catalogue ID: 2

Catalogue 2 consisted of only hard norms selected from the catalogue outlined in Table 8.3. Roughly half of the norms from the hard norm catalogue were added to this testing catalogue. Norms were selected on the basis of being simpler to process as well as regulating less severe violations than the rest of the norms found in the hard catalogue. In this way, it was intended that this catalogue would help test the handling norms which were easy to both process and impose tough penalties on violation.

Identifier	InCondition	dCondition
RequireSecurity	once(@start) or once(#SignOutActivity)	once(#SigninActivity)
EmptyWorkflow	once(@start)	once(@end)
UnexpectedTermination	once(@start)	once(@end)
Fallback	in(#TransactionActivity)	once(@end)
OverGeneralisedAddItem	once(@start)	once(@end)
RequireShoppingSecurity	once(@start) or once(#SignOutActivity)	once(#SigninActivity)

Table 8.6: Catalogue 2

8.7.4.3 Catalogue ID: 3

The remaining norms contained within the hard catalogue outlined above were norms which were typically both hard and complex. The term complex is used here to identify norms which may take longer or require more resources to process. Additionally the norms within this catalogue had even harsher penalties for violation than the other hard norms in Catalogue 2. This included norms protecting against security intrusions and over-generalisation.

Identifier	InCondition	dCondition
OverGeneralisedSearch	next(#PutInCartActivity)	not(in(#Generalised SearchActivity))
ExtraPayment	once(#PaymentActivity)	FALSE
BookMedical Appointment	once(#FindMedical LocationActivity)	once(#BookMedical AppointmentActivity)
RequireLoginForSecure	inactive(#Login)	active(#Login)
RequireRecieptAfter ExtraPayment	triggered(#ExtraPayment Exception)	once(@end)
RequireEncryption	once(@start) or equals(@current.grounding .protocol,http)	equals(@current.grounding .protocol,https)

Table 8.7: Catalogue 3

8.7.4.4 Catalogue ID: 4

Catalogue 4 featured norms from the soft catalogue outlined in Table 8.4. These norms were less harsh than those in the three previous catalogues. Violation of a norm perceived to be soft was less likely to result in halting of processing and more likely to result in a cost being placed on the execution of any service proposed process. This catalogue contained roughly half of the norms in the original soft norm catalogue. These norms were perceived to be simpler and less strict than the others within this catalogue.

Identifier	InCondition	dCondition
FinalizeOrder	once(#PaymentActivity)	once(#FinalizeOrder Activity)
CreditCardRequirement	once(@start)	once(@end)
FinalOffer	once(#TransactionOffer Activity)	once(#FinaliseOfferActivi ty)
FullLogin	once(@start)	once(@end)
RequestResponsePattern	once(#RequestActivity)	once(#ResponseActivity) or once(@end)
FinalizeAction	next(@end)	once(@end)

Table 8.8: Catalogue 4

8.7.4.5 Catalogue ID: 5

The final catalogue featured the rest of the norms from the soft norm catalogue. These norms had been selected as they have the appearance of being slightly stricter and more complex than those placed in catalogue 4. These norms included tests requiring custom modules and other complex processing requirements. Norms within this catalogue could be violated without serious consequences, however, the actions and conditions regulated may be of greater complexity.

8.8 Conclusion

This chapter and the preceding chapter have shown how the input test cases required for the testing and validation of a norm-based system for service selection can be produced. Because of a lack of viable stores of both norms and suitably complex services, novel approaches for the production of norms and services have had to be engineered. Care was taken at all times during this process to remove as much bias as possible from the end results and to ensure that the final test sets would form a strong base on which to conduct the tests and simulations outlined in Chapter 6. Although not directly answering any of the research questions posed in Section 1.4, the investigations detailed in these chapters provide a clearer understanding as to the construction and underlying reliability of the

Identifier	InCondition	dCondition
OverspecializedSearch	next(#ChoosePaperActivity)	once(@end)
ExtraSecurity	in(#SecurityActivity)	once(@end)
UnsupportedControl	equals(@current.construct, 'Any-Order') or equals(@current.construct, 'Split-Join')	not(equals(@current.construct, 'Any-Order') and equals(@current.construct, 'Split-Join'))
SpellingError	not(custom_function (spelling_en, @current.description))	custom_function(spelling_en, @current.description)
UnsupportedLanguage	not(equals(english, @current.language))	equals(english, @current.language)
OverspecializedProcess	next(#UseItinerary-Activity)	once(#UseItinerary-Activity)

Table 8.9: Catalogue 5

inputs, as well as fulfilling two of the solution requirements laid out in Section 1.4.2; the requirement for a set of norms and services (service descriptions) which can be machine processed. This has also provided an opportunity to refine the underlying API architectures on which the simulation detailed in Chapter 6 relied.

Chapter 9

Testing NPE

Up to this point, this thesis has focused on the investigation of norms, their potential uses in semantically aware computing scenarios, the design of a framework for the norm-based evaluation of process models and the production of unbiased datasets for testing. All of these previous tasks treat the notion of norms and semantically aware normative engines as being purely theoretical. The design of NPE and NPE-L has been left intentionally open so as to make the concepts and ideas formed throughout this thesis available to all and any who might want to form their own implementations. It is the intention of the authors that the sum of the ideas contained within this work be more valuable to the community as a whole than the final conclusions of the following research.

This chapter investigates how the NPE designs can be implemented as a concrete selection system. That system is then reused in a full validation scenario in Chapter 10. The design of NPE and NPE-L has centred around the production of an open and extensible framework with no ties to any one implementing language or structure. As such the options available for the realisation of an NPE module are virtually unlimited. Such openness might be seen as a hindrance, promoting scope creep. This is especially true in a thesis where the main objective is to design a potentially generic framework to handle the representation and processing of norms for use in service selection. It will be shown that the work in this thesis upto this point and the selection system produced in this chapter fulfills the solution requirements in Section 1.4.2, thus enabling an attempt to be made to answer the govering research questions in Section 1.4.

The core NPE module NPE-M was designed and implemented as part of Chapter 5. The core module forms the foundation for any NPE based system; receiving inputs of process descriptions from services and norms, calculating scores for each and returning sets of results. This module is reused in this chapter where it provides key NPE processing abilities and enable the core values behind NPE to be evaluated.

Chapter 6 outlined the testing methodology which was used to take the current proposed norm-based service selection module NPE-M and demonstrate that it answers both

the final research sub-question and the governing research question. In doing so it was shown that all of the solution requirements as stated in Section 1.4.2 had been fulfilled allowing the solution to be evaluated against the governing research question. This chapter will take the provided methodology and use it to formal validate the first two research sub-questions. Additionally this chapter produces a set of base-line results on which Chapter 10 builds to formally answer the final research sub-question.

The first step to proving abilities of the NPE module is prototyping. Prototyping is a common tool used during the debugging and testing of new systems throughout computing. It is a method for proving semantically driven service selection techniques and aids and has been used in a number of recent research efforts (Sycara et al., 1999; Padovitz et al., 2003; Ambrosi et al., 2005).

For the purposes of this research, prototyping was considered to be the testing of the complete NPE module in a manner which enabled the visual checking of progress and results, as well as the ability to rapidly change any associated parameters.

The prototyping procedure took the form of a design cycle focussing on high-level module interdependencies and reuse followed by a period of development. The software development phase of this design followed a spiral development model with successive iterations being altered to improve the original based on testing data. It was envisaged that by the end of this chapter a final version of the NPE system could be completed, including performance and stability testing. This enabled fuller testing to be carried out in following stages of this work where the authors' focus shifted to the performance of NPE in "real world" examples.

9.1 Plan

This stage of the research was planned around the assumption that before full and accurate testing of NPE can be carried out, a working NPE-M based system would have to be produced. The test module had to be of high enough quality and with the ability to operate outside of human intervention, to enable it to be used for any automated testing procedure. It was also critical that any module used for testing had been subjected to rigorous testing cycles in order to reduce and quantify any static performance costs which could have been incurred during execution.

The plan for testing and design of a test system used in this chapter can be found in Chapter 6. This chapter builds on this and improves the suggested framework, ready for full simulations in Chapter 10.

Having selected existing modules and designed any new required sections, the development of the NPE module took the form of an iterative spiral model. This enabled a ridged workflow of: design, build and test to be followed and ensured that the finished module

was thoroughly trailed. The testing procedure used will be different to the experimental procedure used in Chapter 10. Testing was fast and command line based aimed at screening all of the required parts of NPE and NPE-L. This testing did not provide an accurate representation of how NPE will be used, however, it was determined that by using an approach close to that of unit testing in commercial software NPE could be tweaked to improve performance and remove any underlying issues.

9.2 Prototyping

In addition to creating the workflows which enabled the NPE test module to function, a test factory was required to hold test cases and trigger test cycles. For this chapter, the test factory was relatively simple. The control system relied on stores of OWL-S and NPE-L documents (in the form of folders on the underlying file system) which were served to the test module following prescribed test cases. To monitor the test executions, rudimentary logging was implemented as a sink to capture any output or errors. Tests ran sequentially, there was no concurrency involved at this level as since the aim was to measure the performance of a single execution, simultaneous executions would have likely distorted any results. The metrics used to measure the perceived performance of the test module were simple time and load tests measured during the execution by the test factory.

9.3 Testing

The work in this chapter focused on producing a working viable module candidate for in-depth experimentation later on in this thesis. As such the testing referred to in this section was driven not by a need to prove NPE but by the final need to have a reliable working version of a module implementing NPE. Discussions during this testing phase aimed to answer the following questions: what is a good workflow in which NPE can be used? How can inputs and outputs be handled so as to minimise performance costs? And, where might performance gains be found within a module implementing NPE?

The design and production of the NPE module utilised a spiral design pattern (Boehm, 1988), with improvements in subsequent iterations being caused by data from previous ones. This pattern was extended to the testing phase. Each testing run aimed to discover weaknesses in the module and identify issues which would then be solved before the next testing run. As such a new prototype module was created and recorded before testing. Once a version which displayed characteristics in testing making it suitable for full experimentation had been identified, testing ceased and the chosen version was wrapped and prepared for use in Chapter 10.

This general iterative prototyping and testing approach was selected as it enabled the rapid creation of a fully functional module and produces test results and metrics at a very early stage. It was also used as several of the modules present in the module had been used before (in Chapter 7 and Chapter 8), thus there was existing data about the limits and issues surrounding any potential implementation.

9.3.1 Testing Plan

The testing plan for the first round of NPE tests was primarily based around the need to create a stable and reliable final candidate for experimentation. Of note in the above definition is the lack of the terms investigate and compare. During these initial tests, there was no focus on the performance of NPE in a realistic setting, nor was a comparative analysis conducted as to the limits of the system. This takes place in the simulations detailed in Chapter 10. The plan for this initial round of testing was to have a single copy of the NPE test module running within a synthetic environment. The focus was on profiling the module to uncover any bottlenecks and investigate under what circumstances the test module might fail or produce undesirable results.

The test harness for this round was a straight forward Java application running within the Eclipse IDE. The harness was set up to provide the NPE test module with one or a number of NPE-L catalogues and one or a number of OWL-S documents. The test module was then loaded and reasoned over the provided norms and services (as process model descriptions) and returned a result. At this point, the result was ignored and only the presence of a result was recorded, along with the finish time and any data indicating loads at runtime. The input test sets for each run were a subset of the full test set catalogue provided in Section 10.1.3. Numerous tests were aggregated into sets and run automatically by the testing system. Each test run was kept separate and the test module reloaded at the start of every test.

Tests were carried out surrounding the loading process. During testing there were times when tests will be rerun without reloading the test module so that the NPE-L and OWL-S documents previously loaded could be reused without loading being required. This was intended to aid in the investigation of the loading process along with the investigation of NPE when running from input provided as objects.

9.3.2 First Run

The first set of tests involving the NPE test module passed without incident. The module succeeded at every task with no external interference, other than defining the test run and providing input test cases. These were followed by a second run which featured higher loads generated by greater numbers of test cases. Results from both of these tests are details in Table 9.1 and outlined in the following paragraphs.

Test	Complexity	Runs	Mean TTF (sec)	Time SD
1	One simple NPE-L catalogue, one simple OWL-S document.	8	23.54	0.65
2	One large NPE-L catalogue, five complex OWL-S documents.	8	54.72	2.3

Table 9.1: First Round NPE Test Results

Test 1 featured a number of separate runs executed in the Eclipse IDE with a single NPE-L norm catalogue and a single OWL-S document. Both were uncomplicated and relatively short (for discussions of complexity in test cases see Chapter 7 and Chapter 8). The average results can be found in Table 9.1. As can be seen, the runs were relatively quick. However, when the low input loadings are considered these results can be seen to be worrying. The system was working without error at a micro level, however, if scaled the results generated might be of concern.

Test 2 followed on by introducing a greater level of complexity to the test cases used as input. One large catalogue of norms was imported and a set of five more complex OWL-S documents. The results of test 2 can again be seen in Table 9.1. This second run confirmed what had been suspected from the first; that although the NPE test module ran without error, the time to complete could not have scaled. This would not pose a problem for small clients running on enclosed systems, however, it would likely pose problems for systems wishing to operate in dynamic or time constrained conditions. In essence test 2 was designed to test the suitability of the NPE test module to validate the core purpose of NPE; to select a suitable service out of numerous ones provided, based on a set of normative facts. In this core purpose it succeeded. However, due to the time taken this module could be seen to be a failure and not suitable for large-scale testing.

9.3.2.1 Lessons

The primary driving force behind this first set of tests was to learn from any mistakes which have been made and to improve the NPE test module ready for experimentation later in this thesis. From the first two tests lessons must be drawn which can help drive NPE forwards. As the tests were functionally a success, the process required small changes rather than large-scale alterations. All of the issues surrounding the current incarnation of the test module stemmed not from concrete bugs discovered at runtime, but from performance short falls. Using profiling, it was discovered that the majority of the performance issues came from the importing of XML documents, the greedy matching of norms to processes and excessive logging. Of these three the largest performance gains could be sought through improved logging and an improvement to the core NPE module.

Changes could have been made to the importing APIs, however as these rely mostly on external code only small changes to the way libraries are used was made.

The initial NPE logging system sent messages back to the testing system which prints them straight out to the console. The synchronous process was implemented to allow the verbose monitoring of actions undertaken by the test system. However, the constant printing of log messages caused a time penalty and needed to be altered. This was particularly noticeable as the initial test version logs the creation and parsing of every Petri net for debugging purposes.

The second major issue detected during testing was the memory and time consumed by the core NPE-M module whilst matching processes. In this initial version, a Jena knowledge base was queried over SPARQL for each run and every norm was checked during every norm checking cycle. This was a very greedy method for detecting when to activate norms and as such was dropped. Furthermore, it was predicted that performance issues were also have been caused by the excessive looping over facts and activities during runtime. For example, a catalogue of five norms might operated over a single service process definition with five processes (four atomic and one encompassing complex process) needed to make a minimum of 25 queries even before allowing for the activation and deactivation of norms. This figure needed to be reduced (especially so queries could potentially be made to remote sources) if the module was to be of use. It is clear that a degree of internalisation of concepts into objects was needed to improve the performance of the core of the NPE system.

9.3.2.2 Changes

The above lessons were turned into an action plan for improvements before the next round of testing. These changes are outlined in Table 9.2 and roughly fell into two categories: permanent changes and experimental changes. Permanent changes were alterations derived from observations in the previous testing round and were designed to fix shortfalls. For the test harness, the focus was on the production of a quiet mode to reduce logging and implementation of a, SLF4J (Gülcü, 2011) backed asynchronous queued logging system. Both of these aim to reduce the time and resources given over to logging and reporting. Once the new logging system was produced its inclusion was relatively trivial and as such this section was relatively simple. For the core NPE-M module and surrounding knowledge base stores a larger change as needed to improve the handling of norms and processes. As such a small redesign of the NPE-L module has led to the inclusion of parts of the OWLS-API into the core structure to provide objects for the representation of activities. These along with an initial base of norms are loaded at the start of runtime to reduce the number of SPARQL queries to two. These were also be tagged at runtime with matching data as a way of caching results.

Experimental changes at this phase were potential alterations which could be added and removed to test their viability. These included entirely including the OWLS-API and/or the NPEL-API into the core NPE-M module for just in time processing of documents (without storing to a knowledge base). It was decided that these changes would unlikely be included in any final NPE version but were tested never the less for longer term viability.

Category	Module	Description
Permanent	NPE test module	Improvements in logging and inclusion of OWL-S processing code loosely based on the OWLS-API.
Experimental	NPE-M	Bring OWLS-API and NPEL-API on board and process elements in a "just in time" fashion.

Table 9.2: Test Module Changes

9.3.3 Testing Run

The second round of NPE testing focused on measuring the effects of the alterations made in Section 9.3.2.2. These tests used the same harness as the previous round of testing. For Test 3 of 8 (the first run in this set of test runs) the harness used was unchanged to verify that the alterations made to the system did not prevent the NPE from executing successfully. Following runs utilised a modified test system which had a list of catalogues and service descriptions to a single simulation run (as described in Section 10.1.3). This automated process aimed to enable the NPE test module to be utilised in an environment more closely resembling that which it would encounter during testing. Testing was automated by a simple system holding lists of test cases with references to folders containing XML files. The performance of this system was excluded from the profiling information provided by the Eclipse IDE. As this chapter focusses on the performance of the proposed test module and not on the results generated, the output from each test run was discarded. Logging which caused a great deal of performance degradation during the previous round of testing was altered to the system outlined in Section 9.3.2.2. As such, results were no longer available in real-time. Results were written to a flat file as a report following the completion of all testing runs. There was a total of four test runs in this second round of tests, one simple manual test and three automated tests as described above. The final set of automated tests (Test 6 of 8) focused on deploying some of the experimental features outlined above. This included the ability to handle OWL-S documents in a just in time fashion. As these features were unlikely to be included in the final version of the module (used for simulations) this round of testing was given lower significance and reserved to the last test once a clear improvement had been seen in the host module.

It was decided if changes to the NPE test module were needed during this round of testing, for example, if a new point of performance deterioration was identified, the NPE module would be remade and retested. Any alterations made in this fashion were recorded and are detailed in the following results and outcomes sections.

9.3.3.1 Results

From the first run in this new round of tests, the performance gain over previous test was perceivable. Even during simple manual testing, each run time was significantly shorter and in places required fewer resources. Initial load times were reduced thanks to a reduction in erroneous log entries. However, logging still remained a large part of the total runtime for a single test. The initial resources required for loading and maintaining the OWL-S and NPE-L APIs also remained relatively high. This was most probably caused by the requirement of each to load Jena for processing and storage of concepts. It is likely that future versions of NPE might be able to improve on this by utilising a single shared knowledge base loaded once during initialisation. The core NPE-M module required relatively the same resources for this set of test runs. However, the time taken for each service description to be processed dropped significantly. This was most the most notable change in performance between this set of tests and the last. It is likely that this change was attributable to the internalisation of process concepts previously stored in an external knowledge base, along with a reduction in the amount of logging produced during matching. Table 9.3 presents the final average results for each test run. As can be seen Test 3 of 8 can be equated with the previous test 1 and 2 and shows a favourable improvement in both resource use and in the time taken to complete. Tests 4 and 5 give the average final result for eight runs of five automated tests.

Test 6 of 8 was different from the previous tests as new structures and processes were included to test experimental features. The most notable of all of these is the inclusion of the OWLS-API into the NPE-M core to enable just in time processing of OWL-S documents. This trial was a success in that the module still provided reliable output and for single runs the module was slightly faster and consumed fewer resources (thanks to the removal of the knowledge base storing processes). However, where more than one service description needed to be processed the required resources jumped as did the time to complete. This suggests that this alteration might be suitable for environments requiring simple processing (where a service process description might fail early on in processing) with minimal resources.

9.3.3.2 Outcomes

The primary outcome from the second set of testing is that sensible logging is clearly a good idea, allowing properly designed code to operate without constant verification

Test	Complexity	Runs	Mean TTF (sec)	Time SD
3	One simple NPE-L catalog, one simple OWL-S document.	8	12.85	0.3
4	One large NPE-L catalog, five complex OWL-S documents.	8x5	22.35	0.7
5	One large NPE-L catalog, five complex OWL-S documents.	8x5	24.85	0.9
6	One large NPE-L catalog, five complex OWL-S documents.	8	40.24	1.2

Table 9.3: Second Round NPE Test results

by a human source is preferable to repeated checks. The logging systems devised for the NPE test module operated well with that of the component modules and enabled the validation of individual steps within the system. For each full testing run (where results are displayed in Table 9.3) the added quiet mode was employed to further reduce the amount of logging produced offering greater performance benefits. Furthermore, this testing has enabled the production of a more reliable faster final NPE test system. The system demonstrated in Test 4 and 5 had the potential to be used in any final experimentation system, as even scaling to large numbers of concurrent tests on a single platform should be possible.

The outcome from the experimental changed was that although they showed promising results for certain circumstances, changes caused greater performance losses than gains. As such these experimental options would be dropped and were deemed to be out of the scope of this research. Future work potentially based on these changes can be found in Chapter 11.

9.4 Conclusions

This chapter has detailed the testing of an NPE based service selection module suitable for use in wider experimentation. The NPE-M core module created in Chapter 5 was tested and improved throughout this chapter so that research could be conducted as to the limits of the system without undue interference from performance constraints posed by unchecked code.

The choice of a modular design paradigm backed by a test based development and evaluation structure was a good one. This solution enabled the rapid production of a working NPE test module without having to conduct unnecessary full runs with an experimental styled system. This modular design paradigm also enabled the rapid modification of the test system to implement improvements.

The current incarnation of the NPE test module was taken forward to detailed simulation in Chapter 10. Following Test 5 of 8, results were shown to have greatly improved on previous versions, making NPE potentially suitable for large scale concurrent experimentation. This module (use in Test 4 and 5, see Table 9.3), features an improved logging system along with a modified NPE-M core component. Adjustments included changes to the NPE-M module incorporating parts of the OWLS-API to improve processing speeds. Such an inclusion works for experimentation and testing with OWL-S based process models. However, it is likely that this will be replaced with a generic object set for future versions after this thesis. More can be found on these future changes in Chapter 11.

This chapter consolidates the work undertaken in this thesis so far. It has presented the first concrete implementation of a complete NPE based system, bringing together the NPE model (Chapter 3), NPE-L ontology (Chapter 4), NPE-M module (Chapter 5) and research sub-questions of creating a system for the reasoning of service descriptions with regards to formative facts and fabricating a test environment for the proving of the primary goal (Section 1.4). This thesis submits that work conducted up to this point demonstrates that a system for the reasoning over service descriptions with regards to held normative beliefs can be produced and that such a system fulfills the solution requirements laid out in Section 1.4.2 and thus is suitable for use in simulation to answer the primary research question.

The remainder of this thesis uses the module created here and the lessons learned during creation as a basis for full experimentation. Proceeding chapters investigate the limits of use of this example NPE based system when it is used in situations which attempt to accurately reproduce predicted usage patterns.

Chapter 10

Validating the Limits of NPE

Throughout this thesis, reference has been made to the use of service selection techniques in agent-based systems both on and off the World Wide Web and Semantic Web. It has been acknowledged that service selection is a topic which, in a culture of increasing automation, is growing in importance. This chapter focuses on testing the limits of NPE in an environment representative of that which might be expected in commercial deployment. Here it has been decided that the deployment scenario most likely to utilise an NPE based selection system would be an agent based system. Thanks to the open nature and design of the NPE and NPE-L specifications, the options available for implementation are numerous. This chapter investigates avenues of research towards a suitable implementation plan. However, as this thesis is primarily interested in the creation and testing of a norm-based system for partner selection, this chapter should not be viewed as a survey of potential agent frameworks. Numerous papers exist critiquing available agent systems; these are quoted throughout this chapter and in the literature review conducted previously. The reader is urged to consult these works for an in-depth discussion of potential agent approaches.

There is currently no single agreed testing solution for service selection on the Semantic Web. Research efforts looking into purely mathematical solutions often rely on complex simulations to prove their worth. Projects working with semantic solutions, or solutions based on matching do not adhere to such testing routines. This disparity stems from the distinctions between the two approaches. As this research falls into the second category, a set of exclusively mathematical simulations will not suffice for testing. For this reason, testing of the NPE module was split into two sections: testing and simulations. Testing has been carried out previously in Chapter 9. This chapter focuses on simulations as a means to validate the governing research question.

At the start of this thesis, a governing research question of "How can an actor with a set of known normative beliefs use these beliefs to aid service selection where IOPE matching typically falls short?". The governing research question has since been supported by a

further three research sub-questions surrounding the creation, implementation and testing of a norm-based approach for service selection. The combination of the work carried out to this point has resulted in the first three research sub-questions being successfully answered. Chapter 3 outlined the creation of a model for norms which could be used in service selection, Chapter 4 created an ontology for the representation of norms in semantically aware actors, Chapter 5 proposed the module NPE-M for the reasoning over service description with regards to normative facts and Chapters 6, 7, 8 and 9 demonstrated a concrete implementation of a system for the selection of services based on normative beliefs can be created. This chapter employs simulations to demonstrate that the solution (NPE) proposed in this thesis answers the governing research question.

10.1 Planning

The testing methodology outlined in Chapter 6 formed the basis for the validations carried out in this chapter. It built on this using methods taken from formal experimental design to effect a full and reliable validation of the ideas proposed in this thesis. The agent-based validation methodology was undertaken with variables controlled using the latin squares based protocol outlined in Section 10.1.3. This along with the information gained in Chapter 9 ensured that the goal of answering the final research sub-question can be reached and thus the governing research question answered.

10.1.1 Existing System

As has already been explained, the module perfected in the previous chapter is reused in this. The plan for this thesis was to build a normative framework for service selection, step by step showing at every decision why a specific approach has been adopted. Thus, the production of the NPE test module and simulation environment were separated. The existing system was tested independently, the result of which can be found in Table 9.1 and Table 9.2. This final round of simulations did not modify the structure of this existing module beyond any initial optimisations listed in the following subsection. Any changes made were for the purpose of aiding the inclusion of NPE in any testing system and not to affect the core operation of the existing modules.

10.1.2 Modifications To Existing System

Modifications were carried out as part of an optimisation exercise in Chapter 9. Before the NPE test module could be used for full simulation detailed in this chapter it was necessary to be modified further. These modifications were for the purpose of preparing the NPE module for inclusion into an agent-based computation environment. As such

they centred around the removal of any of the previous test harness code, further reduction of extraneous logging and simplification of access to the module through a well-defined interface. In addition to these changes a new configurable output module was included to handle the production of well-formatted results objects, thereby enabling the host agent to view how the NPE module has rated all available process definitions. From this information, a selection was made. This requires a slight modification to the NPE system since it now passed responsibility for the service selection to the host. The decision to alter responsibility for final selection was taken to enable the true use of NPE as a service rating module to be exposed.

The full NPE module simulation design was laid out in Chapter 6. This included the NPE-M module in situ within one of any number of agents operating in a simulated environment. The reader is encouraged to refer back to Chapter 6 and Section 6.4.2 for further information as to the technical details behind these simulations.

10.1.3 Protocol

Definition 21. $DESCRIPTIONS = \{INFO, SERVICE, PURCHASE\}$

$TESTCASE_n = DESCRIPTION_n \times RATING_n$
$(RATING_1 < RATING_5) \wedge (RATING_6 < RATING_{10}) \wedge (RATING_{11} < RATING_{15})$
$RATING_n \approx RATING_{n+5}$

$INFO = \{TESTCASE_1 \ldots TESTCASE_5\}$
$SERVICE = \{TESTCASE_6 \ldots TESTCASE_{10}\}$
$PURCHASE = \{TESTCASE_{11} \ldots TESTCASE_{15}\}$

ID	Description
A	1,2,3,4
B	1,2,3,5
C	1,2,4,5
D	1,3,4,5
E	2,3,4,5

Table 10.1: Description Test Sets

		Length	
		Short	Long
Severity	Soft	α	β
	Hard	γ	ε

Table 10.2: Norm Catalog Assignments

Sound results can only be gained from experiments which are based on a sound design. The questions posed above were answered not by experimentation, but by simulation. However, the differences between the two processes does not diminish the importance of good design. For the results of any simulation to be reliable, the design must be thorough, fair and repeatable. To achieve this, it was possible to rely on concepts and processes from the field of experimental design.

Before any design could be started, it was necessary to define the inputs, outputs and variables required by the simulations. This step was integral to the design process as subjects, variables and constants form the core of any simulation. In the case of these simulations, the subjects involved were all agents operating within a restricted universe. The details of this set-up can be found in the section above (Section 6.4.2.2). For every simulation run, there were be five agents operating within the simulated container (excluding the administration agent). One control (plain) agent who had no NPE module (*Agent*0) and selected services using basic input-output matching and four NPE enabled agents. The four NPE agents were labeled *Agent*1, *Agent*2, *Agent*3, *Agent*4. Details of these agents can be found in Section 6.4.3.

Passed to each agent "subject" are a selection of test cases and collections of norms. Test cases are defined in Definition 21. A single test case is a combination of an OWL-S description and a score. The higher a descriptions score, the more preferable it was for potential interaction. There were 30 primary test cases in total split into three categories. These categories were chosen as three areas for Web service use which might be of interest. Each iteration governs one category only. Categories were identified to agents using the NAICS classification system and the OWL-S documents were those created in Chapter 7. During each run an agent was passed a specific number of test cases from which they had to choose one (which in the "real world" would be the one they would interact with).

As well as test cases containing descriptions, sets of norms were needed (for all but the control agent). The five initial catalogues created in Chapter 8 were mixed, so that a "hard" catalogue always matched with a "soft" catalogue, producing five catalogues which affect more of the test cases and would improve final results. These five collections of norms were provided within the NPE simulation environment. One set was held internally within each agent and four external sources one of which was passed to each agent at runtime. These four external catalogues were split into two main categories (Table 10.2): hard and soft. Within each of these categories, there was a long and a short catalogue of norms. These four catalogues were intended to test both the impact of parsing variable sized catalogues on the performance of an agent and also the impact of service selection when there were more or less, harder or softer norms. The single internal norm catalogue was the same within all of the agents (except the norm-free control agent) and represented a set of "super-norms".

A single test within a single simulation run was a combination of a subject (agent) provided with a single norm catalog and a combination of test cases. There were five combinations of test cases; a breakdown of which can be found in Table 10.1. These combinations were designed to test NPE's ability to handle comparing very similar descriptions. A set of balanced block diagrams for the first few runs outlining the test case assignments can be found in Section 10.1.3.2. Rather than run every combination in order or pick ones at random, a solution based on the experimental design theory of Latin squares was produced.

10.1.3.1 Latin Square Design

In experimental design, sources of variables may be referred to as blocking values. Blocks refer to the arrangement of a set of experiments into a single group or "run". The key to good experimental design is to design these blocks so as to enable a fair representation of the blocking factors to be captured (Mead, 1990). Using careful design, it is possible to simplify the blocks required by either eliminating blocks, or balancing any design. The simulations suggested in this phase of the thesis rely on multiple blocking variables. This complicated design further as it was not feasible to test every combination of variable and subject. A purely random approach was also insufficient. Therefore, it was decided that the experimental design technique of Latin and Graeco-Latin squares be used (Box, 1978).

Ignoring the blocking variable "norm catalogues", enabled the design to be limited to a Latin square for the first step. Latin squares are extensions of randomised design with blocking. A Latin square can be defined as a grid with t^2 units arranged in a double blocking system with t blocks in each system and t treatments occurring in each block. In the proposed system, this meant that a 5x5 Latin square was produced with five runs (down the y axis), five agents (along the x axis) and five combinations of four of the five test cases distributed evenly. This approach was relatively restrictive, but, it could be improved by repeating simulations using a number of different Latin square combinations chosen at random.

For the actual simulations to test NPE, an extra variable was needed on top of those present in the Latin square above. The simulations provided each agent with one internal and four external catalogues of norms, these were randomised and added to the design. To do this a Graeco-Latin square was used. With a Graeco-Latin square, the extra variable is mapped on top of the existing design using an extra separate Latin square (effectively interpolating two Latin squares). This enables the fair testing of all variables in a limited number of tests. Again there is an element to which this approach is restrictive, to overcome this multiple simulations should be used with random combinations of two Latin squares to give random Graeco-Latin squares. The Graeco-Latin square for the first simulation run can be found in Table 10.3.

		Agent				
		0	1	2	3	4
Run	R_1	$A\alpha\varepsilon$	$B\beta\varepsilon$	$C\gamma\alpha$	$D\delta$	$E\varepsilon\beta$
	R_2	$B\gamma\alpha$	$C\delta$	$D\varepsilon\beta$	$E\alpha\varepsilon$	$A\beta\varepsilon$
	R_3	$C\varepsilon\beta$	$D\alpha\varepsilon$	$E\beta\varepsilon$	$A\gamma\alpha$	$B\delta$
	R_4	$D\beta\varepsilon$	$E\gamma\alpha$	$A\delta$	$B\varepsilon\beta$	$C\alpha\varepsilon$
	R_5	$E\delta$	$A\varepsilon\beta$	$B\alpha\varepsilon$	$C\beta\varepsilon$	$D\gamma\alpha$

Table 10.3: Graeco-Latin Square for First Iteration

10.1.3.2 Balanced Block Diagrams for Test Case Selection

Balanced block design is a form of experimental design procedure, which can be used in addition to the Graeco-Latin squares illustrated in Figure 10.3. The motivation behind balanced block design is to help in the removal of unwanted blocking variables from an experimental scenario. This can be achieved by planning a protocol for experimentation by arranging similar variables together and utilising test runs whereby variations created by an unwanted extra variable are excluded. The design for the NPE simulations was backed up by balanced block techniques. This was used in conjunction with the Latin Squares designs in Section 10.1.3.1 to reinforce the choice of protocol. For NPE simulations, the balanced block designed in Table 10.4, Table 10.5, Table 10.6, Table 10.7 and Table 10.8 were created. These represented the first five runs for each agent, showing the test case selection for each. The blocking variable "norm catalogue" was removed for this design as it was for the initial stages of Latin Squares design. This enabled a focus on ensuring that the dispersion of the test case variable, then the blocked norm catalogue variable, could be added later and factored in using multiple runs (as in Table 10.3). This design also helped in the tempering of the blocking variable created by variations in the underlying simulation system (fluctuating system resource availably for example).

		Test Case				
		1	2	3	4	5
Run	R_1	X	X	X	X	
	R_2	X	X	X		X
	R_3	X	X		X	X
	R_4	X		X	X	X
	R_5		X	X	X	X

Table 10.4: Balanced Block Diagram for Agent 0

		Test Case				
		1	2	3	4	5
Run	R_1	X	X	X		X
	R_2	X	X		X	X
	R_3	X		X	X	X
	R_4		X	X	X	X
	R_5	X	X	X	X	

Table 10.5: Block Diagram for Agent 1

		Test Case				
		1	2	3	4	5
Run	R_1	X	X		X	X
	R_2	X		X	X	X
	R_3		X	X	X	X
	R_4	X	X	X	X	
	R_5	X	X	X		X

Table 10.6: Balanced Block Diagram for Agent 2

		Test Case				
		1	2	3	4	5
Run	R_1	X		X	X	X
	R_2		X	X	X	X
	R_3	X	X	X	X	
	R_4	X	X	X		X
	R_5	X	X		X	X

Table 10.7: Block Diagram for Agent 3

10.1.4 Risks

Undoubtedly the largest single risk to the simulations required during the testing of this system was the introduction of bias into the protocol. This was most likely to be caused

		Test Case				
		1	2	3	4	5
Run	R_1		X	X	X	X
	R_2	X	X	X	X	
	R_3	X	X	X		X
	R_4	X	X		X	X
	R_5	X		X	X	X

Table 10.8: Block Diagram for Agent 4

by a non-automated or human based variable introduced during design. An analysis was conducted into the probable sources for bias in the following simulations. Of these, bias inherit in the NPE-M core module was recorded and accepted as design flaws or bugs. Of most concern was bias introduced into the test cases during generation. This included both the OWL-S service test cases and the NPE-L norm catalogue test cases. During the design of these variables and the construction of the simulation system, efforts were made to reduce potential levels of bias. One of the main issues surrounding simulating service selection for services describing themselves using OWL-S was the lack of test cases (this is true for services using other description languages as well). A few do exist, most notably the OWLS-TC collection (Klusch et al., 2008). A good test collection is necessary to ensure that the simulations produce fair results. Chapter 7 and Chapter 8 recorded the design decisions behind the creation of both service and norm-based test cases. During these processes, the emphasis was on the reduction of bias through the use of unattended and automated tools. The final collections had levels of bias which were low enough for meaningful results to be collected. To further prevent the undue influence of bias on the outcome of the simulations, this phase outlined the experimental design methodologies which created a fair protocol for testing.

Reliability in the context of these simulations was a measure of how well, the agents could rank services and choose a preferable one. Simulations in this phase aimed to test which services each agent would choose (if they managed to choose one at all) and how suitable that selection was. These simulations were run using the method outlined above. Ranking was compared to ranking scores given in the test case. These figures were not available to the agents whilst they were selecting services; instead they were used to evaluate any selections during analysis. This is a further point where bias could occur. Again these figures were produced as part of the automated processes conducted in Chapter 7 and Chapter 8 with the intention that the same protections which prevented undue bias within the test cases would also apply to the metadata which described them.

10.2 Results

The simulations described in the Chapter 6 and protocol above were executed in full. A complete simulation consisted of six iterations with a complete iteration consisting of five test runs. Each iteration was expected to yield acceptable results on it own based on the protocol created in Section 10.1.3. Full simulations were also executed to provide a complete set of results and each simulation was replicated a further five times to complete the testing. As a result of protocol design and speed of execution, a total of six full simulations were completed allowing averages to be taken, further removing variation caused by underlying system loads. All simulations competed successfully. Results were collected by the administration agent and tabulated post-execution in a spreadsheet program. The following sections describe the simulations, any issues that occurred and give a full set of results. It should be noted that all results are exclusive of loading and resources consumed by the Jason container and underlying Java virtual machine.

10.2.1 Initial Simulation

A first simulation termed "Simulation 0" was conducted as an initial check to validate the simulation framework. This utilised all of the methodologies created Chapter 6 and followed the testing protocol laid out in Section 10.1.3. A single simulation was conducted with six iterations of the Graeco-Latin square described in Table 10.3. For each iteration, a different service test case was used as an input for the agents. Thus over six iterations each test case was fully explored (as there were three categories with two test cases per category). Results were collected in the manner outlined previously in this chapter and were tabulated after all six iterations had finished. Furthermore, as the purpose of this first simulation was to test the processes used for all simulations, results collected during execution were fully analysed on completion. This is in contrast to later simulations which were executed in quick succession with a long period of analysis and reflection after completion of the final run.

Simulation 0 completed successfully. At this point, this result helps to demonstrate the worth of the NPE system it also helped to validate the simulation framework as, being capable of producing consistent results. Of the lessons learned from this first round of simulations, it was found that as expected Agent 0 presented suitable baseline results as an agent with only basic selection criteria by finishing the fastest. Agent 1 employing both the NPE system and IO matching took the longest and provided an interesting insight into the likely outcomes if NPE were to be used with an additional matching algorithm, randomly choosing "winning" service descriptions when multiple descriptions are provided by NPE produces results which feature an amount of "noise" but are not so degraded as to warrant removal of this method. Results for this Simulation can be found in full in Table 2 in the Appendix at the end of this thesis.

10.2.2 Further Simulations

The Latin squares design detailed in Table 10.3 allowed each set of service test cases to be simulated only once with the norm catalogues being rotated using an overlaid Latin square. As the first simulation (Simulation 0) took only 17 minutes to complete it was decided that although there was no requirement to repeat a full simulation, the full simulation was to be rerun and repeated a further four times. This not only enabled every set of test cases to be run, it also enabled the Latin square detailing norm catalogue assignments to be rotated a full rotation. Thus, every combination of test case and catalogue was tested. The completed set of simulations was then completed by experimental design and complete by exploring all combinations of inputs. Five full simulations were completed in this final phase of exploration. The complete set of simulations took nearly 85 minutes to complete with each iteration having to wait for the last agent before finishing. In total 150 runs were recorded.

10.2.3 Final Results

The final simulations are summarised in Table 10.9 and Table 10.10, full results can be found in Appendix A.5.7 (at the end of this thesis). The results were compiled from logs saved after each iteration and exclude time to load (or reload) any underlying frameworks. These results are the times taken to finish and resources used by each matching run with any pre and post iteration processing removed. Of particular interest are the means and standard deviations for each. These demonstrate the potential variation and realistically achievable times for each. The final set of scores in each table is the result of the complete matching algorithm. In these scores a score of 0 depicts that the "best" service description was selected, a score of 4 depicts the "worst" was selected and a score of 5 depicts that no description was selected. Ideally service selection would result in the production of scores of 0 with the occasional 1. With the testing protocol as outlined above, it was impossible for all 0s to be achieved as one of the five test case sets does not contain the "best" description; in such cases for statistical analysis of the precision and recall, 1 was taken as the "correct" result.

Agent	Mean	SD	SEM	R^2
0	12.64	1.99	0.16	0.0113
1	27.10	4.23	0.35	0.0034
2	22.00	2.76	0.23	0.0034
3	19.48	3.07	0.25	0.0112
4	18.58	2.80	0.23	0.0012
Means	19.96	2.97	0.24	0.0061

Table 10.9: Results Summary (Time to Finish)

Agent	Mean	SD	SEM	R^2
0	2.03	1.45	0.12	0.0379
1	0.97	1.32	0.11	0.0001
2	0.99	1.22	0.10	0.0000
3	0.95	1.24	0.10	0.0001
4	1.71	2.04	0.17	0.0087
Means	1.33	1.45	0.12	0.0094

Table 10.10: Results Summary (Scores)

Figure 10.3 presents the time to finish results for all agents. In this view, it is possible to see that each agent appears to have its own general "height" (the average of which can be seen in Table 10.9) in the results. It is possible to see that in general Agent 0 took the shortest time to complete each test run and Agent 1 took the longest (as had previously been observed in Simulation 0). In this view, it is also possible to make out some small localised patterns in the data. Trends can be seen in clearer detail when connecting runs with lines and enlarge, however, this requires more space than is possible in this thesis. Variations occur in regular patterns showing as flowing peaks and troughs. Further investigations found that these were most probably caused by the processing of simpler or more complex service descriptions and norm catalogues. Local variations such as these were likely to be caused by a combination of increased initial loading times and increased processing times as more elements had to be considered by the NPE-M module. However, as the standard deviations for each agent in any given simulation are relatively small, the effects of these local variations are small. When considering standard deviations in processing time, of more interest is the larger figure for Agent 1. As this test subject had greater overall restrictions on the test cases being processed (as it was evaluating test cases using both IO matching and NPE) this manifests as large changes between the extra processing being a help and a hindrance. A more extensive look into processing times is conducted later in this section when means and standard deviations are revisited.

	Agent 0	Agent 1	Agent 2	Agent 3	Agent 4
Relevant retrieved	43	93	89	96	80
Retrieved	150	150	150	150	118
Relevant	150	150	150	150	150
Precision	0.29	0.62	0.59	0.64	0.68
Recall	0.29	0.62	0.59	0.64	0.53
F1	0.29	0.62	0.59	0.64	0.60
$R(time, score)$	0.15	0.29	0.22	0.33	0.34

Table 10.11: Precision, Recall and Collelation

The scores collected by the administrator agent at the end of each run gave an approximation of how "accurate" each agent was at choosing a service based on a set of norms (or requirements). Figure 10.1 presents the frequency of each score for each agent over all

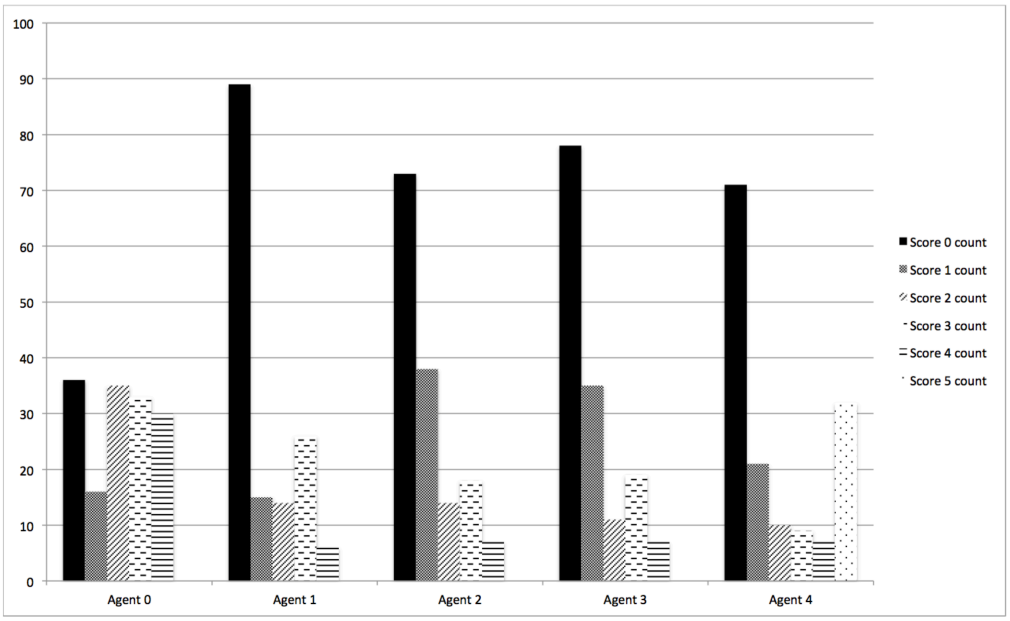

Figure 10.1: Final Results (Score counts)

runs. In the case of scores, frequencies demonstrate a better picture than total results (as used in Figure 10.3). The scores showed that on the whole the three actors implementing NPE as a sole mechanism for service selection performed reasonably the same. This was expected as the simulations conducted covered the full combination of test cases and catalogues. Any variation between the two "Basic" NPE agents (Agent 2 and Agent 3) will have been caused by "noise" introduced during final selection. A lower count of 0 and 1 score services was returned by Agent 4 which might be surprising as this actor was utilising the strictest approach to NPE. However, the drop in 0 and 1 scored services can be accounted for by the large number of 5 scores received. Only Agent 4 produced a score of 5 as this represented a complete fail in finding a suitable service; an outcome which could only be achieved when all of the services processed by NPE violated norms to a cost above a set threshold resulting in processing being stopped. In this case, a number of runs undertaken by Agent 4 resulted in no service being chosen.

Figure 10.2 shows the mean±SEM (standard error) and the mean±CI (confidence interval). Overlapping SEM and CI bars are a fair graphical indication of significance (Payton et al., 2004). Looking at Figure 10.2 suggests that there may be no difference in the scores picked by the three agents utilising NPE without being strict. All three display a difference between agent 0 (using random selection) and agent 4 (who's score is affected by a large number of 5's denoting non-selection of services due to multiple violations by all options). There is some suggestion that agent 0 and agent 4 may not be statistically different (i.e. that an agent picking at random may be as accurate as an agent which relies on norms to such an extent that they fail to match services). However, removing the scores of five from agent 4 changes this suggesting that in such an example

an implementing agent would likely fall back on random selection after failing to find a match wth NPE (or would gain from the protection NPE offers and thus a score of five is not representative of the benefit to the agent).

It worth explaining the lack of 1 score services selected by Agent 0 (the baseline actor undertaking basic IO matching). This was likely due to variations in the test case generation process causing minor specialisations (or generalisations) in the top level activity for 1 score services and thus causing the IO matching to succeed in not selecting such service description. This was not seen as an issue, but a validation that the IO matching approach taken works as expected and in some cases out-performed NPE where only slight changes were detected.

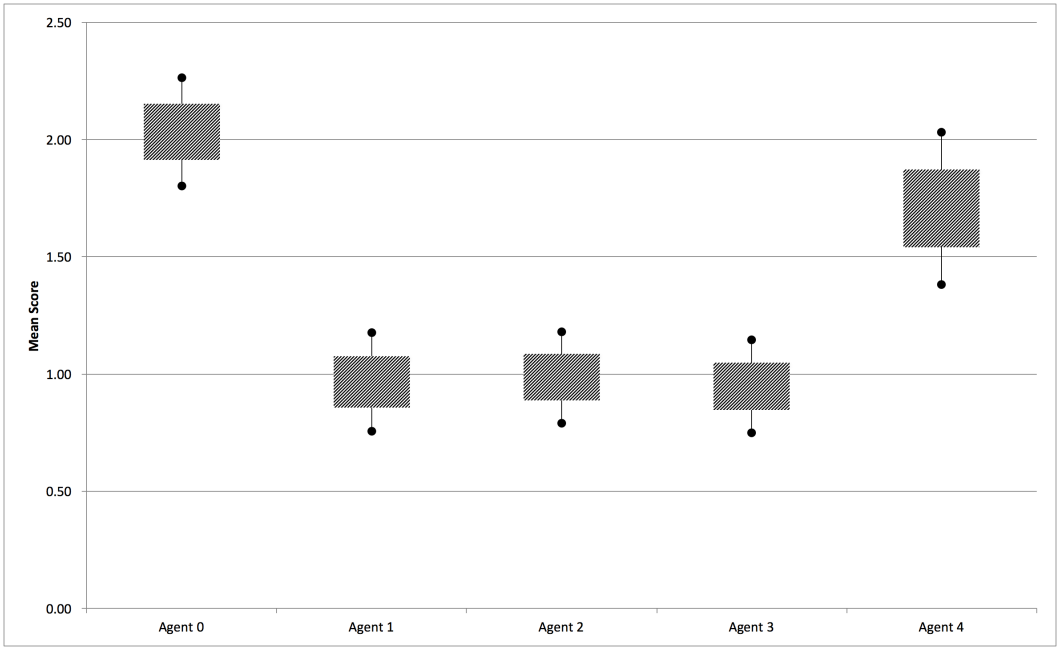

Figure 10.2: Score Variations (SEM and Confidence Interval)

The results from Figure 10.1 are complemented by the calculations in Table 10.11. These figures for precision and recall underline the improvement which can be gained by utilising NPE in the selection of service descriptions. As the first 4 agents were all guaranteed to return a selected service, the precision and recall were the same (as the number of relevant services is 150 and the number of retrieved services is 150). From Figure 10.1 it would seem that the most reliable actor was Agent 1 (with the most 0 score service descriptions selected), however, these calculations demonstrate that Agent 3 was slightly better. This discrepancy was caused by the IO matching (which further improves the selection of Agent 1) causing some perfectly acceptable 1 score service descriptions to be discounted. In tests where score 1 services were the "correct" choice (there is no "perfect" solution available) the inflexibility of the IO matching resulted in an incorrect service description being selected and Agent 1 scoring lower (or higher in real terms). It is also worth noting that Agent 4 scored almost the same in terms of reliability because the poor

recall (as picked out from Figure 10.1) was offset by a much higher precision, meaning that when Agent 4 selected a service it usually provided a "higher quality" result. In addition to this an improving collection can be seen between the best possible score avaliable to an agent and the chance that the agent will select this score, as the level of adherence to the NPE result increases.

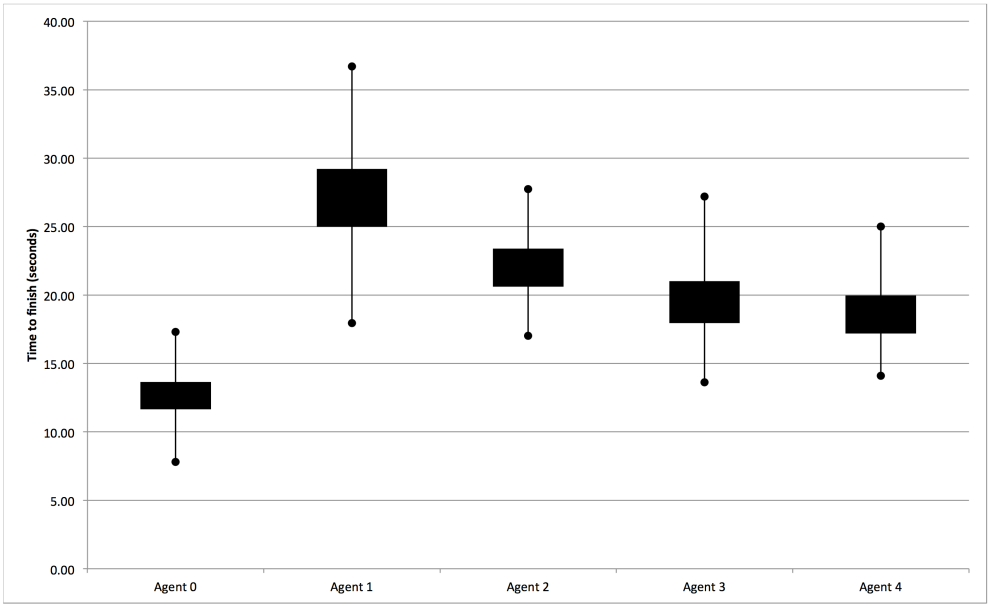

Figure 10.3: Timing Variations (Mean, SD, High and Low)

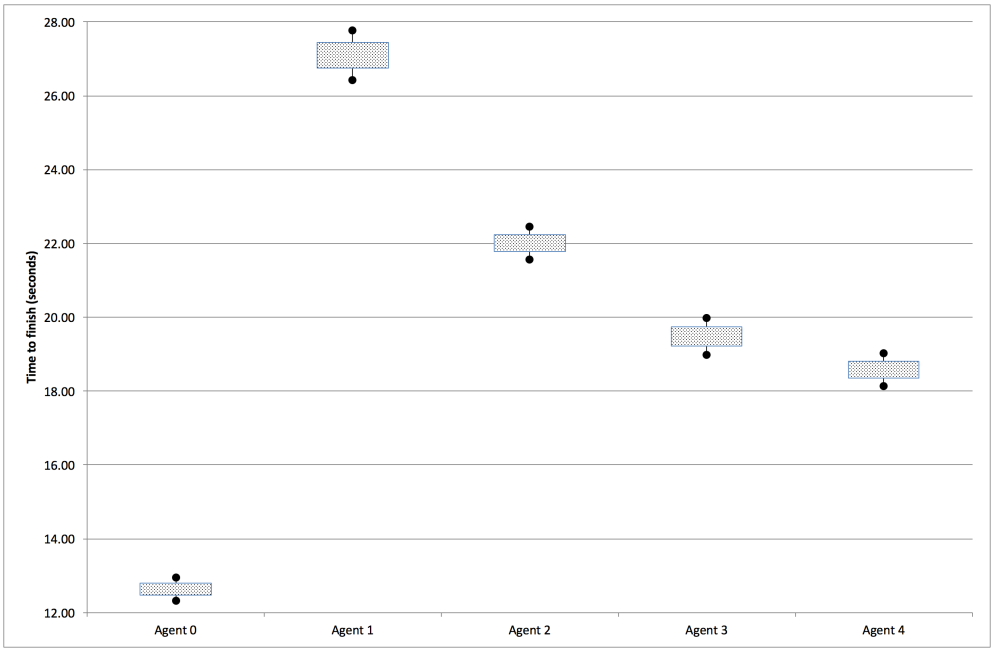

Figure 10.4: Timing Variations (SEM and Confidence Interval)

Figure 10.3 presents the results from Table 10.9 as highs and lows around the standard deviation and mean. These best demonstrate the trends in time to finish with regards to

each agent. From this, it can easily be seen that NPE based agents take up to twice as long to complete a test run as the basic IO matching Agent 0. This is to be expected as there are significant overheads involved in the modelling of process definitions, loading and processing of norms. Of the three NPE only agents, Agent 2 can be seen to take slightly longer. This resulted from the implementation of NPE contained within not halting on unrecoverable violations and so every process model completely explored. Compared to Agent 2, Agent 3 had a lower mean run-time, but larger variation in run-times. This is simply due to this implementation of NPE stopping processing on unrecoverable violations and thus spending less time on certain descriptions. Agent 4 featured a version of NPE which halted if the cost of any perceived violations reached a set limit. This had a small effect over Agent 3 (which only stops on unrecoverable violations) as seen by a lower mean. Finally, preprocessing services descriptions using IO matching by Agent 1 caused the times taken to not only be the highest, but also the most widely spread. This was caused by the initial matching algorithm cutting out some of the service description which would otherwise have been passed to NPE for processing. In such circumstances, this successfully reduced the processing time for Agent 0. However, where IO matching was unable to decide between services, the effect was to cause the total time to rise.

Figure 10.4 shows the mean±SEM (standard error) and the mean±CI (confidence interval). As has been stated previously overlapping SEM and CI bars are a fair graphical indication of significance (Payton et al., 2004). Looking at the run times, it can be seen that the results are likely all to be different, with exception of agent 3 and 4 who demonstrate a degree of similarity (although this similarity is not conclusive).

10.3 Statistical Analysis

With 750 individual test runs (split across 5 agents), the data set gathered from simulations of NPE is large enough to warrent analysis using advanced techniques. In particular it is key to answering the research questions layed out in Section 1.4 that the relationships between the results from various agents, catalogs of norms and test cases is explored. To achive this all 750 test runs were separated out, and added into a single dataset with the run results for each agent as a new entry. This set was then fed into SPSS (Field, 2013) where it was analysed, using a number of techniques. When reviewing the results of these tests, it should be noted that a p value of .000 should be read as $p < .001$, as SPSS reduces all values less than .001 to .000.

10.3.1 One Way ANOVA

For the first set of analysis two One Way Anova (Tabachnick et al., 2001) tests were carried out on the scores and times from each agent. As initial tests showed the homogeneity

of values to be less than the chosen significance value of 0.05, this signifies that the standard deviations for each of the sets of data are significantly different and so ANOVA cannot be reliably used. The post hoc test chosen was the Games-Howell test (Ruxton and Beauchamp, 2008; Stoline, 1981), not Tukey as might be expected if the result of the Leven's test was greater than 0.05.

The Brown-Forsythe (Brown and Forsythe, 1974) (and Welch) test for the equity of means were used to confirm that there is a significant difference between the groups of mean rather than the ANOVA test for analysis of both time and score. These tests are used rather than the ANOVA, as the Leven's test resolves to $p < 0.05$. As the Brown-Forsythe test passed, it is possible to use the results from the Games-Howell test to look at the results gained from comparing the times and scores of various agents.

A summary of the result of the Games-Howell test for the runtimes of each agent can be found in Table 10.12. Remembering that .000 should be read as $p < .001$, it is easy to see that the run times for the majority of the agents are statisticaly significantly different. This is with the exception of agents 3 and 4 where $p = .065$ is greater than .05 and it can be assumed that the means of agents 3 and 4 with respect to runtime per-test are not statisticaly different.

	Agent				
	0	1	2	3	4
0		.000	.000	.000	.000
1	.000		.000	.000	.000
2	.000	.000		.000	.000
3	.000	.000	.000		.062
4	.000	.000	.000	.062	

Table 10.12: Games-Howell Multiple Comparisons (Time p values only)

A summary of the results of the Games-Howell test for scores of each agent can be found in Table 10.13. These show a wider set of variation than the results for time, with one third of the comparisons found to be not statisticaly significantly different. It is to be expected that the results for score might reflect a pattern where the agents running NPE all get similar scores, these results confirm this. It is also interesting that the correlation between agent 0 and agent 4 might suggest that there is a limit to how much an implementing party might want to rely on NPE (although this could also be a missleading figure as a failure to find any matches was recorded as the highest score).

The full results of both One Way ANOVAs can be found in Appendix A.5.7.

	Agent				
	0	1	2	3	4
0		.000	.000	.000	.498
1	.000		1.0	1.0	.002
2	.000	1.0		.999	.002
3	.000	1.0	.999		.001
4	.498	.002	.002	.001	

Table 10.13: Games-Howell Multiple Comparisons (Score p values only)

10.3.2 Independent Samples T-Test

To confirm the findings of the One Way ANOVA and Games-Howell tests a number of Independent Samples T-Tests (McCrum-Gardner, 2008) have been carried out (on the same data set). Specifically T-Tests comparing the mean score and runtime of Agent 3 and Agent 4, and Agent 0 and Agent 3.

Full results of all T-Tests can be found in Appendix A.5.7. A summary of the comparisons listed in the previous paragraph can be found in Table 10.14. As with the ANOVA tests is should be noted that .000 should be read as $p < .001$. One result to consider is that so far all of the Leven tests that have been run have given values which have caused the null hypothosis of that test to be rejected (the data shows Heteroscedasticity). When comparing the mean scores of Agent 3 and Agent 4 the values are conversly found to have Homoscedasticity. This will likely be due to the scores possible for any (norm) catalog and selection of service descriptions being set by the result obtained from the NPE system which we would assume to be the same, and only differ by the implementing agents model of useage for NPE. The results of the T-Tests for Agent 3, Agent 4 and Agent 0 all confirm the results of the ANOVA tests with there being statistical significance between the means of each group.

	Sig. (2-Tailed)	
Group	Time	Score
Agent 3 vs. Agent 4	.008	.000
Agent 0 vs. Agent 3	.000	.000

Table 10.14: Independent Samples T-Test Results (Agent 3 vs. Agent 4 and Agent 0 vs. Agent 3)

As a final set of tests to see if there was a statistical difference between using NPE or not the NPE utilizing agents (Agent 1, Agent 2, Agent 3 and Agent 4) were grouped together, and Agent 0 which does not use NPE was held as the control. Two Independent Samples T-Tests were conducted on the times and scores of these two groups (NPE-based and control). The results of these tests can be found in Appendix A.5.7 or summarize in Table 10.15. Again looking at the full tests it can be seen that the values for scores have Homoscedasticity and pass the null hypothosis of the Leven test. Both of the p values

for these two final tests are $p < 0.05$ and so the means for NPE-based are statistically significantly different form those of the control, signifying that the use of NPE does positively effect the score but also negatively effects the runtime.

	Sig. (2-Tailed)	
	Time	Score
NPE-based Agents vs. Control	.000	.000

Table 10.15: Independent Samples T-Test Results (NPE-based Agents vs. Control)

10.4 Outcomes

The simulations conducted during this chapter shed much-needed light on the effectiveness of norm-based approaches to service selection. By measuring up the performance of several techniques in terms of benefits and costs a comparable analysis of the limits and bounds of normative processing has been achieved.

The first outcome worth noting is that despite NPE not being of production grade, processing of service descriptions was conducted in a timely manner. Although in some cases a single selection run could take an average of twice as long for NPE as opposed to a profile matching agent, the final time taken was not adversely high. Nor were the extra resources consumed prohibitively expensive. In a production grade form, it is likely that an NPE based module could be constructed to reduce costs further and achieve additional improvements. This might still make a norm-based approach unsuitable for use in critical systems (disregarding statistical-based approaches). However, for many more applications NPE could be a useful mechanism for both the inclusion of cultural context and the improvement of service selection.

One important result to take from this set of simulations is that a potential ceiling in the usefulness of NPE has been identified. There are three key variables limiting the performance of NPE: the aggressiveness of the application of normative facts, the method used to handle unrecoverable norm violations and controls limiting service descriptions before processing. During the simulations, Agent 4 repeatedly failed to find a suitable service description to return to the administrator. This was caused by NPE failing all of the service descriptions for repeated or egregious violations of norms. In a real world situation, this could either cause catastrophic loss of potential business for an implementing actor or protect against a set of unsuitable services. Mechanisms to aid this situation are discussed in Section 11.3 and may include the interpolation of results with those from a profile based approach. The second limit is typically soft; the processing requirements for any single run involving NPE can be easily controlled by alerting the number of levels through which NPE analyses. Agent 3 and Agent 4 both utilised NPE

in what has been referred to as "strict" mode; they cease processing a description if an unrecoverable norm was violated. This reduced the processing time in some circumstances and is arguably more efficient. Hypothetically this could lead to the same situation as above whereby no description can be chosen. However, in the simulations conducted here, such an event was not seen. What was seen was a marginal reduction in processing time for agents undertaking this strategy. Finally, the effect of using NPE with an additional matching algorithm was investigated using Agent 1. Here it can be seen that the agent picked substantially more score 0 "correct" services than any other agent. However, the time cost of using both mechanisms was considerable and the precision when no "correct" service exist was severely affected. As a result, when results are considered as a whole with regards to the weighted F1 score, Agent 1 performed no better than the NPE only approach of Agent 3.

Considering the relationships between the implementing agents using the mean±SEM (standard error) and the mean±CI (confidence interval), two interesting outcomes can be gathered. The first is that using NPE offers better results that basic IOPE selection and that the changes implemented to alter the amount to which NPE was used to govern selection generated insignificant enhancements with regards to scores chosen until agent 4 started failing to find a match. The second outcome is that NPE does have an impact on the runtime of the agent, even when the NPE processing cycles are cut short due to a non-recoverable norm being violated.

10.5 Conclusions

At the start of this thesis, a governing research question was posed, backed by three further sub-questions. The work in this chapter answers the three research sub-questions by producing the complete NPE system and through simulated process models showing that it is possible to detect potential normative violations and use this information to select potential partner services. The work detailed in this chapter took the answers produced to solve the research sub-questions and applied them to a number of scenarios with the intention to answer the governing research question. It has been shown in this chapter that a norm-based approach can provide a solid base on which to conduct service selection. Investigations have shown that the methods produced in this thesis can improve the accuracy of service selection in situations when a more traditional algorithm may fail. Furthermore, the governing research question was extended and it has been shown that there are ways to tweak and alter NPE to produce improved performance over base.

This chapter set out to demonstrate that the ideas behind NPE can be applied successfully to a simulated service selection environment. The test was conducted in a careful, well-designed manner and the outcomes have been shown to be reliable. Following analysis, it

has been demonstrated that the NPE module applied to the agent based structure during testing succeeded in demonstrating the capacity of this particular normative approach in increasing the accuracy of service selection. The chosen simulation environment has the supplementary appeal in that it can be transferred to one of many "real world" situations and thus it can be extrapolated that NPE will react in roughly the same manner for almost any other use.

The simulations undertaken in this chapter also exposed some issues with the design and current implementation of NPE. In particular, areas have been identified where improvements can be made to the design of the NPE module in regards to loading and resource usage. It has been seen that the current designs are limited by the extra costs of use and that this is exacerbated when processing large or complex inputs. Conversely, at this point no major issues have been identified in the underlying normative concepts which underpin NPE. These simulations have shown that the ideas used to aid service selection with NPE can produce good results when ancillary implementation issues (such as resource costs) are removed.

A full set of simulations as conducted in the work detailed in this chapter now concludes the current section of this thesis. Attempts have been made to answer both research questions and documentary evidence has been collected to back up any assumptions made. This section has answered many more questions than were initially posed and leaves many more unanswered. Areas such as real-time and hybrid usage as well as investigations into resource issues are all of interest but currently outside of the scope of this thesis. The remainder of this thesis considers ancillary norm usage issues and the use of NPE in a wider service selection context.

Chapter 11

Conclusions and Future Work

The focus of this thesis has been on the use of normative concepts in autonomous computing systems. Particular interest has been paid to the areas of computation on the Semantic Web and the use of agent-based software models. A framework and supporting language for the representation of normative facts in relation to activities has been produced and these concepts have been applied to the selection of Web Service process models in a semantic setting. Following the detailed design and implementation of a module for service selection based on the normative process evaluation structure (referred to as NPE), a set of simulations have been undertaken to explore the viability and limitations of any potential system. The results of these simulations have been used to back up the belief that a normative based approach for service selection can be effective and reliable. Having explored the limits of any core selection module, a full framework for normative partner selection has been outlined. In addition to exploring service selection using normative concepts, a wider view of the requirements of normative based approaches has been investigated. This has included the production, storage and maintainance, all of which are key to the success of any norm based process.

Throughout this thesis research has been guided and given focus by using the procedure of services selection as a test bed on which normative concepts can be applied. This was chosen after initial investigations and has lead to the production of the framework for norm-based process expectations (NPE), an associated language for the representation of norms in relation to activities (NPE-L) and a core module for the selection of services based on service descriptions (NPE-M).

This thesis has constructed an argument for an expanded use of norms in computing outside of that of agent-based systems. It is asserted that normative concepts can be used to both augment current semantic frameworks and also to add much-needed cultural context to globally present computation systems. During this work, references have been made to the need for an increased recognition of the cultural context during the processing of potentially disparate global requirements data. This work has shown that

the inclusion of normative concepts into current computing systems is easy and can yield positive results. NPE as a framework and supporting implementation provides a starting point for continual research into norm-based topics.

11.1 Research Question

In Section 1.4 at the start of this thesis, a single research aim was provided in the form of a governing question. Throughout this thesis, the work contained within has aimed to answer this question through areas where existing research can be reused and gaps where new ideas are needed. To reiterate, the governing question for this thesis was:

> *"Can an actor with a set of known normative beliefs use these beliefs to aid service selection where IOPE matching typically falls short?"*

In addition to this single theme, three sub-questions were raised by splitting the governing question into work-packages. Thus, the intention of this thesis was to answer each of these three questions in turn so as to enable the governing question to be answered. Each of the sub-questions generate answers which create the base on which the final question can be answered. To reiterate, these questions were:

- How can norms be represented so as to best relate to the proposed process models (workflows) of semantically aware actors?
- What is an appropriate mechanism for the reasoning of norms with regards to expected processes which best suits the selection of a partner service?
- What costs or bounds might a normative approach place on actors undertaking service selection?

In answering these questions, it has been proposed that the following contributions will have been made to the research community:

- A model for reasoning over norm based process expectations and an associated ontology for the representation and storage (NPE-L) will be produced. This will have close ties to existing Semantic Web languages, as well as taking cues from the field of normative agent research.
- A module for the reasoning of norms with regards to expected behaviours as derived from semantic web services (NPE-M). This stand-alone module will be built to allow an implementing actor to reason over expected processes and produce results which can be used to influence services based on existing normative beliefs.

- A framework for the testing and verification of normative service selection techniques. This will include a set of input services and a set of input norms, produced in a manner intended to minimise bias.

This conclusion will demonstrate that all of these objectives have been met by the work carried out in this thesis and as such the governing research question has been successfully answered.

11.2 Questions Answered in This Thesis

As has been stated, the research agenda for this thesis has been set through the governing question (Section 1.4). To resolve this question, three sub-questions, identified in Section 1.4, have been proposed. The answers to which create the base on which the governing question has been resolved. To demonstrate that this has been achieved, each research question will be evaluated in turn along with the contributions made by this research:

How can norms be represented so as to best relate to the proposed workflows of semantically aware actors? Chapter 2 identifies that norms are concepts used increasingly in agent-based computing approaches. Numerous languages and approaches have been noted which enable the introduction of normative facts into reasoning cycles. However, a disconnect was identified between these research efforts and the field of service sections (separate from that of agent-based computing). A model for norms which includes the relevant expressions and exceptions to enable utility based calculations to be made was proposed in Chapter 3. This model was then mapped to a concrete language and ontology in Chapter 4. The ontology builds on the structure of the previously described model, adding semantic constructs for the representation of elements and binding tightly with a popular semantic Web service description language (OWL-S). It is submitted that this final language and ontology, based on well founded existing research, forms a suitable base for a language for the representation of norms with regards to processes and activities exposed by service descriptions.

Is there a mechanism for the reasoning of norms with regards to expected processes which best suits the selection of a partner service? If not what features make one stand out from those of existing systems? In a review of existing selection methods conducted in Chapter 2 many research efforts in the field of service selection were identified. Many of these utilised IOPE matching, ignoring process models and QOS based approaches. One of the key tenets of this thesis is that current approaches are mature and suitable for numerous circumstances, but that service

selection can be improved beyond this threshold for actors with access to normative facts. Thus as no single current approach was identified which would fulfil this research question and the aforementioned disconnect between the service selection and normative agent communities prevented any agent approach from being fully utilised, Chapter 5 proposes a module for the reasoning over service descriptions with regards to normative facts. Reasoning is conducted using simulation over an internal Petri net based representation of a services process description and the results are left to any implementing actor to action. The NPE system proposed in this thesis fulfils the criteria of being able to take services descriptions and norms as inputs and produce a set of associated costings which may be used for further reasoning. Above any other systems identified by this research this approach bridges the divide between normative agent solutions and service selection, enabling normative believes to influence service selection before interaction while placing flexible constraints on implementing actors.

Does a normative approach to service selection place any undue limits or bounds on an implementing actor? Having successfully created a system for the reasoning over actions proposed in service descriptions with regards to normative facts, the final question asks if this system could be usefully deployed in a "production" environment. This asks not only "does the system work as expected?" but also "is it useful in scenarios where potential time and resources are also costly". In such "real world" scenarios, any solution must provide adequate utility over the cost of use to be effectively deployed. Chapter 6 outlines the methodology identified to answer this question. A straight forward test was used to confirm that NPE fulfils its original goals. Chapter 9 outlines the results of testing and shows that the NPE module can be implemented so as to fulfil its goal of being able to reason over service descriptions with regards to norms. To fully answer this research question, a simulation protocol was designed (Chapter 6 and Chapter 10) to model how NPE would affect the processing and decisions made by an implementing actor (in this case an agent). These simulations drew on previous research into service selection and aimed to provide an accurate picture as to the effects of NPE. It is submitted that these simulations answer the question of what limits and bounds are placed on an implementing actor. Full conclusions as to the effects recognised can be found in Section 11.3.

Other Questions As a wider point, the question of "Is the area of normative computing worthy of further research?" has also in part been raised during this thesis. Chapter 2 weighs up whether the area of norms in computing is of significant interest to a wider audience. The findings of this thesis back up the decision to research this topic. Normative approaches have the ability to provide large gains in accuracy in processing actions and deeds, as well as enabling new light to be shed on the intentions and activities of others. In addition, the increasing globalisation of computing and the rise of automation bring

into focus the need to have context and culture sensitive processing systems, which may most effectively be realised using normative approaches.

11.2.1 Future Questions to be Answered

The governing and sub-questions identified in Chapter 1.4 have been answered. As with any large research effort in answering these questions, numerous new uncertainties have been unearthed. Issues such as "How might normative facts be stored?", "Should more attention be paid to cultural divisions when choosing business partners?" and "Who should construct normative facts?" are all left unanswered by this thesis. For now they all fall outside of the scope of this work. However, suggestions for future research topics based on questions raised throughout this thesis can be found in Section 11.5.

11.3 Analysis of Simulations

As part of this thesis, to aid answering the questions posed in Section 1.4. two separate and complementary simulation phases were undertaken. These took the form of a testing phase and a full set of simulations. Chapter 9 details the full testing phase including incremental updates as performance was monitored and improved. Chapter 10 follows on from this by taking the form of a full set of simulations testing the limits of the NPE module. Results have been gathered from both and used to generate conclusions in-line with the questions asked in Section 1.4. Detailed analysis of results and outcomes can be found within the previously mentioned chapters. What follows is a brief summary of these simulations and an analysis of the final conclusions drawn from both.

11.3.1 Risk Factors

The primary risk factor involved in producing simulations for use in evaluating NPE was discovered to be one of bias. An almost complete lack of service and norm test cases lead to the requirement that new examples had to be created. The generation of test cases is a common source for the introduction of unwanted bias within the simulations. Producing well-formed test cases which best reflect likely uses in "real world" examples has been at the centre of this thesis.

Chapter 7 and Chapter 8 detail the processes undertaken to utilise automated techniques in the creation of test cases. A large amount of research was conducted to locate the largest set of existing service descriptions possible. In the end, some 1300+ were discovered. As a large number of these are descriptions with only a base level of processes, automated matching modules were used to generate examples of increased complexity. By removing as much human input as possible from this process, it is envisaged that the

test case sets created will draw more from trends and bias present in the "real world" than from the ideas and tendencies of the authors.

To aid in the detection of bias, attention was also paid to the design and monitoring of the NPE system. A testing phase was introduced to allow bias to be detected before full simulations were undertaken. Furthermore the Latin Squares experimental design technique, which was used to generate a protocol for simulations, ensured that fair results were achieved independant of any blocking variables. In summation at all times during this thesis every care has been taken to reduce the potential effects of bias and that the final results of this thesis can be held to be as valid and fair as was possible.

11.3.2 Progress

The chosen two-phase approach to simulations has worked well throughout this thesis. By testing and then simulating, extra care was paid to achieving a platform on which stable results were be achieved. This approach also ensured that the second sub-question had been fully answered before the third sub-question was investigated, by proving the viability of a working system. It was not the intention of this research to produce a production-ready module. The goal was to create a platform to study the effects of varying levels of NPE usage. Measured against this, the development and simulations conducted easily fulfilled the criteria of the goal and research questions phrased previously.

Lessons learned in early development (including those learned from the generation of test cases and other supporting materials) were carried forward and incorporated into final implementations. In particular, the information gained as to the limitations of description processing in Chapter 7 and Chapter 8 provided guidance for every subsequent part of the finished module and test system. In the same way, the lessons learned from early testing were easily fed back into the final simulations. This later enabled full simulations to focus on investigating the theories put in place by NPE and solving the research questions posed previously, instead of concentrating on the relative performance of any one software module.

11.3.3 Outcomes

The simulation phase answered the question "Does a normative approach to service selection place any undue limits or bounds on an implementing actor?" and thus by rendering all of the sub-questions answered, answered the final governing research question. A full review of the testing phase and the simulation phase can be found in Chapter 9 and Chapter 10 respectively. The key outcomes were:

- An NPE based system can be used to help an actor reason over service descriptions with regard to held normative beliefs;

- The results of any simulated process model provides an accurate measure of that process with regards to any norms being used;
- Outcomes are heavily dependent on the quality and relevance of the norms in use;
- An NPE based system can perform better at scoring service descriptions than an equivalent IOPE matching system when sufficient norms are provided;
- Using NPE places only a small burden on an implementing actor (where time is only a marginal constraint);
- The NPE process can be tuned depending on an actors' circumstance to provide results faster (where limits can be placed on the acceptable cost of any interaction and where unrecoverable norms are adhered to).

These outcomes answer the third sub-question and offer a set of calculations on which the answer to the governing question has been constructed. That is to say, that from the above it can be found that a normative approach to service selection places some limits on implementing actors (in terms of time cost and recall). However, for actors where time is a flexible and precision is important, the NPE system can be of benefit. In addition, an answer can now be offered to the governing question of "Can an actor with a set of known normative beliefs use these beliefs to aid service selection where IOPE matching typically falls short?". In this case, the research within this thesis shows that yes normative beliefs can be used to aid service selection and that this process can produce greatly improved results over that of "traditional" IOPE matching techniques used before. What is more, using NPE does not place an unduly high burden on implementing actors compared with the potential gains in utility from increased precision.

11.4 Conclusions

This section concludes this thesis by providing a retrospective look at the goals, processes and achievements covered within. As a whole it has been the aim throughout this thesis to investigate the suitability of normative concepts for use in computing; specifically, for service selection in social situations where actors may or may not be agent based. This has been achieved through the use of a familiar modern computing paradigm; that of service selection by automated actors operating using semantically rich data. The creation of a normative module for service selection suitable for use on the Semantic Web has acted to guide research and provide a focus for development. Through the question posed in Section 1.4 and the results which proceeded them, a final analysis has been conducted as to the suitability of a general model for normative service selection.

What follows is a set of short conclusions from the most notable sections of this thesis.

11.4.1 Literature Review

An extensive literature review conducted in Chapter 2 researched a broad set of issues surrounding distributed, automated and semantic based computing. It was the intention of this investigation to provide a well-developed base of information on which to build. This chapter is a key component of the thesis process as the ability to learn from, adapt and reuse existing ideas enables any end conclusions to be reliably based.

A wealth of research was discovered surrounding the area of computing on the Semantic Web and the use of agent-based architectures. This area was chosen for investigation as there was a perception that it is currently an active area on research and a good base on which to test normative techniques. On the theme of norms, a large section of research into the use of norms in international relations was discovered. Originating in the disciplines of philosophy and social sciences, this work has not yet been fully merged with that of distributed or automated computing. One key issue raised by this literature review is the disconnect between normative agent research and services selection research. No single normative model or service selection technique capable of fulfilling the research goals was identified during this review.

11.4.2 A Model and Ontology for Norms Regarding Service Descriptions

As Chapter 2 uncovered no suitable languages or existing frameworks for the representation of or reasoning of norms with regards to expected processes as exposed by service descriptions, Chapter 3 proposes a model built on existing elements that were identified as useful. The model takes the form of an abstract model which, as it has a base in agent focused norm research, could readily be used for agent reasoning on its own. At this early stage, a decision was made to constrain this model, by choosing a process model constrained action element. This is made clear in Chapter 4 where an ontology and language for the representation of norms with regards to web services exposed using the semantic language OWL-S is produced. Both of these chapters form some of the novel contributions offered up by this thesis.

11.4.3 Norm-Based Selection Module

Building on the preceding chapters, Chapter 5 proposes a system for reasoning over service descriptions with regards to provided norms. This system builds on existing technologies used currently in both agent and Semantic Web communities. This thesis submits that this module, along with the language and representation, form the contribution provided to the community by this research.

11.4.4 Validating NPE

To answer the governing question posed in this thesis, the NPE system proposed and implemented as a solution for situations where service selection involving IOPE matching typically falls short has had to be validated using a minimal set of criteria. The testing and validation sections of this thesis outline how this validation was carried out and what the general results were. Details of these validations can be found in Chapter 10. In general, it was discovered that a norm-based approach does outperform an IOPE only approach, but that it comes at a cost in terms of time taken to retrieve any results.

11.5 Future Work

Throughout this thesis attention has been paid to the identification of topics and areas suitable for further research. One of the key intentions of this work has been to review of the current state of norms and cultural awareness in automated computing structures. This review has then been used to draw attention to matters which may be worthy of wider consideration. What follows is a brief analysis of the areas identified for potential research. With the exception of norm production, all of these fall outside of the scope of this work and so are left as research stubs for future completion.

The majority of focus within future research opportunities falls on the identification (production) and storage of norms. Both these notions are key to the wider adoption and adaptation of any normative computing system. In particular, these areas are required for NPE to be of use on a larger scale.

11.5.1 Norm Production

Norm production is the one area of future work which has been covered as part of the research towards this thesis. Chapter 8 details the creation of norms for test cases using automated approaches. However, this research has barely touched the great wealth of topics available for investigation in this area. The single biggest barrier to the adoption of normative techniques in computer-based processing is the lack of existing norms. Thus this single research area outweighs all others identified here. If a single effective method (or multiple complimentary methods) for the production of normative facts from data is sourced, normative techniques, which have been until now have been ignored, have the potential to be utilised adding much needed cultural and social awareness to computing systems.

Notions can be drawn for this research from numerous different fields. Chapter 2 details existing research efforts based around statistical analysis and complex mathematical simulations. Both these approaches offer good solutions. Of more interest is the use of

concepts drawn from law and psychology to aid in the understanding and construction of norms. Both these disciplines have an existing body of research in the area of conceptual normative thinking. It may be prudent to explore the limits of normative processing and production using notions already confirmed by social sciences.

11.5.2 Norm Storage

Whereas the production of norms falls under the category of an immediate requirement, the storage of norms is a longer term requisite of an active norm-based computing community. Key to the viability of norms as a long term solution to process reasoning on a global scale, norm storage enables norms to be held and distributed in a standardised manner. This facilitates the rapid reuse of existing norms and thus a reduction in the need for new norms to be constructed. On a global scale, stores of norms enable actors from "foreign" networks to join new groups and immediately have access to normative information on the best practices and expected behaviours. Such storage solutions should utilise well-formed ontologies such as NPE-L and permit the rapid sharing of normative information between partners. In addition to external stores, internal stores should be explored to speed up the access to norms as beliefs by actors and enable larger sets to be indexed.

11.5.2.1 Norm Repositories

The success of the best-known research efforts in automated, semantic and online computing has been down to the easy access that well-formed repositories give to information required for processing. UDDI and non-UDDI repositories have helped to increase the popularity of SOAP and WSDL-based Web service applications. OWL repositories offer large stores of ontological data and one might even argue that HTTP-based Web servers are essentially repositories for HTML-based data. A well-formed description language for norms (such as NPE-L) when combined with an easy to use communication and indexing protocol, can enable easier access to existing sets of norms. This would further encourage the use of normative techniques and aid in the adoption of automated norm-based processing.

11.5.2.2 Commercial Repositories

Having repositories of norms may be useful. However, uses where such norms may include the reasoning over decisions with security or monetary significance with a higher state of reliability may be required. For this, it may be possible to rely on commercial entities to create and distribute norms. One could imagine a situation where an entity exists to produce and store norms for purchase by third parties wishing to process information,

evaluate workflows with which they are unfamiliar, or grade proposed process models. Such a configuration can be used to offer remote business partners a chance to interact with a greater degree of confidence that actions suggested by the other are reasonable and to be expected.

11.5.2.3 Description Repositories

As an aside, one of the largest risk factors identified during this thesis was the introduction of bias into test sets for use in simulations. This primarily arose thanks to a lack of complex OWL-S descriptions available for public consumption. There is a clear research gap in the creation of a large set of complex service descriptions. The availability of description languages and a wealth of potential workflows (both fictional and concrete) make the creation of a repository trivial and a boost to researchers and early adopters of semantic technologies.

11.5.3 Global Norms

With the rise of globalisation and specifically a rise in computer use by individuals and businesses across the world, it is more important than now ever to take into account cultural and social principles when exchanging data. Global norms are gradually gaining legitimacy at a local level and local norms are increasingly becoming relevant to international actors. The ability for systems reasoning over actions across disparate groups of self-interested brokers, to utilise normative facts when processing intentions, is of interest when preferable results can be gained. Further research into the uses and effects of global norms is likely to be driven by the wealth of information already gathered by non-governmental organisations such as the UN.

11.5.4 Real-time Norm Usage

The simulations conducted in Chapter 10 have shown that the NPE based system produced as part of this thesis works to aid the selection of services by a self-interested agent. However, the results from these simulations have also shown that in this current form the suggested NPE module is unsuitable for use in scenarios where time is critical such as real-time situations. In such situations, the NPE module is too slow. However, of interest is the ability to represent complex issues as uncomplicated well-formed objects that underpins NPE is of interest. This capacity is easily transferred to the storage and processing of detected activities and rating of potential plans. In a real-time agent scenario utilising a BDI approach, such a procedure has the potential to rival those based on statistical or historical based approaches. This area is another where the basic structures of NPE could be easily reused following additional research.

11.5.5 Norm Validation

A last key domain outlined in numerous areas of this thesis is that of the validity norms both individually and as a group. This is arguably as important as the initial norm creation and covers both syntactic validity as well as semantic validity. Issues in validation spread from a need to ensure that local norms are held to be mutually consistent through to a need to ensure that individual norms are unique on a global scale. These subjects all pose complex questions in the form of processing, addressing and transferring valid norm catalogues. Ideas to address these issues are likely to be drawn from research into both computer science and mathematics.

11.5.6 Evaluation of Complex Process Models Using Petri Nets

In Chapter 5 described the use of Petri nets to represent process models. A mechanism for the conversion from OWL-S to a Petri net is discussed. The NPE system parses resulting nets when simulating service usage. To carry out this simulation tokens are passed around the Petri net, each containing a different belief base resulting from a different execution path. This works for basic control constructs, but in complex operations (available in OWL-S) such as Split-Joins there needs to be a mechanism to merge tokens as paths are rejoined. This thesis has acknowledged this problem but does not propose a solution. Instead, it is noted that this is an area which should receive future attention.

11.6 Evaluation of Thesis

The primary goals of this thesis have been the investigation of normative issues in computing and the creation of an example norm-based system for improving service selection. Norm-based approaches in computing have been historically under-researched and there are large gaps in the understanding of what norms are and how they might best be applied. It has been shown in this thesis that norms can aid interactions between actors by giving cultural and social context to beliefs about actions and activities. Hence, normative approaches may be a powerful addition to the sphere of agent and semantic based computing in an increasingly connected global world.

This thesis has demonstrated a clear framework to enable the description of and reasoning over normative concepts. The NPE platform and associated NPE-L language have been presented as concrete examples for the representation of norms in regard to actions and contextual triggers. The design of this work has been influenced by research from the areas of agent-based computing, semantic data processing, social sciences and law. Furthermore, additional information utilised from sources such as non-governmental organisations and news organisations has helped maintain a focus on current use and

context. Agents exploiting a module based on NPE have been shown to be capable of increased accuracy in service selection. This has emphasised the use of norm-based approaches as tools to aid contextual data processing. In addition to this core research strand, numerous topics have been investigated in support of the design and future implementation of normative systems. In particular, a full-service selection structure has been proposed along with a set of tools and methods for the extraction of norms to defined objects.

At the conclusion of this thesis, a framework for the representation of norms has been identified. This has been used to create a logic based-language for the reasoning of norms in a constant computable manner. A genuine need for a semantic based approach was identified early on during research and an ontology for norms in relation to actions was produced utilising process descriptions taken from the popular OWL-S ontology. As a concrete example of the use of such an ontology, a norm-base approach for service selection has been designed and implemented. This implementation has been fully tested and shown to offer substantial improvement over conventional profile based IO techniques. The end result of this process is a new approach to selection of services where IOPE matching typically falls short.

Appendix: Code

A.1 Example Services

A.1.1 Frango.owl

```
   <?xml version='1.0' encoding='ISO-8859-1'?>
<!DOCTYPE uridef[
  <!ENTITY rdf     "http://www.w3.org/1999/02/22-rdf-syntax-ns">
  <!ENTITY rdfs    "http://www.w3.org/2000/01/rdf-schema">
  <!ENTITY xsd     "http://www.w3.org/2001/XMLSchema">
  <!ENTITY owl     "http://www.w3.org/2002/07/owl">
]>

<rdf:RDF
  xmlns:rdf=     "&rdf;#"
  xmlns:rdfs=    "&rdfs;#"
  xmlns:xsd =     "&xsd;#"
  xmlns:owl = "&owl;#
  xml:base="http://www.ecs.soton.ac.uk/~akd07r/owl1.1/frango.owl">

<owl:versionInfo>
    $Id: frango.owl,v 1.5 14/01/2010 a.douglas Exp $
</owl:versionInfo>

<owl:Ontology>
        <owl:imports rdf:resource="http://www.daml.org/services/owl-s/1.1/Service.owl#"/>
        <owl:imports rdf:resource="http://www.daml.org/services/owl-s/1.1/Profile.owl#"/>
        <owl:imports rdf:resource="http://www.daml.org/services/owl-s/1.1/Process.owl#"/>
        <owl:imports rdf:resource="http://www.daml.org/services/owl-s/1.1/Grounding.owl#"/>
</owl:Ontology>

<!--
        Data Types
-->

<owl:Class rdf:ID="CreditCardType">
  <owl:oneOf rdf:parseType="Collection">
    <CreditCardType rdf:ID="MasterCard"/>
    <CreditCardType rdf:ID="VISA"/>
    <CreditCardType rdf:ID="AmericanExpress"/>
    <CreditCardType rdf:ID="DiscoverCard"/>
   </owl:oneOf>
</owl:Class>
```

```
<owl:Class rdf:ID="PackagingType">
  <owl:oneOf rdf:parseType="Collection">
    <PackagingType rdf:ID="Giftwrap"/>
    <PackagingType rdf:ID="Ordinary"/>
  </owl:oneOf>
</owl:Class>

<owl:Class rdf:ID="DeliveryType">
  <owl:oneOf rdf:parseType="Collection">
    <DeliveryType rdf:ID="FedExOneDay"/>
    <DeliveryType rdf:ID="FedEx2-3day"/>
    <DeliveryType rdf:ID="UPS"/>
    <DeliveryType rdf:ID="OrdinaryMail"/>
  </owl:oneOf>
</owl:Class>

<owl:Class rdf:ID="ValidityType">
  <owl:oneOf rdf:parseType="Collection">
    <ValidityType rdf:ID="Valid"/>
    <ValidityType rdf:ID="Expired"/>
    <ValidityType rdf:ID="InvalidCCNumber"/>
    <ValidityType rdf:ID="InvalidCCType"/>
    <ValidityType rdf:ID="AuthorizationRefused"/>
  </owl:oneOf>
</owl:Class>

<owl:Class rdf:ID="Shipment">
  <rdfs:subClassOf>
    <owl:Restriction>
       <owl:onProperty rdf:resource="#shippedTo"/>
       <owl:cardinality rdf:datatype="&xsd;#integer">1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
       <owl:onProperty rdf:resource="#shippedItem"/>
       <owl:cardinality rdf:datatype="&xsd;#integer">1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
       <owl:onProperty rdf:resource="#deliveryType"/>
       <owl:cardinality rdf:datatype="&xsd;#integer">1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
       <owl:onProperty rdf:resource="#packagingType"/>
       <owl:cardinality rdf:datatype="&xsd;#integer">1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>

<owl:ObjectProperty rdf:ID="shippedTo">
  <rdfs:domain rdf:resource="#Shipment"/>
  <rdfs:range rdf:resource="#AcctID"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="shippedBook">
```

```
  <rdfs:domain rdf:resource="#Shipment"/>
  <rdfs:range rdf:resource="#Item"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="deliveryType">
  <rdfs:domain rdf:resource="#Shipment"/>
  <rdfs:range rdf:resource="#DeliveryType"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="packagingType">
  <rdfs:domain rdf:resource="#Shipment"/>
  <rdfs:range rdf:resource="#PackagingType"/>
</owl:ObjectProperty>


<owl:Class rdf:ID="CreditCard">
  <rdfs:subClassOf>
    <owl:Restriction>
       <owl:onProperty rdf:resource="#cardNumber"/>
       <owl:cardinality rdf:datatype="&xsd;#integer">1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
       <owl:onProperty rdf:resource="#cardType"/>
       <owl:cardinality rdf:datatype="&xsd;#integer">1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
       <owl:onProperty rdf:resource="#cardExpiration"/>
       <owl:cardinality rdf:datatype="&xsd;#integer">1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
       <owl:onProperty rdf:resource="#validity"/>
       <owl:cardinality rdf:datatype="&xsd;#integer">1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>

<owl:DatatypeProperty rdf:ID="cardNumber">
  <rdfs:domain rdf:resource="#CreditCard"/>
  <rdfs:range rdf:resource="&xsd;#integer"/>
</owl:DatatypeProperty>

<owl:ObjectProperty rdf:ID="cardType">
  <rdfs:domain rdf:resource="#CreditCard"/>
  <rdfs:range rdf:resource="#CreditCardType"/>
</owl:ObjectProperty>

<owl:DatatypeProperty rdf:ID="cardExpiration">
  <rdfs:domain rdf:resource="#CreditCard"/>
  <rdfs:range rdf:resource="&xsd;#gYearMonth"/>
</owl:DatatypeProperty>

<owl:ObjectProperty rdf:ID="validity">
  <rdfs:domain rdf:resource="#CreditCard"/>
```

```
  <rdfs:range rdf:resource="#ValidityType"/>
</owl:ObjectProperty>

<owl:Class rdf:ID="OrderShippedAcknowledgment"/>

<owl:Class rdf:ID="FailureNotification">
  <owl:oneOf rdf:parseType="Collection">
    <FailureNotification rdf:ID="NotifyItemNotFound"/>
    <FailureNotification rdf:ID="NotifyItemOutOfStock"/>
  </owl:oneOf>
</owl:Class>

<owl:Class rdf:ID="FrangoOrderOutputType">
  <owl:unionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#OrderShippedAcknowledgment"/>
    <owl:Class rdf:about="#FailureNotification"/>
  </owl:unionOf>
</owl:Class>

<owl:Class rdf:ID="LocateItemOutputType">
  <owl:unionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#ProductID"/>
    <owl:Class rdf:about="#FailureNotification"/>
  </owl:unionOf>
</owl:Class>

<owl:Class rdf:ID="InStockItem">
  <rdfs:subClassOf rdf:resource="#Item"/>
</owl:Class>

<owl:Class rdf:ID="OutOfStockItem">
  <rdfs:subClassOf rdf:resource="#Item"/>
  <owl:disjointWith rdf:resource="#InStockItem"/>
</owl:Class>

<owl:ObjectProperty rdf:ID="hasItem">
  <owl:inverseOf rdf:resource="#hasProductID"/>
</owl:ObjectProperty>

<owl:Class rdf:ID="SignInData"/>

<owl:DatatypeProperty rdf:ID="acctName">
  <rdfs:domain rdf:resource="#SignInData"/>
  <rdfs:range rdf:resource="&xsd;#string"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="password">
  <rdfs:domain rdf:resource="#SignInData"/>
  <rdfs:range rdf:resource="&xsd;#string"/>
</owl:DatatypeProperty>

<owl:Class rdf:ID="UserProfileInfo"/>

<owl:Class rdf:ID="AcctInfo">
 <owl:unionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#SignInData"/>
    <owl:Class rdf:about="#UserProfileInfo"/>
 </owl:unionOf>
</owl:Class>
```

```
<owl:Class rdf:ID="AcctID"/>

<owl:ObjectProperty rdf:ID="hasAcctID">
  <rdf:type rdf:resource="&owl;#FunctionalProperty"/>
  <rdfs:domain rdf:resource="#AcctInfo"/>
  <rdfs:range rdf:resource="#AcctID"/>
</owl:ObjectProperty>

<owl:Class rdf:ID="NonExistingAcct">
  <owl:intersectionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#AcctInfo"/>
    <owl:Restriction>
       <owl:onProperty rdf:resource="#hasAcctID"/>
       <owl:maxCardinality rdf:datatype="&xsd;#integer">0</owl:maxCardinality>
    </owl:Restriction>
  </owl:intersectionOf>
</owl:Class>

<owl:Class rdf:ID="Product">
        <rdfs:comment>
                Top level class for a product sold in the store.
        </rdfs:comment>
</owl:Class>

<owl:Class rdf:ID="Item">
        <rdfs:subClassOf rdf:resource="#Product"/>
</owl:Class>

<owl:ObjectProperty rdf:ID="hasProductID">
        <rdfs:domain rdf:resource="#Item"/>
        <rdfs:range rdf:resource="#ProductID"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="hasItem">
        <owl:inverseOf rdf:resource="#hasProductID"/>
</owl:ObjectProperty>

<owl:DatatypeProperty rdf:ID="name">
        <rdfs:domain rdf:resource="#Item"/>
        <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>

<owl:Class rdf:ID="ISBN"/>

<owl:Class rdf:ID="StockCheck"/>

<owl:DatatypeProperty rdf:ID="inStock">
        <rdfs:domain rdf:resource="#StockCheck"/>
        <rdfs:range rdf:resource="&xsd;#boolean"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="ProductID">
        <rdfs:domain rdf:resource="#StockCheck"/>
        <rdfs:range rdf:resource="&xsd;#integer"/>
</owl:DatatypeProperty>

<!--
        Service Model
```

```
-->

<service:Service rdf:ID="FrangoExpressOrderService">
        <service:presents rdf:resource="#FrangoExpressOrderProfile"/>
        <service:describedBy rdf:resource="#FrangoExpressOrderProcess"/>
        <service:supports rdf:resource="#FrangoExpressOrderGrounding"/>
</service:Service>

<service:Service rdf:ID="FrangoOrderService">
        <service:presents rdf:resource="#FrangoOrderProfile"/>
        <service:describedBy rdf:resource="#FrangoOrderProcess"/>
        <service:supports rdf:resource="#FrangoOrderGrounding"/>
</service:Service>

<!--
        Profile Model
-->

<profile:Profile rdf:ID="FrangoExpressOrderProfile">
        <service:isPresentedBy rdf:resource="#FrangoExpressOrderService"/>
        <profile:serviceName xml:lang="en">Frango Express Order Service</profile:serviceName>
        <profile:has_process rdf:resource="#FrangoExpressOrderProcess"/>

        <profile:textDescription>
                        An express (one step) version of the FrangoOrderService
        </profile:textDescription>

        <profile:hasInput rdf:resource="#FrangoExpressOrderProductID"/>
        <profile:hasInput rdf:resource="#FrangoExpressOrderSignInData"/>
        <profile:hasInput rdf:resource="#FrangoExpressOrderCreditCardNumber"/>
        <profile:hasInput rdf:resource="#FrangoExpressOrderCreditCardType"/>
        <profile:hasInput rdf:resource="#FrangoExpressOrderCreditCardExpirationDate"/>
        <profile:hasInput rdf:resource="#FrangoExpressOrderDeliveryAddress"/>
        <profile:hasInput rdf:resource="#FrangoExpressOrderPackagingSelection"/>
        <profile:hasInput rdf:resource="#FrangoExpressOrderDeliveryTypeSelection"/>
        <profile:hasOutput rdf:resource="#FrangoExpressOrderOutput"/>
        <profile:hasResult rdf:resource="#FrangoExpressOrderNegativeResult"/>
        <profile:hasResult rdf:resource="#FrangoExpressOrderPositiveResult"/>
</profile:Profile>

<profile:Profile rdf:ID="FrangoOrderProfile">
        <service:isPresentedBy rdf:resource="#FrangoOrderService"/>
        <profile:serviceName xml:lang="en">Frango Order Service</profile:serviceName>
        <profile:has_process rdf:resource="#FrangoOrderProcess"/>

        <profile:textDescription>
                        This agentified service provides the opportunity to browse a
                        book selling site and buy books there
        </profile:textDescription>

        <profile:contactInformation>
                <actor:Actor rdf:ID="FrangoBuy_contacts">
                <actor:name>Frango Inc.</actor:name>
                <actor:title>Sales Representative</actor:title>
                <actor:phone>+44 (0)23 8059 6000</actor:phone>
                <actor:email>frango@ecs.soton.ac.uk</actor:email>
                <actor:physicalAddress>
                        Electronics and Computer Science
                        University of Southampton
```

```
                        SO17 1BJ
                        United Kingdom
                </actor:physicalAddress>
                <actor:webURL>http://www.ecs.soton.ac.uk/~akd07r/owl1.1/frango.html</actor:webURL>
                </actor:Actor>
        </profile:contactInformation>

        <profile:hasInput rdf:resource="#FrangoOrderItemName"/>
        <profile:hasInput rdf:resource="#FrangoOrderSignInData"/>
        <profile:hasInput rdf:resource="#FrangoOrderCreditCardNumber"/>
        <profile:hasInput rdf:resource="#FrangoOrderCreditCardType"/>
        <profile:hasInput rdf:resource="#FrangoOrderCreditCardExpirationDate"/>
        <profile:hasOutput rdf:resource="#FrangoOrderOutput"/>
        <profile:hasResult rdf:resource="#FrangoOrderNegativeResult"/>
        <profile:hasResult rdf:resource="#FrangoOrderPositiveResult"/>
</profile:Profile>

<!--
        Process Models: FrangoExpressOrderProcess (Not Complete)
-->

<process:AtomicProcess rdf:ID="FrangoExpressOrderProcess">
  <process:hasInput>
    <process:Input rdf:ID="FrangoExpressOrderProductID">
      <process:parameterType rdf:datatype="&xsd;#anyURI">#ProductID</process:parameterType>
    </process:Input>
  </process:hasInput>

  <process:hasInput>
    <process:Input rdf:ID="FrangoExpressOrderSignInInfo">
      <process:parameterType rdf:datatype="&xsd;#anyURI">#SignInData</process:parameterType>
    </process:Input>
  </process:hasInput>

  <process:hasInput>
    <process:Input rdf:ID="FrangoExpressOrderCreditCardNumber">
      <process:parameterType rdf:datatype="&xsd;#anyURI">&xsd;#integer</process:parameterType>
    </process:Input>
  </process:hasInput>

  <process:hasInput>
    <process:Input rdf:ID="FrangoExpressOrderCreditCardType">
      <process:parameterType rdf:datatype="&xsd;#anyURI">#CreditCardType</process:parameterType>
    </process:Input>
  </process:hasInput>

  <process:hasInput>
    <process:Input rdf:ID="FrangoExpressOrderCreditCardExpirationDate">
      <process:parameterType rdf:datatype="&xsd;#anyURI">&xsd;#gYearMonth</process:parameterType>
    </process:Input>
  </process:hasInput>

  <process:hasLocal>
    <process:Local rdf:ID="FrangoExpressOrderAcctID">
      <process:parameterType rdf:datatype="&xsd;#anyURI">#AcctID</process:parameterType>
    </process:Local>
  </process:hasLocal>

  <process:hasLocal>
```

```
  <process:Local rdf:ID="FrangoExpressOrderCreditCard">
    <process:parameterType rdf:datatype="&xsd;#anyURI">#CreditCard</process:parameterType>
  </process:Local>
</process:hasLocal>

<process:hasPrecondition>
  <expr:SWRL-Condition rdf:ID="FrangoExpressOrderAcctExists">
    <expr:expressionLanguage rdf:resource="&expr;#SWRL"/>
      <expr:expressionBody rdf:parseType="Literal">
        <swrl:AtomList>
          <rdf:first>
            <swrl:IndividualPropertyAtom>
              <swrl:propertyPredicate rdf:resource="#hasAcctID"/>
              <swrl:argument1 rdf:resource="#FrangoExpressOrderSignInInfo"/>
              <swrl:argument2 rdf:resource="#FrangoExpressOrderyAcctID"/>
            </swrl:IndividualPropertyAtom>
          </rdf:first>
          <rdf:rest rdf:resource="&rdf;#nil"/>
        </swrl:AtomList>
      </expr:expressionBody>
  </expr:SWRL-Condition>
</process:hasPrecondition>

<process:hasPrecondition>
  <expr:SWRL-Condition rdf:ID="FrangoExpressOrderCreditExists">
    <expr:expressionBody rdf:parseType="Literal">
      <swrl:AtomList>
        <rdf:first>
          <swrl:IndividualPropertyAtom>
            <swrl:propertyPredicate rdf:resource="#creditNumber"/>
            <swrl:argument1 rdf:resource="#FrangoExpressOrderCreditCard"/>
            <swrl:argument2 rdf:resource="#FrangoExpressOrderCreditCardNumber"/>
          </swrl:IndividualPropertyAtom>
        </rdf:first>
        <rdf:rest>
          <swrl:AtomList>
            <rdf:first>
              <swrl:ClassAtom>
                <swrl:classPredicate rdf:resource="#FrangoExpressOrderCreditCard"/>
                <swrl:argument1 rdf:resource="#Valid"/>
              </swrl:ClassAtom>
            </rdf:first>
            <rdf:rest rdf:resource="&rdf;#nil"/>
          </swrl:AtomList>
        </rdf:rest>
      </swrl:AtomList>
    </expr:expressionBody>
  </expr:SWRL-Condition>
</process:hasPrecondition>

<process:hasOutput>
  <process:Output rdf:ID="FrangoExpressOrderOutput">
    <process:parameterType rdf:datatype="&xsd;#anyURI">#FrangoExpressOrderOutputType</
   process:parameterType>
  </process:Output>
</process:hasOutput>

<process:hasResult>
  <process:Result rdf:ID="FrangoExpressOrderPositiveResult">
```

```
<process:hasResultVar>
   <process:ResultVar rdf:ID="FrangoExpressOrderItem">
      <process:parameterType rdf:datatype="&xsd;#anyURI">#Item</process:parameterType>
   </process:ResultVar>
</process:hasResultVar>
<process:inCondition>
    <expr:SWRL-Condition rdf:ID="FrangoExpressOrderItemInStock">
      <expr:expressionBody rdf:parseType="Literal">
        <swrl:AtomList>
          <rdf:first>
            <swrl:IndividualPropertyAtom>
              <swrl:propertyPredicate rdf:resource="#hasProductID"/>
              <swrl:argument1 rdf:resource="#FrangoExpressOrderItem"/>
              <swrl:argument2 rdf:resource="#FrangoExpressOrderProductID"/>
            </swrl:IndividualPropertyAtom>
          </rdf:first>
          <rdf:rest>
            <swrl:AtomList>
              <rdf:first>
                <swrl:ClassAtom>
                  <swrl:classPredicate rdf:resource="#InStockItem"/>
                  <swrl:argument1 rdf:resource="#FrangoExpressOrderItem"/>
                </swrl:ClassAtom>
              </rdf:first>
              <rdf:rest rdf:resource="&rdf;#nil"/>
            </swrl:AtomList>
          </rdf:rest>
        </swrl:AtomList>
      </expr:expressionBody>
    </expr:SWRL-Condition>
</process:inCondition>
<process:hasEffect>
  <expr:SWRL-Expression>
    <expr:expressionBody rdf:parseType="Literal">
      <swrl:AtomList>
        <rdf:first>
          <swrl:IndividualPropertyAtom>
            <swrl:propertyPredicate rdf:resource="&rdf;#type"/>
            <swrl:argument1 rdf:resource="#FrangoExpressOrderOutput"/>
            <swrl:argument2 rdf:resource="#OrderShippedAcknowledgment"/>
          </swrl:IndividualPropertyAtom>
        </rdf:first>
        <rdf:rest rdf:resource="&rdf;#nil"/>
      </swrl:AtomList>
    </expr:expressionBody>
  </expr:SWRL-Expression>
</process:hasEffect>
<process:hasEffect>
  <expr:SWRL-Expression rdf:ID="FrangoExpressOrderShippedEffect">
      <expr:expressionBody rdf:parseType="Literal">
        <swrl:AtomList>
          <rdf:first>
            <swrl:ClassAtom>
              <swrl:classPredicate rdf:resource="#Shipment"/>
              <swrl:argument1 rdf:resource="#FrangoExpressOrderShipment"/>
            </swrl:ClassAtom>
          </rdf:first>
          <rdf:rest>
            <swrl:AtomList>
```

```
                <rdf:first>
                  <swrl:IndividualPropertyAtom>
                    <swrl:propertyPredicate rdf:resource="#shippedTo"/>
                    <swrl:argument1 rdf:resource="#Shipment"/>
                    <swrl:argument2 rdf:resource="#FrangoExpressOrderoAcctID"/>
                  </swrl:IndividualPropertyAtom>
                </rdf:first>
                <rdf:rest>
                  <swrl:AtomList>
                    <rdf:first>
                      <swrl:IndividualPropertyAtom>
                        <swrl:propertyPredicate rdf:resource="#shippedItem"/>
                        <swrl:argument1 rdf:resource="#Shipment"/>
                        <swrl:argument1 rdf:resource="#FrangoExpressOrderItem"/>
                      </swrl:IndividualPropertyAtom>
                    </rdf:first>
                    <rdf:rest rdf:resource="&rdf;#nil"/>
                  </swrl:AtomList>
                </rdf:rest>
              </swrl:AtomList>
            </rdf:rest>
          </swrl:AtomList>
        </expr:expressionBody>
      </expr:SWRL-Expression>
    </process:hasEffect>
  </process:Result>
</process:hasResult>

<process:hasResult>
  <process:Result rdf:ID="FrangoExpressOrderNegativeResult">
    <process:inCondition>
        <expr:SWRL-Condition rdf:ID="FrangoExpressOrderBItemOutOfStock">
          <expr:expressionBody rdf:parseType="Literal">
            <swrl:AtomList>
              <rdf:first>
                <swrl:ClassAtom>
                  <swrl:classPredicate>
                    <owl:Restriction>
                       <owl:onProperty rdf:resource="#hasItem"/>
                       <owl:allValueFrom rdf:resource="OutOfStockItem"/>
                    </owl:Restriction>
                  </swrl:classPredicate>
                  <swrl:argument1 rdf:resource="#FrangoExpressOrderProductID"/>
                </swrl:ClassAtom>
              </rdf:first>
              <rdf:rest rdf:resource="&rdf;#nil"/>
            </swrl:AtomList>
          </expr:expressionBody>
        </expr:SWRL-Condition>
    </process:inCondition>
    <process:hasEffect>
      <expr:SWRL-Expression>
        <expr:expressionBody rdf:parseType="Literal">
          <swrl:AtomList>
            <rdf:first>
              <swrl:IndividualPropertyAtom>
                <swrl:propertyPredicate rdf:resource="&rdf;#type"/>
                <swrl:argument1 rdf:resource="#FrangoExpressOrderOutput"/>
                <swrl:argument2 rdf:resource="#NotifyItemOutOfStock"/>
```

```
              </swrl:IndividualPropertyAtom>
            </rdf:first>
            <rdf:rest rdf:resource="&rdf;#nil"/>
          </swrl:AtomList>
        </expr:expressionBody>
      </expr:SWRL-Expression>
    </process:hasEffect>
  </process:Result>
</process:hasResult>

</process:AtomicProcess>


<!--
        Process Models: FrangoOrderProcess
-->

<process:CompositeProcess rdf:ID="FrangoOrderProcess">
  <process:hasInput>
    <process:Input rdf:ID="FrangoOrderItemName">
      <process:parameterType rdf:datatype="&xsd;#anyURI">&xsd;#string</process:parameterType>
    </process:Input>
  </process:hasInput>

  <process:hasInput>
    <process:Input rdf:ID="FrangoOrderSignInData">
      <process:parameterType rdf:datatype="&xsd;#anyURI">#SignInData</process:parameterType>
    </process:Input>
  </process:hasInput>

  <process:hasInput>
    <process:Input rdf:ID="FrangoOrderCreditCardNumber">
      <process:parameterType rdf:datatype="&xsd;#anyURI">&xsd;#integer</process:parameterType>
    </process:Input>
  </process:hasInput>

  <process:hasInput>
    <process:Input rdf:ID="FrangoOrderCreditCardType">
      <process:parameterType rdf:datatype="&xsd;#anyURI">#CreditCardType</process:parameterType>
    </process:Input>
  </process:hasInput>

  <process:hasInput>
    <process:Input rdf:ID="FrangoOrderCreditCardExpirationDate">
      <process:parameterType rdf:datatype="&xsd;#anyURI">&xsd;#gYearMonth</process:parameterType>
    </process:Input>
  </process:hasInput>

  <process:hasInput>
    <process:Input rdf:ID="FrangoOrderDeliveryAddress">
      <process:parameterType rdf:datatype="&xsd;#anyURI">&xsd;#string</process:parameterType>
    </process:Input>
  </process:hasInput>

  <process:hasInput>
    <process:Input rdf:ID="FrangoOrderPackagingSelection">
      <process:parameterType rdf:datatype="&xsd;#anyURI">&xsd;#string</process:parameterType>
    </process:Input>
  </process:hasInput>
```

```
<process:hasInput>
  <process:Input rdf:ID="FrangoOrderDeliveryTypeSelection">
    <process:parameterType rdf:datatype="&xsd;#anyURI">#DeliveryType</process:parameterType>
  </process:Input>
</process:hasInput>

<process:hasOutput>
  <process:Output rdf:ID="FrangoOrderOutput">
    <process:parameterType rdf:datatype="&xsd;#anyURI">#FrangoOrderOutputType</
   process:parameterType>
  </process:Output>
</process:hasOutput>

<process:composedOf>
  <process:Sequence>
    <process:components>
      <process:ControlConstructList>
        <objList:first>
          <process:Perform rdf:ID="LocateItemPerform">
            <process:process rdf:resource="#LocateItem"/>
            <process:hasDataFrom>
              <process:InputBinding>
                <process:toParam rdf:resource="#LocateItemName"/>
                <process:valueSource>
                  <process:ValueOf>
                    <process:theVar rdf:resource="#FrangoOrderItemName"/>
                    <process:fromProcess rdf:resource="&process;#TheParentPerform"/>
                  </process:ValueOf>
                </process:valueSource>
              </process:InputBinding>
            </process:hasDataFrom>
          </process:Perform>
        </objList:first>
        <objList:rest>
          <process:ControlConstructList>
            <objList:first>
              <process:If-Then-Else>
                <process:ifCondition>
                  <expr:SWRL-Condition rdf:ID="FrangoOrderItemInStock">
                    <expr:expressionBody rdf:parseType="Literal">
                      <swrl:AtomList>
                        <rdf:first>
                          <swrl:DatavaluedPropertyAtom>
                            <swrl:propertyPredicate rdf:resource="#name"/>
                            <swrl:argument1>
                              <swrl:Variable rdf:ID="#aItem"/>
                            </swrl:argument1>
                            <swrl:argument2 rdf:resource="#FrangoOrderItemName"/>
                          </swrl:DatavaluedPropertyAtom>
                        </rdf:first>
                        <rdf:rest>
                          <swrl:AtomList>
                            <rdf:first>
                              <swrl:ClassAtom>
                                <swrl:classPredicate rdf:resource="#InStockItem"/>
                                <swrl:argument1 rdf:resource="#aItem"/>
                              </swrl:ClassAtom>
                            </rdf:first>
```

```
            <rdf:rest rdf:resource="&rdf;#nil"/>
          </swrl:AtomList>
        </rdf:rest>
      </swrl:AtomList>
    </expr:expressionBody>
  </expr:SWRL-Condition>
</process:ifCondition>
<process:then>
  <process:Perform rdf:ID="FrangoOrderPerform">
    <process:process rdf:resource="#FrangoOrder"/>
    <process:hasDataFrom>
      <process:InputBinding>
        <process:toParam rdf:resource="#FrangoProductID"/>
        <process:valueSource>
          <process:ValueOf>
            <process:theVar rdf:resource="#LocateItemOutput"/>
            <process:fromProcess rdf:resource="#LocateItemPerform"/>
          </process:ValueOf>
        </process:valueSource>
      </process:InputBinding>
    </process:hasDataFrom>
    <process:hasDataFrom>
      <process:InputBinding>
        <process:toParam rdf:resource="#FrangoOrderSignInData"/>
        <process:valueSource>
          <process:ValueOf>
            <process:theVar rdf:resource="#FrangoOrderProcessSignInData"/>
            <process:fromProcess rdf:resource="&process;#TheParentPerform"/>
          </process:ValueOf>
        </process:valueSource>
      </process:InputBinding>
    </process:hasDataFrom>
    <process:hasDataFrom>
      <process:InputBinding>
        <process:toParam rdf:resource="#FrangoOrderCreditCardNumber"/>
        <process:valueSource>
          <process:ValueOf>
            <process:theVar rdf:resource="#FrangoOrderProcessCreditCardNumber"/>
            <process:fromProcess rdf:resource="&process;#TheParentPerform"/>
          </process:ValueOf>
        </process:valueSource>
      </process:InputBinding>
    </process:hasDataFrom>
    <process:hasDataFrom>
      <process:InputBinding>
        <process:toParam rdf:resource="#FrangoOrderCreditCardType"/>
        <process:valueSource>
          <process:ValueOf>
            <process:theVar rdf:resource="#FrangoOrderProcessCreditCardType"/>
            <process:fromProcess rdf:resource="&process;#TheParentPerform"/>
          </process:ValueOf>
        </process:valueSource>
      </process:InputBinding>
    </process:hasDataFrom>
    <process:hasDataFrom>
      <process:InputBinding>
        <process:toParam rdf:resource="#FrangoOrderCreditCardExpirationDate"/>
        <process:valueSource>
          <process:ValueOf>
```

```
                          <process:theVar rdf:resource="#
    FrangoOrderProcessCreditCardExpirationDate"/>
                          <process:fromProcess rdf:resource="&process;#TheParentPerform"/>
                        </process:ValueOf>
                      </process:valueSource>
                    </process:InputBinding>
                  </process:hasDataFrom>
                  <process:hasDataFrom>
                    <process:InputBinding>
                      <process:toParam rdf:resource="#FrangoOrderDeliveryAddress"/>
                      <process:valueSource>
                        <process:ValueOf>
                          <process:theVar rdf:resource="#FrangoOrderProcessDeliveryAddress"/>
                          <process:fromProcess rdf:resource="&process;#TheParentPerform"/>
                        </process:ValueOf>
                      </process:valueSource>
                    </process:InputBinding>
                  </process:hasDataFrom>
                  <process:hasDataFrom>
                    <process:InputBinding>
                      <process:toParam rdf:resource="#FrangoOrderPackagingSelection"/>
                      <process:valueSource>
                        <process:ValueOf>
                          <process:theVar rdf:resource="#FrangoOrderProcessPackagingSelection"
    />
                          <process:fromProcess rdf:resource="&process;#TheParentPerform"/>
                        </process:ValueOf>
                      </process:valueSource>
                    </process:InputBinding>
                  </process:hasDataFrom>
                  <process:hasDataFrom>
                    <process:InputBinding>
                      <process:toParam rdf:resource="#FrangoOrderDeliveryTypeSelection"/>
                      <process:valueSource>
                        <process:ValueOf>
                          <process:theVar rdf:resource="#
    FrangoOrderProcessDeliveryTypeSelection"/>
                          <process:fromProcess rdf:resource="&process;#TheParentPerform"/>
                        </process:ValueOf>
                      </process:valueSource>
                    </process:InputBinding>
                  </process:hasDataFrom>
                </process:Perform>
              </process:then>
            </process:If-Then-Else>
          </objList:first>
          <objList:rest rdf:resource="&objList;#nil"/>
        </process:ControlConstructList>
      </objList:rest>
    </process:ControlConstructList>
  </process:components>
 </process:Sequence>
</process:composedOf>

<process:hasResult>
 <process:Result rdf:ID="FrangoOrderPositiveResult">
  <process:inCondition rdf:resource="#FrangoOrderItemInStock"/>
  <process:hasEffect>
   <expr:SWRL-Expression>
```

```
        <expr:expressionBody rdf:parseType="Literal">
          <swrl:AtomList>
            <rdf:first>
              <swrl:IndividualPropertyAtom>
                <swrl:propertyPredicate rdf:resource="&rdf;#type"></swrl:propertyPredicate>
                <swrl:argument1 rdf:resource="#FrangoOrderOutput"></swrl:argument1>
                <swrl:argument2 rdf:resource="#OrderShippedAcknowledgment"></swrl:argument2>
              </swrl:IndividualPropertyAtom>
            </rdf:first>
            <rdf:rest rdf:resource="&rdf;#nil"/>
          </swrl:AtomList>
        </expr:expressionBody>
      </expr:SWRL-Expression>
    </process:hasEffect>
    <process:hasEffect>
      <expr:SWRL-Expression rdf:ID="FrangoOrderShippedEffect">
          <expr:expressionBody rdf:parseType="Literal">
            <swrl:AtomList>
              <rdf:first>
                <swrl:ClassAtom>
                  <swrl:classPredicate rdf:resource="#Shipment"/>
                  <swrl:argument1 rdf:resource="#FrangoBuyShipment"/>
                </swrl:ClassAtom>
              </rdf:first>
              <rdf:rest rdf:resource="&rdf;#nil"/>
            </swrl:AtomList>
          </expr:expressionBody>
      </expr:SWRL-Expression>
    </process:hasEffect>
  </process:Result>
</process:hasResult>

<process:hasResult>
  <process:Result rdf:ID="FrangoOrderNegativeResult">
    <process:inCondition>
      <expr:SWRL-Condition rdf:ID="FrangoOrderItemOutOfStock">
        <expr:expressionBody rdf:parseType="Literal">
          <swrl:AtomList>
            <rdf:first>
              <swrl:DatavaluedPropertyAtom>
                <swrl:propertyPredicate rdf:resource="#name"/>
                <swrl:argument1>
                          <swrl:Variable rdf:ID="#aItem"/>
                </swrl:argument1>
                <swrl:argument2 rdf:resource="#FrangoOrderItemName"/>
              </swrl:DatavaluedPropertyAtom>
            </rdf:first>
            <rdf:rest>
              <swrl:AtomList>
                <rdf:first>
                  <swrl:ClassAtom>
                    <swrl:classPredicate rdf:resource="#OutOfStockItem"/>
                    <swrl:argument1 rdf:resource="#aItem"/>
                  </swrl:ClassAtom>
                </rdf:first>
                <rdf:rest rdf:resource="&rdf;#nil"/>
              </swrl:AtomList>
            </rdf:rest>
          </swrl:AtomList>
```

```
          </expr:expressionBody>
         </expr:SWRL-Condition>
      </process:inCondition>
      <process:withOutput>
        <process:OutputBinding>
          <process:toParam rdf:resource="#FrangoOrderOutput"/>
          <process:valueSource>
            <process:ValueOf>
              <process:theVar rdf:resource="#LocateItemOutput"/>
              <process:fromProcess rdf:resource="#LocateItemPerform"/>
            </process:ValueOf>
          </process:valueSource>
        </process:OutputBinding>
      </process:withOutput>
    </process:Result>
  </process:hasResult>
</process:CompositeProcess>

<process:CompositeProcess rdf:ID="FrangoOrder">

  <process:hasInput>
    <process:Input rdf:ID="FrangoProductID">
      <process:parameterType rdf:datatype="&xsd;#anyURI">#ProductID</process:parameterType>
    </process:Input>
  </process:hasInput>

  <process:hasInput>
    <process:Input rdf:ID="FrangoOrderSignInData">
      <process:parameterType rdf:datatype="&xsd;#anyURI">#SignInData</process:parameterType>
    </process:Input>
  </process:hasInput>

  <process:hasInput>
    <process:Input rdf:ID="FrangoOrderCreditCardNumber">
      <process:parameterType rdf:datatype="&xsd;#anyURI">&xsd;#integer</process:parameterType>
    </process:Input>
  </process:hasInput>

  <process:hasInput>
    <process:Input rdf:ID="FrangoOrderCreditCardType">
      <process:parameterType rdf:datatype="&xsd;#anyURI">#CreditCardType</process:parameterType>
    </process:Input>
  </process:hasInput>

  <process:hasInput>
    <process:Input rdf:ID="FrangoOrderCreditCardExpirationDate">
      <process:parameterType rdf:datatype="&xsd;#anyURI">">&xsd;#gYearMonth</process:parameterType
     >
    </process:Input>
  </process:hasInput>

  <process:hasInput>
    <process:Input rdf:ID="FrangoOrderDeliveryAddress">
      <process:parameterType rdf:datatype="&xsd;#anyURI">&xsd;#string</process:parameterType>
    </process:Input>
  </process:hasInput>

  <process:hasInput>
    <process:Input rdf:ID="FrangoOrderPackagingSelection">
```

```
    <process:parameterType rdf:datatype="&xsd;#anyURI">#PackagingType</process:parameterType>
  </process:Input>
</process:hasInput>

<process:hasInput>
  <process:Input rdf:ID="FrangoOrderDeliveryTypeSelection">
    <process:parameterType rdf:datatype="&xsd;#anyURI">#DeliveryType</process:parameterType>
  </process:Input>
</process:hasInput>


<process:composedOf>
  <process:Sequence>
    <process:components>
      <process:ControlConstructList>
        <objList:first>
          <process:Perform rdf:ID="OrderSequencePerform">
            <process:process rdf:resource="#OrderSequence"/>
            <process:hasDataFrom>
              <process:InputBinding>
                <process:toParam rdf:resource="#OrderSequenceProductID"/>
                <process:valueSource>
                  <process:ValueOf>
                    <process:theVar rdf:resource="#FrangoProductID"/>
                    <process:fromProcess rdf:resource="&process;#TheParentPerform"/>
                  </process:ValueOf>
                </process:valueSource>
              </process:InputBinding>
            </process:hasDataFrom>
            <process:hasDataFrom>
              <process:InputBinding>
                <process:toParam rdf:resource="#OrderSequenceSignInData"/>
                <process:valueSource>
                  <process:ValueOf>
                    <process:theVar rdf:resource="#FrangoOrderSignInData"/>
                    <process:fromProcess rdf:resource="&process;#TheParentPerform"/>
                  </process:ValueOf>
                </process:valueSource>
              </process:InputBinding>
            </process:hasDataFrom>
            <process:hasDataFrom>
              <process:InputBinding>
                <process:toParam rdf:resource="#OrderSequenceCreditCardNumber"/>
                <process:valueSource>
                  <process:ValueOf>
                    <process:theVar rdf:resource="#FrangoOrderCreditCardNumber"/>
                    <process:fromProcess rdf:resource="&process;#TheParentPerform"/>
                  </process:ValueOf>
                </process:valueSource>
              </process:InputBinding>
            </process:hasDataFrom>
            <process:hasDataFrom>
              <process:InputBinding>
                <process:toParam rdf:resource="#OrderSequenceCreditCardType"/>
                <process:valueSource>
                  <process:ValueOf>
                    <process:theVar rdf:resource="#FrangoOrderCreditCardType"/>
                    <process:fromProcess rdf:resource="&process;#TheParentPerform"/>
                  </process:ValueOf>
```

```
        </process:valueSource>
      </process:InputBinding>
    </process:hasDataFrom>
    <process:hasDataFrom>
      <process:InputBinding>
        <process:toParam rdf:resource="#OrderSequenceCreditCardExpirationDate"/>
        <process:valueSource>
          <process:ValueOf>
            <process:theVar rdf:resource="#FrangoOrderCreditCardExpirationDate"/>
            <process:fromProcess rdf:resource="&process;#TheParentPerform"/>
          </process:ValueOf>
        </process:valueSource>
      </process:InputBinding>
    </process:hasDataFrom>
  </process:Perform>
</objList:first>
<objList:rest>
  <process:ControlConstructList>
    <objList:first>
      <process:Perform rdf:ID="SpecifyDeliveryDetailsPerform">
        <process:process rdf:resource="#SpecifyDeliveryDetails"/>
        <process:hasDataFrom>
          <process:InputBinding>
            <process:toParam rdf:resource="#DeliveryAddress"/>
            <process:valueSource>
              <process:ValueOf>
                <process:theVar rdf:resource="#FrangoOrderDeliveryAddress"/>
                <process:fromProcess rdf:resource="&process;#TheParentPerform"/>
              </process:ValueOf>
            </process:valueSource>
          </process:InputBinding>
        </process:hasDataFrom>
        <process:hasDataFrom>
          <process:InputBinding>
            <process:toParam rdf:resource="#PackagingSelection"/>
            <process:valueSource>
              <process:ValueOf>
                <process:theVar rdf:resource="#FrangoOrderPackagingSelection"/>
                <process:fromProcess rdf:resource="&process;#TheParentPerform"/>
              </process:ValueOf>
            </process:valueSource>
          </process:InputBinding>
        </process:hasDataFrom>
        <process:hasDataFrom>
          <process:InputBinding>
            <process:toParam rdf:resource="#DeliveryTypeSelection"/>
            <process:valueSource>
              <process:ValueOf>
                <process:theVar rdf:resource="#FrangoOrderDeliveryTypeSelection"/>
                <process:fromProcess rdf:resource="&process;#TheParentPerform"/>
              </process:ValueOf>
            </process:valueSource>
          </process:InputBinding>
        </process:hasDataFrom>
      </process:Perform>
    </objList:first>
    <objList:rest>
      <process:ControlConstructList>
        <objList:first>
```

```
                    <process:Perform rdf:ID="FinalizeOrderPerform">
                      <process:process rdf:resource="#FinalizeOrder"/>
                    </process:Perform>
                  </objList:first>
                  <objList:rest rdf:resource="&objList;#nil"/>
                </process:ControlConstructList>
              </objList:rest>
            </process:ControlConstructList>
          </objList:rest>
        </process:ControlConstructList>
      </process:components>
    </process:Sequence>
  </process:composedOf>
</process:CompositeProcess>

<process:CompositeProcess rdf:ID="OrderSequence">
  <process:hasInput>
    <process:Input rdf:ID="OrderSequenceProductID">
      <process:parameterType rdf:datatype="&xsd;#anyURI">#ProductID</process:parameterType>
   </process:Input>
  </process:hasInput>

  <process:hasInput>
    <process:Input rdf:ID="OrderSequenceSignInData">
      <process:parameterType rdf:datatype="&xsd;#anyURI">#SignInData</process:parameterType>
    </process:Input>
  </process:hasInput>

  <process:hasInput>
    <process:Input rdf:ID="OrderSequenceCreditCardNumber">
      <process:parameterType rdf:datatype="&xsd;#anyURI">&xsd;#integer</process:parameterType>
    </process:Input>
  </process:hasInput>

  <process:hasInput>
    <process:Input rdf:ID="OrderSequenceCreditCardType">
      <process:parameterType rdf:datatype="&xsd;#anyURI">#CreditCardType</process:parameterType>
    </process:Input>
  </process:hasInput>

  <process:hasInput>
    <process:Input rdf:ID="OrderSequenceCreditCardExpirationDate">
      <process:parameterType rdf:datatype="&xsd;#anyURI">&xsd;#gYearMonth</process:parameterType>
    </process:Input>
  </process:hasInput>

  <process:hasOutput>
    <process:Output rdf:ID="OrderSequenceAcctID">
      <process:parameterType rdf:datatype="&xsd;#anyURI">#AcctID</process:parameterType>
    </process:Output>
  </process:hasOutput>

  <process:composedOf>
    <process:Sequence>
      <process:components>
        <process:ControlConstructList>
          <objList:first>
            <process:Perform rdf:ID="PutInCartPerform">
              <process:process rdf:resource="#PutInCart"/>
```

```
    <process:hasDataFrom>
      <process:InputBinding>
        <process:toParam rdf:resource="#PutInCartProductID"/>
        <process:valueSource>
          <process:ValueOf>
            <process:theVar rdf:resource="#OrderSequenceProductID"/>
            <process:fromProcess rdf:resource="&process;#TheParentPerform"/>
          </process:ValueOf>
        </process:valueSource>
      </process:InputBinding>
    </process:hasDataFrom>
  </process:Perform>
</objList:first>
<objList:rest>
  <process:ControlConstructList>
    <objList:first>
      <process:Perform rdf:ID="SignInPerform">
        <process:process rdf:resource="#SignInSequence"/>
        <process:hasDataFrom>
          <process:InputBinding>
            <process:toParam rdf:resource="#SignInData"/>
            <process:valueSource>
              <process:ValueOf>
                <process:theVar rdf:resource="#OrderSequenceSignInData"/>
                <process:fromProcess rdf:resource="&process;#TheParentPerform"/>
              </process:ValueOf>
            </process:valueSource>
          </process:InputBinding>
        </process:hasDataFrom>
      </process:Perform>
    </objList:first>
    <objList:rest>
      <process:ControlConstructList>
        <objList:first>
          <process:Perform rdf:ID="SpecifyPaymentMethodPerform">
            <process:process rdf:resource="#SpecifyPaymentMethod"/>
            <process:hasDataFrom>
              <process:InputBinding>
                <process:toParam rdf:resource="#SpecifyPaymentMethodCreditCardNumber"/>
                <process:valueSource>
                  <process:ValueOf>
                    <process:theVar rdf:resource="#OrderSequenceCreditCardNumber"/>
                    <process:fromProcess rdf:resource="&process;#TheParentPerform"/>
                  </process:ValueOf>
                </process:valueSource>
              </process:InputBinding>
            </process:hasDataFrom>
            <process:hasDataFrom>
              <process:InputBinding>
                <process:toParam rdf:resource="#SpecifyPaymentMethodCreditCardType"/>
                <process:valueSource>
                  <process:ValueOf>
                    <process:theVar rdf:resource="#OrderSequenceCreditCardType"/>
                    <process:fromProcess rdf:resource="&process;#TheParentPerform"/>
                  </process:ValueOf>
                </process:valueSource>
              </process:InputBinding>
            </process:hasDataFrom>
            <process:hasDataFrom>
```

```
                    <process:InputBinding>
                      <process:toParam rdf:resource="#
    SpecifyPaymentMethodCreditCardExpirationDate"/>
                      <process:valueSource>
                        <process:ValueOf>
                          <process:theVar rdf:resource="#OrderSequenceCreditCardExpirationDate
    "/>
                          <process:fromProcess rdf:resource="&process;#TheParentPerform"/>
                        </process:ValueOf>
                      </process:valueSource>
                    </process:InputBinding>
                  </process:hasDataFrom>
                </process:Perform>
              </objList:first>
              <objList:rest rdf:resource="&objList;#nil"/>
            </process:ControlConstructList>
          </objList:rest>
        </process:ControlConstructList>
      </objList:rest>
    </process:ControlConstructList>
  </process:components>
 </process:Sequence>
</process:composedOf>

<process:hasResult>
 <process:Result>
  <process:inCondition rdf:resource="&expr;#AlwaysTrue"/>
  <process:withOutput>
   <process:OutputBinding>
    <process:toParam rdf:resource="#OrderSequenceAcctID"/>
    <process:valueSource>
     <process:ValueOf>
      <process:theVar rdf:resource="#SignInAcctID"/>
      <process:fromProcess rdf:resource="#SignInPerform"/>
     </process:ValueOf>
    </process:valueSource>
   </process:OutputBinding>
  </process:withOutput>
 </process:Result>
</process:hasResult>

</process:CompositeProcess>

<process:CompositeProcess rdf:ID="SignInSequence">

 <process:hasInput>
  <process:Input rdf:ID="SignInInfo">
   <process:parameterType rdf:datatype="&xsd;#anyURI">#SignIData</process:parameterType>
  </process:Input>
 </process:hasInput>

 <process:hasOutput>
  <process:Output rdf:ID="SignInAcctID">
   <process:parameterType rdf:datatype="&xsd;#anyURI">#AcctID</process:parameterType>
  </process:Output>
 </process:hasOutput>

 <process:composedOf>
  <process:Sequence>
```

```
      <process:components>
        <process:ControlConstructList>
          <objList:first>
            <process:Perform rdf:ID="SignInPerform">
              <process:process rdf:resource="#SignIn"/>
              <process:hasDataFrom>
                <process:InputBinding>
                  <process:toParam rdf:resource="#SignInInfo"/>
                  <process:valueSource>
                    <process:ValueOf>
                      <process:theVar rdf:resource="#SignInInfo"/>
                      <process:fromProcess rdf:resource="&process;#TheParentPerform"/>
                    </process:ValueOf>
                  </process:valueSource>
                </process:InputBinding>
              </process:hasDataFrom>
            </process:Perform>
          </objList:first>
          <objList:rest>
            <process:ControlConstructList>
              <objList:first>
                <process:Perform rdf:ID="LoadUserProfilePerform">
                  <process:process rdf:resource="#LoadUserProfile"/>
                </process:Perform>
              </objList:first>
              <objList:rest rdf:resource="&objList;#nil"/>
            </process:ControlConstructList>
          </objList:rest>
        </process:ControlConstructList>
      </process:components>
    </process:Sequence>
  </process:composedOf>
</process:CompositeProcess>


<process:AtomicProcess rdf:ID="LocateItem">
  <process:hasInput>
    <process:Input rdf:ID="LocateItemName">
      <process:parameterType rdf:datatype="&xsd;#anyURI">&xsd;#string</process:parameterType>
    </process:Input>
  </process:hasInput>

  <process:hasOutput>
    <process:Output rdf:ID="LocateItemOutput">
      <process:parameterType rdf:datatype="&xsd;#anyURI">#LocateItemOutputType</
     process:parameterType>
    </process:Output>
  </process:hasOutput>

  <process:hasResult>
    <process:Result>
      <process:hasResultVar>
         <process:ResultVar rdf:ID="LocatedItem">
            <process:parameterType rdf:datatype="&xsd;#anyURI">#Item</process:parameterType>
        </process:ResultVar>
      </process:hasResultVar>
      <process:inCondition>
        <expr:SWRL-Condition rdf:ID="ItemInStock">
          <expr:expressionBody rdf:parseType="Literal">
```

```
          <swrl:AtomList>
            <rdf:first>
              <swrl:DatavaluedPropertyAtom>
                <swrl:propertyPredicate rdf:resource="#name"/>
                <swrl:argument1 rdf:resource="#LocatedItem"/>
                <swrl:argument2 rdf:resource="#LocateItemName"/>
              </swrl:DatavaluedPropertyAtom>
            </rdf:first>
            <rdf:rest>
              <swrl:AtomList>
                <rdf:first>
                  <swrl:ClassAtom>
                    <swrl:classPredicate rdf:resource="#InStockItem"/>
                    <swrl:argument1 rdf:resource="#LocatedItem"/>
                  </swrl:ClassAtom>
                </rdf:first>
                <rdf:rest rdf:resource="&rdf;#nil"/>
              </swrl:AtomList>
            </rdf:rest>
          </swrl:AtomList>
        </expr:expressionBody>
      </expr:SWRL-Condition>
    </process:inCondition>
    <process:hasEffect>
      <expr:SWRL-Expression>
        <expr:expressionBody rdf:parseType="Literal">
          <swrl:AtomList>
            <rdf:first>
              <swrl:IndividualPropertyAtom>
                <swrl:propertyPredicate rdf:resource="&rdf;#type"></swrl:propertyPredicate>
                <swrl:argument1 rdf:resource="#LocateItemOutput"></swrl:argument1>
                <swrl:argument2 rdf:resource="#ProductID"></swrl:argument2>
              </swrl:IndividualPropertyAtom>
            </rdf:first>
            <rdf:rest rdf:resource="&rdf;#nil"/>
          </swrl:AtomList>
        </expr:expressionBody>
      </expr:SWRL-Expression>
    </process:hasEffect>
  </process:Result>
</process:hasResult>

<process:hasResult>
  <process:Result>
    <process:inCondition>
      <expr:SWRL-Condition rdf:ID="ItemOutOfStock">
        <expr:expressionBody rdf:parseType="Literal">
          <swrl:AtomList>
            <rdf:first>
              <swrl:DatavaluedPropertyAtom>
                <swrl:propertyPredicate rdf:resource="#name"/>
                <swrl:argument1>
                   <swrl:Variable rdf:ID="#aItem"/>
                </swrl:argument1>
                <swrl:argument2 rdf:resource="#LocateItemName"/>
              </swrl:DatavaluedPropertyAtom>
            </rdf:first>
            <rdf:rest>
              <swrl:AtomList>
```

```
                <rdf:first>
                  <swrl:ClassAtom>
                    <swrl:classPredicate rdf:resource="#OutOfStockItem"/>
                    <swrl:argument1 rdf:resource="#aItem"/>
                  </swrl:ClassAtom>
                </rdf:first>
                <rdf:rest rdf:resource="&rdf;#nil"/>
              </swrl:AtomList>
            </rdf:rest>
          </swrl:AtomList>
        </expr:expressionBody>
      </expr:SWRL-Condition>
    </process:inCondition>
    <process:hasEffect>
      <expr:SWRL-Expression>
        <expr:expressionBody rdf:parseType="Literal">
          <swrl:AtomList>
            <rdf:first>
              <swrl:IndividualPropertyAtom>
                <swrl:propertyPredicate rdf:resource="&rdf;#hasValue"></swrl:propertyPredicate>
                <swrl:argument1 rdf:resource="#LocateItemOutput"></swrl:argument1>
                <swrl:argument2 rdf:resource="#NotifyItemOutOfStock"></swrl:argument2>
              </swrl:IndividualPropertyAtom>
            </rdf:first>
            <rdf:rest rdf:resource="&rdf;#nil"/>
          </swrl:AtomList>
        </expr:expressionBody>
      </expr:SWRL-Expression>
    </process:hasEffect>
  </process:Result>
</process:hasResult>

</process:AtomicProcess>

<process:AtomicProcess rdf:ID="PutInCart">
  <process:hasInput>
    <process:Input rdf:ID="PutInCartProductID">
      <process:parameterType rdf:datatype="&xsd;#anyURI">#ProductID</process:parameterType>
    </process:Input>
  </process:hasInput>
</process:AtomicProcess>

<process:AtomicProcess rdf:ID="SignIn">
  <process:hasInput>
    <process:Input rdf:ID="SignInInfo">
      <process:parameterType rdf:datatype="&xsd;#anyURI">#SignInData</process:parameterType>
    </process:Input>
  </process:hasInput>
</process:AtomicProcess>

<process:AtomicProcess rdf:ID="LoadUserProfile">
</process:AtomicProcess>

<process:AtomicProcess rdf:ID="SpecifyPaymentMethod">
  <process:hasInput>
    <process:Input rdf:ID="SpecifyPaymentMethodCreditCardNumber">
      <process:parameterType rdf:datatype="&xsd;#anyURI">&xsd;#decimal</process:parameterType>
    </process:Input>
  </process:hasInput>
```

```
  <process:hasInput>
    <process:Input rdf:ID="SpecifyPaymentMethodCreditCardType">
      <process:parameterType rdf:datatype="&xsd;#anyURI">#CreditCardType</process:parameterType>
    </process:Input>
  </process:hasInput>
  <process:hasInput>
    <process:Input rdf:ID="SpecifyPaymentMethodCreditCardExpirationDate">
      <process:parameterType rdf:datatype="&xsd;#anyURI">&xsd;#gYearMonth</process:parameterType>
    </process:Input>
  </process:hasInput>
</process:AtomicProcess>

<process:AtomicProcess rdf:ID="SpecifyDeliveryDetails">
  <process:hasInput>
    <process:Input rdf:ID="DeliveryAddress">
      <process:parameterType rdf:datatype="&xsd;#anyURI">&xsd;#string</process:parameterType>
    </process:Input>
  </process:hasInput>
  <process:hasInput>
    <process:Input rdf:ID="PackagingSelection">
      <process:parameterType rdf:datatype="&xsd;#anyURI">#PackagingType</process:parameterType>
    </process:Input>
  </process:hasInput>
  <process:hasInput>
    <process:Input rdf:ID="DeliveryTypeSelection">
      <process:parameterType rdf:datatype="&xsd;#anyURI">#DeliveryType</process:parameterType>
    </process:Input>
  </process:hasInput>
</process:AtomicProcess>

<process:AtomicProcess rdf:ID="FinalizeOrder"/>

<process:SimpleProcess rdf:ID="AbstractFrangoOrder">
  <process:expandsTo rdf:resource="#FrangoOrder"/>
</process:SimpleProcess>

<process:CompositeProcess rdf:about="#FrangoOrder">
  <process:collapsesTo rdf:resource="#AbstractFrangoOrder"/>
</process:CompositeProcess>


<!--
        Grounding Model (Abridged)
-->

<grounding:WsdlGrounding rdf:ID="FrangoExpressOrderGrounding">
        <service:supportedBy rdf:resource="#FrangoExpressOrderService"/>
        <grounding:hasAtomicProcessGrounding rdf:resource="#FrangoExpressOrderProcessGrounding"/>
</grounding:WsdlGrounding>

<grounding:WsdlGrounding rdf:ID="FrangoOrderGrounding">
        <service:supportedBy rdf:resource="#FrangoOrderService"/>
        <grounding:hasAtomicProcessGrounding rdf:resource="#FrangoOrderProcessGrounding"/>
</grounding:WsdlGrounding>

</rdf:RDF>
```

A.2 NPE-L Ontology

A.2.1 Catalog.owl

```
<?xml version="1.0"?>


<!DOCTYPE rdf:RDF [
    <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
    <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
    <!ENTITY owl2xml "http://www.w3.org/2006/12/owl2-xml#" >
    <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
    <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
    <!ENTITY catalog "http://web.me.com/adouglas/ontologies/catalog.owl#" >
]>


<rdf:RDF xmlns="http://web.me.com/adouglas/ontologies/catalog.owl#"
     xml:base="http://web.me.com/adouglas/ontologies/catalog.owl"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns:owl2xml="http://www.w3.org/2006/12/owl2-xml#"
     xmlns:owl="http://www.w3.org/2002/07/owl#"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:catalog="http://web.me.com/adouglas/ontologies/catalog.owl#">
    <owl:Ontology rdf:about="">
        <owl:versionInfo
            >$Id: catalog.owl,v 1.1  $</owl:versionInfo>
        <rdfs:comment
            >Base ontology for a catalog of norms.</rdfs:comment>
    </owl:Ontology>



    <!--
    ///////////////////////////////////////////////////////////////////////////////////////
    //
    // Object Properties
    //
    ///////////////////////////////////////////////////////////////////////////////////////
     -->




    <!-- http://web.me.com/adouglas/ontologies/catalog.owl#administeredBy -->

    <owl:ObjectProperty rdf:about="&catalog;administeredBy">
        <rdfs:range rdf:resource="&catalog;Catalog"/>
        <rdfs:domain rdf:resource="&catalog;CatalogExceptions"/>
        <owl:inverseOf rdf:resource="&catalog;administers"/>
        <owl:equivalentProperty rdf:resource="&catalog;isAdministeredBy"/>
    </owl:ObjectProperty>



    <!-- http://web.me.com/adouglas/ontologies/catalog.owl#administers -->
```

```
<owl:ObjectProperty rdf:about="&catalog;administers">
    <rdfs:domain rdf:resource="&catalog;Catalog"/>
    <rdfs:range rdf:resource="&catalog;CatalogExceptions"/>
</owl:ObjectProperty>



<!-- http://web.me.com/adouglas/ontologies/catalog.owl#governedBy -->

<owl:ObjectProperty rdf:about="&catalog;governedBy">
    <rdfs:range rdf:resource="&catalog;Catalog"/>
    <rdfs:domain rdf:resource="&catalog;CatalogActivities"/>
    <owl:inverseOf rdf:resource="&catalog;governs"/>
    <owl:equivalentProperty rdf:resource="&catalog;isGovernedBy"/>
</owl:ObjectProperty>



<!-- http://web.me.com/adouglas/ontologies/catalog.owl#governs -->

<owl:ObjectProperty rdf:about="&catalog;governs">
    <rdfs:domain rdf:resource="&catalog;Catalog"/>
    <rdfs:range rdf:resource="&catalog;CatalogActivities"/>
</owl:ObjectProperty>



<!-- http://web.me.com/adouglas/ontologies/catalog.owl#isAdministeredBy -->

<owl:ObjectProperty rdf:about="&catalog;isAdministeredBy"/>



<!-- http://web.me.com/adouglas/ontologies/catalog.owl#isGovernedBy -->

<owl:ObjectProperty rdf:about="&catalog;isGovernedBy"/>



<!-- http://web.me.com/adouglas/ontologies/catalog.owl#isPopulatedBy -->

<owl:ObjectProperty rdf:about="&catalog;isPopulatedBy">
    <owl:equivalentProperty rdf:resource="&catalog;populatedBy"/>
</owl:ObjectProperty>

<!-- http://web.me.com/adouglas/ontologies/catalog.owl#isPopulatedBy -->

<owl:ObjectProperty rdf:about="&catalog;isProvidedBy">
    <owl:equivalentProperty rdf:resource="&catalog;providedBy"/>
</owl:ObjectProperty>

<!-- http://web.me.com/adouglas/ontologies/catalog.owl#isPresentedBy -->

<owl:ObjectProperty rdf:about="&catalog;isPresentedBy">
    <owl:equivalentProperty rdf:resource="&catalog;presentedBy"/>
</owl:ObjectProperty>
```

```
<!-- http://web.me.com/adouglas/ontologies/catalog.owl#populatedBy -->

<owl:ObjectProperty rdf:about="&catalog;populatedBy">
    <rdfs:domain rdf:resource="&catalog;Catalog"/>
    <rdfs:range rdf:resource="&catalog;CatalogNorms"/>
    <owl:inverseOf rdf:resource="&catalog;populates"/>
</owl:ObjectProperty>



<!-- http://web.me.com/adouglas/ontologies/catalog.owl#populates -->

<owl:ObjectProperty rdf:about="&catalog;populates"/>



<!-- http://web.me.com/adouglas/ontologies/catalog.owl#presentedBy -->

<owl:ObjectProperty rdf:about="&catalog;presentedBy">
    <rdfs:range rdf:resource="&catalog;Catalog"/>
    <rdfs:domain rdf:resource="&catalog;CatalogProfile"/>
    <owl:inverseOf rdf:resource="&catalog;presents"/>
</owl:ObjectProperty>



<!-- http://web.me.com/adouglas/ontologies/catalog.owl#presents -->

<owl:ObjectProperty rdf:about="&catalog;presents">
    <rdfs:domain rdf:resource="&catalog;Catalog"/>
    <rdfs:range rdf:resource="&catalog;CatalogProfile"/>
</owl:ObjectProperty>



<!-- http://web.me.com/adouglas/ontologies/catalog.owl#providedBy -->

<owl:ObjectProperty rdf:about="&catalog;providedBy">
    <rdfs:domain rdf:resource="&catalog;Catalog"/>
</owl:ObjectProperty>



<!-- http://web.me.com/adouglas/ontologies/catalog.owl#provides -->

<owl:ObjectProperty rdf:about="&catalog;provides">
    <rdfs:range rdf:resource="&catalog;Catalog"/>
    <owl:inverseOf rdf:resource="&catalog;providedBy"/>
</owl:ObjectProperty>



<!--
///////////////////////////////////////////////////////////////////////////////////
//
// Classes
//
///////////////////////////////////////////////////////////////////////////////////
 -->
```

```
    <!-- http://web.me.com/adouglas/ontologies/catalog.owl#Catalog -->

    <owl:Class rdf:about="&catalog;Catalog">
        <rdfs:label>Catalog</rdfs:label>
        <rdfs:comment
            >A catalog has 0 or more Norms associated with it. It may also have 0 or more
     Activities and Exceptions associated with it. It should be noted though, that Activities and
     Exceptions can both be represeneted within the norms by which they are governed.</
     rdfs:comment>
    </owl:Class>



    <!-- http://web.me.com/adouglas/ontologies/catalog.owl#CatalogActivities -->

    <owl:Class rdf:about="&catalog;CatalogActivities">
        <rdfs:label
            >CatalogActivities</rdfs:label>
        <rdfs:comment
            >See comments above.</rdfs:comment>
    </owl:Class>



    <!-- http://web.me.com/adouglas/ontologies/catalog.owl#CatalogExceptions -->

    <owl:Class rdf:about="&catalog;CatalogExceptions">
        <rdfs:label
            >CatalogExceptions</rdfs:label>
        <rdfs:comment
            >See comments above.</rdfs:comment>
    </owl:Class>



    <!-- http://web.me.com/adouglas/ontologies/catalog.owl#CatalogNorms -->

    <owl:Class rdf:about="&catalog;CatalogNorms">
        <rdfs:label>CatalogNorms</rdfs:label>
        <rdfs:comment
            >See comments above.</rdfs:comment>
    </owl:Class>



    <!-- http://web.me.com/adouglas/ontologies/catalog.owl#CatalogProfile -->

    <owl:Class rdf:about="&catalog;CatalogProfile">
        <rdfs:label>CatalogProfile</rdfs:label>
        <rdfs:comment
            >See comments above.</rdfs:comment>
    </owl:Class>
</rdf:RDF>
```

```
<!-- Generated by the OWL API (version 2.2.1.1138) http://owlapi.sourceforge.net -->
```

A.2.2 CatalogProfile.owl

```
<?xml version="1.0"?>


<!DOCTYPE rdf:RDF [
    <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
    <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
    <!ENTITY owl2xml "http://www.w3.org/2006/12/owl2-xml#" >
    <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
    <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
    <!ENTITY role "http://public.me.com/adouglas/ontologies/role.owl#" >
    <!ENTITY catalog "http://web.me.com/adouglas/ontologies/catalog.owl#" >
    <!ENTITY catalognorms "http://web.me.com/adouglas/ontologies/catalognorms.owl#" >
    <!ENTITY catalogprofile "http://web.me.com/adouglas/ontologies/catalogprofile.owl#" >
]>


<rdf:RDF xmlns="http://web.me.com/adouglas/ontologies/catalogprofile.owl#"
     xml:base="http://web.me.com/adouglas/ontologies/catalogprofile.owl"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns:catalognorms="http://web.me.com/adouglas/ontologies/catalognorms.owl#"
     xmlns:owl2xml="http://www.w3.org/2006/12/owl2-xml#"
     xmlns:catalogprofile="http://web.me.com/adouglas/ontologies/catalogprofile.owl#"
     xmlns:catalog="http://web.me.com/adouglas/ontologies/catalog.owl#"
     xmlns:owl="http://www.w3.org/2002/07/owl#"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:role="http://public.me.com/adouglas/ontologies/role.owl#"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
    <owl:Ontology rdf:about="">
        <owl:versionInfo
            >$Id: catalogprofile.owl,v 1.0  $</owl:versionInfo>
        <owl:versionInfo
            >The catalog profile is a record of the abilites of the norms stored within.</
     owl:versionInfo>
        <owl:imports rdf:resource="http://web.me.com/adouglas/ontologies/catalog.owl"/>
        <owl:imports rdf:resource="http://web.me.com/adouglas/ontologies/catalognorms.owl"/>
        <owl:imports rdf:resource="http://www.daml.org/services/owl-s/1.1/generic/Expression.owl"/
     >
    </owl:Ontology>



    <!--
    ///////////////////////////////////////////////////////////////////////////////////////
    //
    // Object Properties
    //
    ///////////////////////////////////////////////////////////////////////////////////////
     -->
```

```
<!-- http://web.me.com/adouglas/ontologies/catalogprofile.owl#cParameter -->

<owl:ObjectProperty rdf:about="#cParameter">
    <rdfs:domain rdf:resource="#CatalogParameter"/>
    <rdfs:range rdf:resource="&owl;Thing"/>
</owl:ObjectProperty>



<!-- http://web.me.com/adouglas/ontologies/catalogprofile.owl#catalogCategory -->

<owl:ObjectProperty rdf:about="#catalogCategory">
    <rdfs:range rdf:resource="&role;BusinessCategory"/>
    <rdfs:domain rdf:resource="#Profile"/>
</owl:ObjectProperty>



<!-- http://web.me.com/adouglas/ontologies/catalogprofile.owl#catalogParameter -->

<owl:ObjectProperty rdf:about="#catalogParameter">
    <rdfs:range rdf:resource="#CatalogParameter"/>
    <rdfs:domain rdf:resource="#Profile"/>
</owl:ObjectProperty>



<!-- http://web.me.com/adouglas/ontologies/catalogprofile.owl#hasNorm -->

<owl:ObjectProperty rdf:about="#hasNorm">
    <rdfs:range rdf:resource="&catalognorms;Norm"/>
    <rdfs:domain rdf:resource="#Profile"/>
</owl:ObjectProperty>



<!-- http://web.me.com/adouglas/ontologies/catalogprofile.owl#hasParameter -->

<owl:ObjectProperty rdf:about="#hasParameter">
    <rdfs:domain rdf:resource="#Profile"/>
</owl:ObjectProperty>



<!-- http://web.me.com/adouglas/ontologies/catalogprofile.owl#hasRole -->

<owl:ObjectProperty rdf:about="#hasRole">
    <rdfs:range rdf:resource="&role;Role"/>
    <rdfs:domain rdf:resource="#Profile"/>
</owl:ObjectProperty>



<!--
///////////////////////////////////////////////////////////////////////////////////////
//
// Data properties
//
///////////////////////////////////////////////////////////////////////////////////////
```

```
 -->



<!-- http://web.me.com/adouglas/ontologies/catalogprofile.owl#catalogParameterName -->

<owl:DatatypeProperty rdf:about="#catalogParameterName">
    <rdfs:domain rdf:resource="#CatalogParameter"/>
</owl:DatatypeProperty>



<!-- http://web.me.com/adouglas/ontologies/catalogprofile.owl#parameterType -->

<owl:DatatypeProperty rdf:about="#parameterType">
    <rdfs:range rdf:resource="&xsd;anyURI"/>
</owl:DatatypeProperty>



<!-- http://web.me.com/adouglas/ontologies/catalogprofile.owl#parameterValue -->

<owl:DatatypeProperty rdf:about="#parameterValue">
    <rdfs:range rdf:resource="&rdf;XMLLiteral"/>
</owl:DatatypeProperty>



<!-- http://web.me.com/adouglas/ontologies/catalogprofile.owl#textDescription -->

<owl:DatatypeProperty rdf:about="#textDescription">
    <rdfs:domain rdf:resource="#Profile"/>
</owl:DatatypeProperty>



<!--
///////////////////////////////////////////////////////////////////////////////////////
//
// Classes
//
///////////////////////////////////////////////////////////////////////////////////////
 -->




<!-- http://public.me.com/adouglas/ontologies/role.owl#BusinessCategory -->

<owl:Class rdf:about="&role;BusinessCategory"/>



<!-- http://public.me.com/adouglas/ontologies/role.owl#Role -->

<owl:Class rdf:about="&role;Role"/>
```

```
<!-- http://web.me.com/adouglas/ontologies/catalog.owl#CatalogNorms -->

<owl:Class rdf:about="&catalog;CatalogNorms"/>



<!-- http://web.me.com/adouglas/ontologies/catalog.owl#CatalogProfile -->

<owl:Class rdf:about="&catalog;CatalogProfile"/>



<!-- http://web.me.com/adouglas/ontologies/catalognorms.owl#Norm -->

<owl:Class rdf:about="&catalognorms;Norm">
    <rdfs:subClassOf rdf:resource="&catalog;CatalogNorms"/>
</owl:Class>



<!-- http://web.me.com/adouglas/ontologies/catalogprofile.owl#CatalogParameter -->

<owl:Class rdf:about="#CatalogParameter">
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#cParameter"/>
            <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#catalogParameterName"/>
            <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
        </owl:Restriction>
    </rdfs:subClassOf>
</owl:Class>



<!-- http://web.me.com/adouglas/ontologies/catalogprofile.owl#Profile -->

<owl:Class rdf:about="#Profile">
    <rdfs:label>Profile</rdfs:label>
    <rdfs:subClassOf rdf:resource="&catalog;CatalogProfile"/>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#textDescription"/>
            <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:comment
        >A profile can only have one description</rdfs:comment>
    <rdfs:comment
        >A profile can only have one name</rdfs:comment>
</owl:Class>
```

```
    <!-- http://www.w3.org/2002/07/owl#Thing -->

    <owl:Class rdf:about="&owl;Thing"/>
</rdf:RDF>



<!-- Generated by the OWL API (version 2.2.1.1138) http://owlapi.sourceforge.net -->
```

A.2.3 CatalogActivities.owl

```
<?xml version="1.0"?>


<!DOCTYPE rdf:RDF [
    <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
    <!ENTITY swrl "http://www.w3.org/2003/11/swrl#" >
    <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
    <!ENTITY owl2xml "http://www.w3.org/2006/12/owl2-xml#" >
    <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
    <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
    <!ENTITY catalog "http://web.me.com/adouglas/ontologies/catalog.owl#" >
    <!ENTITY catalogactivites "http://web.me.com/adouglas/ontologies/catalogactivites.owl#" >
    <!ENTITY ObjectList "http://www.daml.org/services/owl-s/1.1/generic/ObjectList.owl#" >
    <!ENTITY Expression "http://www.daml.org/services/owl-s/1.1/generic/Expression.owl#" >
]>


<rdf:RDF xmlns="http://web.me.com/adouglas/ontologies/catalogactivites.owl#"
     xml:base="http://web.me.com/adouglas/ontologies/catalogactivites.owl"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns:swrl="http://www.w3.org/2003/11/swrl#"
     xmlns:owl2xml="http://www.w3.org/2006/12/owl2-xml#"
     xmlns:catalog="http://web.me.com/adouglas/ontologies/catalog.owl#"
     xmlns:catalogactivites="http://web.me.com/adouglas/ontologies/catalogactivites.owl#"
     xmlns:owl="http://www.w3.org/2002/07/owl#"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:Expression="http://www.daml.org/services/owl-s/1.1/generic/Expression.owl#"
     xmlns:ObjectList="http://www.daml.org/services/owl-s/1.1/generic/ObjectList.owl#">
    <owl:Ontology rdf:about="">
        <owl:versionInfo
            >$Id: catalogactivities.owl,v 0.1  $</owl:versionInfo>
        <owl:imports rdf:resource="http://web.me.com/adouglas/ontologies/catalog.owl"/>
        <owl:imports rdf:resource="http://www.daml.org/services/owl-s/1.1/generic/Expression.owl"/
     >
        <owl:imports rdf:resource="http://www.daml.org/services/owl-s/1.1/generic/ObjectList.owl"/
     >
    </owl:Ontology>



    <!--
    ///////////////////////////////////////////////////////////////////////////////////////
    //
    // Object Properties
    //
```

```
//////////////////////////////////////////////////////////////////////////////////
 -->



<!-- http://web.me.com/adouglas/ontologies/catalogactivites.owl#collapse -->

<owl:ObjectProperty rdf:about="#collapse">
    <owl:equivalentProperty rdf:resource="#collapsesTo"/>
</owl:ObjectProperty>



<!-- http://web.me.com/adouglas/ontologies/catalogactivites.owl#collapsesTo -->

<owl:ObjectProperty rdf:about="#collapsesTo">
    <rdfs:domain rdf:resource="#CompositeProcess"/>
    <rdfs:range rdf:resource="#SimpleProcess"/>
</owl:ObjectProperty>



<!-- http://web.me.com/adouglas/ontologies/catalogactivites.owl#components -->

<owl:ObjectProperty rdf:about="#components">
    <rdfs:domain>
        <owl:Class>
            <owl:unionOf rdf:parseType="Collection">
                <rdf:Description rdf:about="#Any-Order"/>
                <rdf:Description rdf:about="#Choice"/>
                <rdf:Description rdf:about="#Sequence"/>
                <rdf:Description rdf:about="#Split"/>
                <rdf:Description rdf:about="#Split-Join"/>
            </owl:unionOf>
        </owl:Class>
    </rdfs:domain>
</owl:ObjectProperty>



<!-- http://web.me.com/adouglas/ontologies/catalogactivites.owl#composedOf -->

<owl:ObjectProperty rdf:about="#composedOf">
    <rdfs:domain rdf:resource="#CompositeProcess"/>
    <rdfs:range rdf:resource="#ProcessConstruct"/>
</owl:ObjectProperty>



<!-- http://web.me.com/adouglas/ontologies/catalogactivites.owl#computedInput -->

<owl:ObjectProperty rdf:about="#computedInput">
    <rdfs:domain rdf:resource="#CompositeProcess"/>
    <rdfs:range rdf:resource="&owl;Thing"/>
</owl:ObjectProperty>
```

```
<!-- http://web.me.com/adouglas/ontologies/catalogactivites.owl#computedOutput -->

<owl:ObjectProperty rdf:about="#computedOutput">
    <rdfs:domain rdf:resource="#CompositeProcess"/>
    <rdfs:range rdf:resource="&owl;Thing"/>
</owl:ObjectProperty>



<!-- http://web.me.com/adouglas/ontologies/catalogactivites.owl#else -->

<owl:ObjectProperty rdf:about="#else">
    <rdfs:domain rdf:resource="#If-Then-Else"/>
    <rdfs:range rdf:resource="#ProcessConstruct"/>
</owl:ObjectProperty>



<!-- http://web.me.com/adouglas/ontologies/catalogactivites.owl#encapsulatedBy -->

<owl:ObjectProperty rdf:about="#encapsulatedBy">
    <rdfs:range rdf:resource="#Activity"/>
    <rdfs:domain rdf:resource="#Process"/>
    <owl:equivalentProperty rdf:resource="#isEncapsulatedBy"/>
</owl:ObjectProperty>



<!-- http://web.me.com/adouglas/ontologies/catalogactivites.owl#encapsulates -->

<owl:ObjectProperty rdf:about="#encapsulates">
    <rdfs:domain rdf:resource="#Activity"/>
    <rdfs:range rdf:resource="#Process"/>
    <owl:inverseOf rdf:resource="#encapsulatedBy"/>
</owl:ObjectProperty>



<!-- http://web.me.com/adouglas/ontologies/catalogactivites.owl#expands -->

<owl:ObjectProperty rdf:about="#expands">
    <owl:equivalentProperty rdf:resource="#expandsTo"/>
</owl:ObjectProperty>



<!-- http://web.me.com/adouglas/ontologies/catalogactivites.owl#expandsTo -->

<owl:ObjectProperty rdf:about="#expandsTo">
    <rdfs:range rdf:resource="#CompositeProcess"/>
    <rdfs:domain rdf:resource="#SimpleProcess"/>
    <owl:inverseOf rdf:resource="#collapsesTo"/>
</owl:ObjectProperty>



<!-- http://web.me.com/adouglas/ontologies/catalogactivites.owl#hasInput -->

<owl:ObjectProperty rdf:about="#hasInput">
```

```
    <rdfs:domain rdf:resource="#Activity"/>
    <rdfs:range rdf:resource="#Input"/>
    <rdfs:subPropertyOf rdf:resource="#hasParameter"/>
</owl:ObjectProperty>



<!-- http://web.me.com/adouglas/ontologies/catalogactivites.owl#hasOutput -->

<owl:ObjectProperty rdf:about="#hasOutput">
    <rdfs:domain rdf:resource="#Activity"/>
    <rdfs:range rdf:resource="#Output"/>
    <rdfs:subPropertyOf rdf:resource="#hasParameter"/>
</owl:ObjectProperty>



<!-- http://web.me.com/adouglas/ontologies/catalogactivites.owl#hasParameter -->

<owl:ObjectProperty rdf:about="#hasParameter">
    <rdfs:domain rdf:resource="#Activity"/>
    <rdfs:range rdf:resource="#Parameter"/>
</owl:ObjectProperty>



<!-- http://web.me.com/adouglas/ontologies/catalogactivites.owl#ifCondition -->

<owl:ObjectProperty rdf:about="#ifCondition">
    <rdfs:domain rdf:resource="#If-Then-Else"/>
    <rdfs:range rdf:resource="&Expression;Condition"/>
</owl:ObjectProperty>



<!-- http://web.me.com/adouglas/ontologies/catalogactivites.owl#isEncapsulatedBy -->

<owl:ObjectProperty rdf:about="#isEncapsulatedBy"/>



<!-- http://web.me.com/adouglas/ontologies/catalogactivites.owl#process -->

<owl:ObjectProperty rdf:about="#process">
    <rdfs:domain rdf:resource="#Perform"/>
    <rdfs:range rdf:resource="#Process"/>
</owl:ObjectProperty>



<!-- http://web.me.com/adouglas/ontologies/catalogactivites.owl#realizedBy -->

<owl:ObjectProperty rdf:about="#realizedBy">
    <rdfs:range rdf:resource="#AtomicProcess"/>
    <rdfs:domain rdf:resource="#SimpleProcess"/>
    <owl:inverseOf rdf:resource="#realizes"/>
</owl:ObjectProperty>
```

```
<!-- http://web.me.com/adouglas/ontologies/catalogactivites.owl#realizes -->

<owl:ObjectProperty rdf:about="#realizes">
    <rdfs:domain rdf:resource="#AtomicProcess"/>
    <rdfs:range rdf:resource="#SimpleProcess"/>
</owl:ObjectProperty>



<!-- http://web.me.com/adouglas/ontologies/catalogactivites.owl#then -->

<owl:ObjectProperty rdf:about="#then">
    <rdfs:domain rdf:resource="#If-Then-Else"/>
    <rdfs:range rdf:resource="#ProcessConstruct"/>
</owl:ObjectProperty>



<!-- http://web.me.com/adouglas/ontologies/catalogactivites.owl#untilCondition -->

<owl:ObjectProperty rdf:about="#untilCondition">
    <rdfs:domain rdf:resource="#Repeat-Until"/>
    <rdfs:range rdf:resource="&Expression;Condition"/>
</owl:ObjectProperty>



<!-- http://web.me.com/adouglas/ontologies/catalogactivites.owl#untilProcess -->

<owl:ObjectProperty rdf:about="#untilProcess">
    <rdfs:range rdf:resource="#ProcessConstruct"/>
    <rdfs:domain rdf:resource="#Repeat-Until"/>
</owl:ObjectProperty>



<!-- http://web.me.com/adouglas/ontologies/catalogactivites.owl#whileCondition -->

<owl:ObjectProperty rdf:about="#whileCondition">
    <rdfs:domain rdf:resource="#Repeat-While"/>
    <rdfs:range rdf:resource="&Expression;Condition"/>
</owl:ObjectProperty>



<!-- http://web.me.com/adouglas/ontologies/catalogactivites.owl#whileProcess -->

<owl:ObjectProperty rdf:about="#whileProcess">
    <rdfs:range rdf:resource="#ProcessConstruct"/>
    <rdfs:domain rdf:resource="#Repeat-While"/>
</owl:ObjectProperty>



<!-- http://www.daml.org/services/owl-s/1.1/generic/ObjectList.owl#first -->

<owl:ObjectProperty rdf:about="&ObjectList;first"/>
```

```
<!-- http://www.daml.org/services/owl-s/1.1/generic/ObjectList.owl#rest -->

<owl:ObjectProperty rdf:about="&ObjectList;rest"/>



<!--
///////////////////////////////////////////////////////////////////////////////////////
//
// Data properties
//
///////////////////////////////////////////////////////////////////////////////////////
 -->



<!-- http://web.me.com/adouglas/ontologies/catalogactivites.owl#activityName -->

<owl:DatatypeProperty rdf:about="#activityName">
    <rdfs:domain rdf:resource="#Activity"/>
</owl:DatatypeProperty>



<!-- http://web.me.com/adouglas/ontologies/catalogactivites.owl#name -->

<owl:DatatypeProperty rdf:about="#name">
    <rdfs:domain rdf:resource="#Process"/>
</owl:DatatypeProperty>



<!-- http://web.me.com/adouglas/ontologies/catalogactivites.owl#parameterType -->

<owl:DatatypeProperty rdf:about="#parameterType">
    <rdfs:domain rdf:resource="#Parameter"/>
    <rdfs:range rdf:resource="&xsd;anyURI"/>
</owl:DatatypeProperty>



<!-- http://web.me.com/adouglas/ontologies/catalogactivites.owl#parameterValue -->

<owl:DatatypeProperty rdf:about="#parameterValue">
    <rdfs:domain rdf:resource="#Parameter"/>
    <rdfs:range rdf:resource="&rdf;XMLLiteral"/>
</owl:DatatypeProperty>



<!--
///////////////////////////////////////////////////////////////////////////////////////
//
// Classes
//
///////////////////////////////////////////////////////////////////////////////////////
```

```
 -->



<!-- http://web.me.com/adouglas/ontologies/catalog.owl#CatalogActivities -->

<owl:Class rdf:about="&catalog;CatalogActivities"/>



<!-- http://web.me.com/adouglas/ontologies/catalogactivites.owl#Activity -->

<owl:Class rdf:about="#Activity">
    <rdfs:subClassOf rdf:resource="&catalog;CatalogActivities"/>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#activityName"/>
            <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:maxCardinality>
        </owl:Restriction>
    </rdfs:subClassOf>
</owl:Class>



<!-- http://web.me.com/adouglas/ontologies/catalogactivites.owl#Any-Order -->

<owl:Class rdf:about="#Any-Order">
    <owl:equivalentClass rdf:resource="#Unordered"/>
    <rdfs:subClassOf rdf:resource="#ProcessConstruct"/>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#components"/>
            <owl:allValuesFrom rdf:resource="#ProcessConstructBag"/>
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#components"/>
            <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
        </owl:Restriction>
    </rdfs:subClassOf>
</owl:Class>



<!-- http://web.me.com/adouglas/ontologies/catalogactivites.owl#AtomicProcess -->

<owl:Class rdf:about="#AtomicProcess">
    <rdfs:subClassOf rdf:resource="#Process"/>
    <owl:disjointWith rdf:resource="#CompositeProcess"/>
    <owl:disjointWith rdf:resource="#SimpleProcess"/>
</owl:Class>



<!-- http://web.me.com/adouglas/ontologies/catalogactivites.owl#Choice -->

<owl:Class rdf:about="#Choice">
```

```
    <rdfs:subClassOf rdf:resource="#ProcessConstruct"/>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#components"/>
            <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#components"/>
            <owl:allValuesFrom rdf:resource="#ProcessConstructBag"/>
        </owl:Restriction>
    </rdfs:subClassOf>
</owl:Class>



<!-- http://web.me.com/adouglas/ontologies/catalogactivites.owl#CompositeProcess -->

<owl:Class rdf:about="#CompositeProcess">
    <owl:equivalentClass>
        <owl:Class>
            <owl:intersectionOf rdf:parseType="Collection">
                <rdf:Description rdf:about="#Process"/>
                <owl:Restriction>
                    <owl:onProperty rdf:resource="#composedOf"/>
                    <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality
 >
                </owl:Restriction>
            </owl:intersectionOf>
        </owl:Class>
    </owl:equivalentClass>
    <rdfs:subClassOf rdf:resource="#Process"/>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#computedInput"/>
            <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:minCardinality>
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#computedOutput"/>
            <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:minCardinality>
        </owl:Restriction>
    </rdfs:subClassOf>
    <owl:disjointWith rdf:resource="#SimpleProcess"/>
</owl:Class>



<!-- http://web.me.com/adouglas/ontologies/catalogactivites.owl#If-Then-Else -->

<owl:Class rdf:about="#If-Then-Else">
    <rdfs:subClassOf rdf:resource="#ProcessConstruct"/>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#then"/>
            <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
        </owl:Restriction>
```

```
        </rdfs:subClassOf>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#else"/>
                <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:maxCardinality>
            </owl:Restriction>
        </rdfs:subClassOf>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#ifCondition"/>
                <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
            </owl:Restriction>
        </rdfs:subClassOf>
    </owl:Class>



    <!-- http://web.me.com/adouglas/ontologies/catalogactivites.owl#Input -->

    <owl:Class rdf:about="#Input">
        <rdfs:subClassOf rdf:resource="#Parameter"/>
        <owl:disjointWith rdf:resource="#Output"/>
    </owl:Class>



    <!-- http://web.me.com/adouglas/ontologies/catalogactivites.owl#Iterate -->

    <owl:Class rdf:about="#Iterate">
        <rdfs:subClassOf rdf:resource="#ProcessConstruct"/>
    </owl:Class>



    <!-- http://web.me.com/adouglas/ontologies/catalogactivites.owl#Output -->

    <owl:Class rdf:about="#Output">
        <rdfs:subClassOf rdf:resource="#Parameter"/>
    </owl:Class>



    <!-- http://web.me.com/adouglas/ontologies/catalogactivites.owl#Parameter -->

    <owl:Class rdf:about="#Parameter">
        <rdfs:subClassOf rdf:resource="&swrl;Variable"/>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#parameterType"/>
                <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:minCardinality>
            </owl:Restriction>
        </rdfs:subClassOf>
        <rdfs:comment
            >Class used to relate parameters to SWRL variables</rdfs:comment>
    </owl:Class>



    <!-- http://web.me.com/adouglas/ontologies/catalogactivites.owl#Perform -->
```

```
<owl:Class rdf:about="#Perform">
    <rdfs:subClassOf rdf:resource="#ProcessConstruct"/>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#process"/>
            <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
        </owl:Restriction>
    </rdfs:subClassOf>
</owl:Class>



<!-- http://web.me.com/adouglas/ontologies/catalogactivites.owl#Process -->

<owl:Class rdf:about="#Process">
    <owl:equivalentClass>
        <owl:Class>
            <owl:unionOf rdf:parseType="Collection">
                <rdf:Description rdf:about="#AtomicProcess"/>
                <rdf:Description rdf:about="#CompositeProcess"/>
                <rdf:Description rdf:about="#SimpleProcess"/>
            </owl:unionOf>
        </owl:Class>
    </owl:equivalentClass>
    <rdfs:subClassOf rdf:resource="&owl;Thing"/>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#name"/>
            <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:maxCardinality>
        </owl:Restriction>
    </rdfs:subClassOf>
</owl:Class>



<!-- http://web.me.com/adouglas/ontologies/catalogactivites.owl#ProcessConstruct -->

<owl:Class rdf:about="#ProcessConstruct">
    <rdfs:subClassOf rdf:resource="&owl;Thing"/>
</owl:Class>



<!-- http://web.me.com/adouglas/ontologies/catalogactivites.owl#ProcessConstructBag -->

<owl:Class rdf:about="#ProcessConstructBag">
    <rdfs:subClassOf rdf:resource="&ObjectList;List"/>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="&ObjectList;rest"/>
            <owl:allValuesFrom rdf:resource="#ProcessConstructBag"/>
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="&ObjectList;first"/>
            <owl:allValuesFrom rdf:resource="#ProcessConstruct"/>
        </owl:Restriction>
```

```
    </rdfs:subClassOf>
</owl:Class>



<!-- http://web.me.com/adouglas/ontologies/catalogactivites.owl#ProcessConstructList -->

<owl:Class rdf:about="#ProcessConstructList">
    <rdfs:subClassOf rdf:resource="&ObjectList;List"/>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="&ObjectList;rest"/>
            <owl:allValuesFrom rdf:resource="#ProcessConstructList"/>
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="&ObjectList;first"/>
            <owl:allValuesFrom rdf:resource="#ProcessConstruct"/>
        </owl:Restriction>
    </rdfs:subClassOf>
</owl:Class>



<!-- http://web.me.com/adouglas/ontologies/catalogactivites.owl#Produce -->

<owl:Class rdf:about="#Produce">
    <rdfs:subClassOf rdf:resource="#ProcessConstruct"/>
</owl:Class>



<!-- http://web.me.com/adouglas/ontologies/catalogactivites.owl#Repeat-Until -->

<owl:Class rdf:about="#Repeat-Until">
    <rdfs:subClassOf rdf:resource="#Iterate"/>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#untilCondition"/>
            <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#untilProcess"/>
            <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
        </owl:Restriction>
    </rdfs:subClassOf>
</owl:Class>



<!-- http://web.me.com/adouglas/ontologies/catalogactivites.owl#Repeat-While -->

<owl:Class rdf:about="#Repeat-While">
    <rdfs:subClassOf rdf:resource="#Iterate"/>
    <rdfs:subClassOf>
        <owl:Restriction>
```

```
            <owl:onProperty rdf:resource="#whileProcess"/>
            <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#whileCondition"/>
            <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
        </owl:Restriction>
    </rdfs:subClassOf>
</owl:Class>



<!-- http://web.me.com/adouglas/ontologies/catalogactivites.owl#Sequence -->

<owl:Class rdf:about="#Sequence">
    <rdfs:subClassOf rdf:resource="#ProcessConstruct"/>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#components"/>
            <owl:allValuesFrom rdf:resource="#ProcessConstructList"/>
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#components"/>
            <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
        </owl:Restriction>
    </rdfs:subClassOf>
</owl:Class>



<!-- http://web.me.com/adouglas/ontologies/catalogactivites.owl#SimpleProcess -->

<owl:Class rdf:about="#SimpleProcess">
    <rdfs:subClassOf rdf:resource="#Process"/>
</owl:Class>



<!-- http://web.me.com/adouglas/ontologies/catalogactivites.owl#Split -->

<owl:Class rdf:about="#Split">
    <rdfs:subClassOf rdf:resource="#ProcessConstruct"/>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#components"/>
            <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#components"/>
            <owl:allValuesFrom rdf:resource="#ProcessConstructBag"/>
        </owl:Restriction>
    </rdfs:subClassOf>
</owl:Class>
```

```
<!-- http://web.me.com/adouglas/ontologies/catalogactivites.owl#Split-Join -->

<owl:Class rdf:about="#Split-Join">
    <rdfs:subClassOf rdf:resource="#ProcessConstruct"/>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#components"/>
            <owl:allValuesFrom rdf:resource="#ProcessConstructBag"/>
        </owl:Restriction>
    </rdfs:subClassOf>
</owl:Class>



<!-- http://web.me.com/adouglas/ontologies/catalogactivites.owl#Unordered -->

<owl:Class rdf:about="#Unordered">
    <rdfs:subClassOf rdf:resource="#ProcessConstruct"/>
</owl:Class>



<!-- http://www.daml.org/services/owl-s/1.1/generic/Expression.owl#Condition -->

<owl:Class rdf:about="&Expression;Condition"/>



<!-- http://www.daml.org/services/owl-s/1.1/generic/ObjectList.owl#List -->

<owl:Class rdf:about="&ObjectList;List"/>



<!-- http://www.w3.org/2002/07/owl#Thing -->

<owl:Class rdf:about="&owl;Thing"/>



<!-- http://www.w3.org/2003/11/swrl#Variable -->

<owl:Class rdf:about="&swrl;Variable"/>



<!--
///////////////////////////////////////////////////////////////////////////////////
//
// Individuals
//
///////////////////////////////////////////////////////////////////////////////////
 -->
```

```
    <!-- http://web.me.com/adouglas/ontologies/catalogactivites.owl#TheParentPerform -->

    <owl:Thing rdf:about="#TheParentPerform">
        <rdf:type rdf:resource="#Perform"/>
    </owl:Thing>



    <!-- http://web.me.com/adouglas/ontologies/catalogactivites.owl#ThisPerform -->

    <Perform rdf:about="#ThisPerform">
        <rdf:type rdf:resource="&owl;Thing"/>
    </Perform>
</rdf:RDF>



<!-- Generated by the OWL API (version 2.2.1.1138) http://owlapi.sourceforge.net -->
```

A.2.4 Role.owl

```
<?xml version="1.0"?>


<!DOCTYPE rdf:RDF [
    <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
    <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
    <!ENTITY owl2xml "http://www.w3.org/2006/12/owl2-xml#" >
    <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
    <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
    <!ENTITY role "http://public.me.com/adouglas/ontologies/role.owl#" >
    <!ENTITY catalogprofile "http://web.me.com/adouglas/ontologies/catalogprofile.owl#" >
]>


<rdf:RDF xmlns="http://public.me.com/adouglas/ontologies/role.owl#"
     xml:base="http://public.me.com/adouglas/ontologies/role.owl"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns:owl2xml="http://www.w3.org/2006/12/owl2-xml#"
     xmlns:catalogprofile="http://web.me.com/adouglas/ontologies/catalogprofile.owl#"
     xmlns:owl="http://www.w3.org/2002/07/owl#"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:role="http://public.me.com/adouglas/ontologies/role.owl#"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
    <owl:Ontology rdf:about="">
        <owl:versionInfo
            >$Id: role.owl,v 0.1  $</owl:versionInfo>
    </owl:Ontology>



    <!--
    ///////////////////////////////////////////////////////////////////////////////////
    //
    // Object Properties
    //
```

```
//////////////////////////////////////////////////////////////////////////////////
 -->



<!-- http://public.me.com/adouglas/ontologies/role.owl#category -->

<owl:ObjectProperty rdf:about="#category">
    <rdfs:range rdf:resource="#BusinessCategory"/>
    <rdfs:domain rdf:resource="#Role"/>
</owl:ObjectProperty>



<!-- http://web.me.com/adouglas/ontologies/catalogprofile.owl#hasRole -->

<owl:ObjectProperty rdf:about="&catalogprofile;hasRole">
    <rdfs:range rdf:resource="#Role"/>
</owl:ObjectProperty>



<!--
//////////////////////////////////////////////////////////////////////////////////
//
// Data properties
//
//////////////////////////////////////////////////////////////////////////////////
 -->



<!-- http://public.me.com/adouglas/ontologies/role.owl#categoryName -->

<owl:DatatypeProperty rdf:about="#categoryName">
    <rdfs:domain rdf:resource="#BusinessCategory"/>
</owl:DatatypeProperty>



<!-- http://public.me.com/adouglas/ontologies/role.owl#code -->

<owl:DatatypeProperty rdf:about="#code">
    <rdfs:domain rdf:resource="#BusinessCategory"/>
</owl:DatatypeProperty>



<!-- http://public.me.com/adouglas/ontologies/role.owl#taxonomy -->

<owl:DatatypeProperty rdf:about="#taxonomy">
    <rdfs:domain rdf:resource="#BusinessCategory"/>
</owl:DatatypeProperty>



<!-- http://public.me.com/adouglas/ontologies/role.owl#type -->
```

```
<owl:DatatypeProperty rdf:about="#type">
    <rdfs:domain rdf:resource="#Role"/>
    <rdfs:range rdf:resource="&rdfs;Literal"/>
</owl:DatatypeProperty>



<!-- http://public.me.com/adouglas/ontologies/role.owl#value -->

<owl:DatatypeProperty rdf:about="#value">
    <rdfs:domain rdf:resource="#BusinessCategory"/>
</owl:DatatypeProperty>



<!--
///////////////////////////////////////////////////////////////////////////////////////
//
// Classes
//
///////////////////////////////////////////////////////////////////////////////////////
 -->




<!-- http://public.me.com/adouglas/ontologies/role.owl#BusinessCategory -->

<owl:Class rdf:about="#BusinessCategory">
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#taxonomy"/>
            <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#value"/>
            <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#categoryName"/>
            <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#code"/>
            <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
        </owl:Restriction>
    </rdfs:subClassOf>
</owl:Class>



<!-- http://public.me.com/adouglas/ontologies/role.owl#Role -->
```

```
    <owl:Class rdf:about="#Role">
        <rdfs:subClassOf rdf:resource="&owl;Thing"/>
    </owl:Class>



    <!-- http://www.w3.org/2002/07/owl#Thing -->

    <owl:Class rdf:about="&owl;Thing"/>
</rdf:RDF>



<!-- Generated by the OWL API (version 2.2.1.1138) http://owlapi.sourceforge.net -->
```

A.2.5 CatalogNorms.owl

```
<?xml version="1.0"?>


<!DOCTYPE rdf:RDF [
    <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
    <!ENTITY swrl "http://www.w3.org/2003/11/swrl#" >
    <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
    <!ENTITY owl2xml "http://www.w3.org/2006/12/owl2-xml#" >
    <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
    <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
    <!ENTITY role "http://public.me.com/adouglas/ontologies/role.owl#" >
    <!ENTITY catalog "http://web.me.com/adouglas/ontologies/catalog.owl#" >
    <!ENTITY catalognorms "http://web.me.com/adouglas/ontologies/catalognorms.owl#" >
    <!ENTITY catalogactivites "http://web.me.com/adouglas/ontologies/catalogactivites.owl#" >
    <!ENTITY catalogException "http://web.me.com/adouglas/ontologies/catalogException.owl#" >
    <!ENTITY Expression "http://www.daml.org/services/owl-s/1.1/generic/Expression.owl#" >
]>


<rdf:RDF xmlns="http://web.me.com/adouglas/ontologies/catalognorms.owl#"
     xml:base="http://web.me.com/adouglas/ontologies/catalognorms.owl"
     xmlns:catalog="http://web.me.com/adouglas/ontologies/catalog.owl#"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns:catalognorms="http://web.me.com/adouglas/ontologies/catalognorms.owl#"
     xmlns:swrl="http://www.w3.org/2003/11/swrl#"
     xmlns:owl2xml="http://www.w3.org/2006/12/owl2-xml#"
     xmlns:catalogactivites="http://web.me.com/adouglas/ontologies/catalogactivites.owl#"
     xmlns:catalogException="http://web.me.com/adouglas/ontologies/catalogException.owl#"
     xmlns:owl="http://www.w3.org/2002/07/owl#"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:role="http://public.me.com/adouglas/ontologies/role.owl#"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:Expression="http://www.daml.org/services/owl-s/1.1/generic/Expression.owl#">
    <owl:Ontology rdf:about="">
        <owl:imports rdf:resource="http://public.me.com/adouglas/ontologies/role.owl"/>
        <owl:imports rdf:resource="http://web.me.com/adouglas/ontologies/catalog.owl"/>
        <owl:imports rdf:resource="http://web.me.com/adouglas/ontologies/catalogException.owl"/>
        <owl:imports rdf:resource="http://web.me.com/adouglas/ontologies/catalogactivites.owl"/>
    </owl:Ontology>
```

```
<!--
///////////////////////////////////////////////////////////////////////////////////
//
// Object Properties
//
///////////////////////////////////////////////////////////////////////////////////
 -->



<!-- http://web.me.com/adouglas/ontologies/catalognorms.owl#governsActivity -->

<owl:ObjectProperty rdf:about="#governsActivity">
    <rdfs:range rdf:resource="&catalogactivites;Activity"/>
    <rdfs:domain rdf:resource="#Norm"/>
    <rdfs:subPropertyOf rdf:resource="#hasElement"/>
</owl:ObjectProperty>



<!-- http://web.me.com/adouglas/ontologies/catalognorms.owl#hasDCondition -->

<owl:ObjectProperty rdf:about="#hasDCondition">
    <rdfs:domain rdf:resource="#Norm"/>
    <rdfs:subPropertyOf rdf:resource="#hasElement"/>
    <rdfs:range rdf:resource="&Expression;Condition"/>
</owl:ObjectProperty>



<!-- http://web.me.com/adouglas/ontologies/catalognorms.owl#hasElement -->

<owl:ObjectProperty rdf:about="#hasElement"/>



<!-- http://web.me.com/adouglas/ontologies/catalognorms.owl#hasInCondition -->

<owl:ObjectProperty rdf:about="#hasInCondition">
    <rdfs:domain rdf:resource="#Norm"/>
    <rdfs:subPropertyOf rdf:resource="#hasElement"/>
    <rdfs:range rdf:resource="&Expression;Condition"/>
</owl:ObjectProperty>



<!-- http://web.me.com/adouglas/ontologies/catalognorms.owl#hasRole -->

<owl:ObjectProperty rdf:about="#hasRole">
    <rdfs:range rdf:resource="&role;Role"/>
    <rdfs:domain rdf:resource="#Norm"/>
    <rdfs:subPropertyOf rdf:resource="#hasElement"/>
</owl:ObjectProperty>
```

```
<!-- http://web.me.com/adouglas/ontologies/catalognorms.owl#triggersException -->

<owl:ObjectProperty rdf:about="#triggersException">
    <rdfs:range rdf:resource="&catalogException;Exception"/>
    <rdfs:domain rdf:resource="#Norm"/>
    <rdfs:subPropertyOf rdf:resource="#hasElement"/>
</owl:ObjectProperty>



<!--
///////////////////////////////////////////////////////////////////////////////////////
//
// Data properties
//
///////////////////////////////////////////////////////////////////////////////////////
 -->

<!-- http://web.me.com/adouglas/ontologies/catalognorms.owl#normIdentifer -->

<owl:DatatypeProperty rdf:about="#normIdentifer">
    <rdfs:domain rdf:resource="#Norm"/>
</owl:DatatypeProperty>



<!--
///////////////////////////////////////////////////////////////////////////////////////
//
// Classes
//
///////////////////////////////////////////////////////////////////////////////////////
 -->




<!-- http://public.me.com/adouglas/ontologies/role.owl#Role -->

<owl:Class rdf:about="&role;Role"/>



<!-- http://web.me.com/adouglas/ontologies/catalog.owl#CatalogNorms -->

<owl:Class rdf:about="&catalog;CatalogNorms"/>



<!-- http://web.me.com/adouglas/ontologies/catalogException.owl#Exception -->

<owl:Class rdf:about="&catalogException;Exception"/>



<!-- http://web.me.com/adouglas/ontologies/catalogactivites.owl#Activity -->

<owl:Class rdf:about="&catalogactivites;Activity"/>
```

```
<!-- http://web.me.com/adouglas/ontologies/catalognorms.owl#Forb -->

<owl:Class rdf:about="#Forb">
    <rdfs:label>forb</rdfs:label>
    <rdfs:subClassOf rdf:resource="#Norm"/>
    <owl:disjointWith rdf:resource="#Obli"/>
    <owl:disjointWith rdf:resource="#Perm"/>
</owl:Class>



<!-- http://web.me.com/adouglas/ontologies/catalognorms.owl#Norm -->

<owl:Class rdf:about="#Norm">
    <rdfs:label>Norm</rdfs:label>
    <owl:equivalentClass>
        <owl:Class>
            <owl:unionOf rdf:parseType="Collection">
                <rdf:Description rdf:about="#Forb"/>
                <rdf:Description rdf:about="#Obli"/>
                <rdf:Description rdf:about="#Perm"/>
            </owl:unionOf>
        </owl:Class>
    </owl:equivalentClass>
    <rdfs:subClassOf rdf:resource="&catalog;CatalogNorms"/>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#normIdentifer"/>
            <owl:someValuesFrom rdf:resource="&rdfs;Literal"/>
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:comment
        >Base representation of a normative fact</rdfs:comment>
</owl:Class>



<!-- http://web.me.com/adouglas/ontologies/catalognorms.owl#Obli -->

<owl:Class rdf:about="#Obli">
    <rdfs:label>obli</rdfs:label>
    <rdfs:subClassOf rdf:resource="#Norm"/>
    <owl:disjointWith rdf:resource="#Perm"/>
</owl:Class>



<!-- http://web.me.com/adouglas/ontologies/catalognorms.owl#Perm -->

<owl:Class rdf:about="#Perm">
    <rdfs:label>perm</rdfs:label>
    <rdfs:subClassOf rdf:resource="#Norm"/>
</owl:Class>
```

```
    <!-- http://web.me.com/adouglas/ontologies/catalognorms.owl#Result -->

    <owl:Class rdf:about="#Result">
        <rdfs:label>Result</rdfs:label>
    </owl:Class>



    <!-- http://www.daml.org/services/owl-s/1.1/generic/Expression.owl#Condition -->

    <owl:Class rdf:about="&Expression;Condition"/>
</rdf:RDF>



<!-- Generated by the OWL API (version 2.2.1.1138) http://owlapi.sourceforge.net -->
```

A.2.6 CatalogException.owl

```
<?xml version="1.0"?>


<!DOCTYPE rdf:RDF [
    <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
    <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
    <!ENTITY owl2xml "http://www.w3.org/2006/12/owl2-xml#" >
    <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
    <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
    <!ENTITY catalog "http://web.me.com/adouglas/ontologies/catalog.owl#" >
    <!ENTITY catalogException "http://web.me.com/adouglas/ontologies/catalogException.owl#" >
]>


<rdf:RDF xmlns="http://web.me.com/adouglas/ontologies/catalogException.owl#"
     xml:base="http://web.me.com/adouglas/ontologies/catalogException.owl"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns:owl2xml="http://www.w3.org/2006/12/owl2-xml#"
     xmlns:catalog="http://web.me.com/adouglas/ontologies/catalog.owl#"
     xmlns:catalogException="http://web.me.com/adouglas/ontologies/catalogException.owl#"
     xmlns:owl="http://www.w3.org/2002/07/owl#"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
    <owl:Ontology rdf:about="">
        <owl:imports rdf:resource="http://web.me.com/adouglas/ontologies/catalog.owl"/>
    </owl:Ontology>



    <!--
    ///////////////////////////////////////////////////////////////////////////////////////
    //
    // Object Properties
    //
    ///////////////////////////////////////////////////////////////////////////////////////
     -->
```

```
<!-- http://web.me.com/adouglas/ontologies/catalogException.owl#hasPenalty -->

<owl:ObjectProperty rdf:about="#hasPenalty">
    <rdfs:domain rdf:resource="#Exception"/>
</owl:ObjectProperty>



<!-- http://web.me.com/adouglas/ontologies/catalogException.owl#triggers -->

<owl:ObjectProperty rdf:about="#triggers">
    <rdfs:domain rdf:resource="#Exception"/>
</owl:ObjectProperty>



<!--
///////////////////////////////////////////////////////////////////////////////////////
//
// Data properties
//
///////////////////////////////////////////////////////////////////////////////////////
 -->




<!-- http://web.me.com/adouglas/ontologies/catalogException.owl#exceptionIdentifier -->

<owl:DatatypeProperty rdf:about="#exceptionIdentifier">
    <rdfs:domain rdf:resource="#Exception"/>
</owl:DatatypeProperty>



<!-- http://web.me.com/adouglas/ontologies/catalogException.owl#penalty -->

<owl:DatatypeProperty rdf:about="#penalty">
    <rdfs:domain rdf:resource="#Penalty"/>
    <rdfs:range rdf:resource="&xsd;integer"/>
</owl:DatatypeProperty>



<!-- http://web.me.com/adouglas/ontologies/catalogException.owl#recoverable -->

<owl:DatatypeProperty rdf:about="#recoverable">
    <rdfs:domain rdf:resource="#Exception"/>
    <rdfs:range rdf:resource="&xsd;boolean"/>
</owl:DatatypeProperty>



<!--
///////////////////////////////////////////////////////////////////////////////////////
//
// Classes
```

```
    //
    ///////////////////////////////////////////////////////////////////////////////////
     -->



    <!-- http://web.me.com/adouglas/ontologies/catalog.owl#CatalogExceptions -->

    <owl:Class rdf:about="&catalog;CatalogExceptions"/>



    <!-- http://web.me.com/adouglas/ontologies/catalogException.owl#Exception -->

    <owl:Class rdf:about="#Exception">
        <rdfs:subClassOf rdf:resource="&catalog;CatalogExceptions"/>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#recoverable"/>
                <owl:maxQualifiedCardinality rdf:datatype="&xsd;nonNegativeInteger">1</
     owl:maxQualifiedCardinality>
                <owl:onDataRange rdf:resource="&xsd;boolean"/>
            </owl:Restriction>
        </rdfs:subClassOf>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#exceptionIdentifier"/>
                <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:maxCardinality>
            </owl:Restriction>
        </rdfs:subClassOf>
    </owl:Class>



    <!-- http://web.me.com/adouglas/ontologies/catalogException.owl#Penalty -->

    <owl:Class rdf:about="#Penalty">
        <rdfs:subClassOf rdf:resource="&catalog;CatalogExceptions"/>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#penalty"/>
                <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
            </owl:Restriction>
        </rdfs:subClassOf>
    </owl:Class>
</rdf:RDF>



<!-- Generated by the OWL API (version 2.2.1.1138) http://owlapi.sourceforge.net -->
```

A.3 NPE-L Examples

A.3.1 ExampleCatalog.owl

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<!DOCTYPE uridef[
  <!ENTITY rdf     "http://www.w3.org/1999/02/22-rdf-syntax-ns">
  <!ENTITY rdfs    "http://www.w3.org/2000/01/rdf-schema">
  <!ENTITY activities     "http://web.me.com/adouglas/ontologies/examples/ExampleActivities.owl">
  <!ENTITY profile     "http://web.me.com/adouglas/ontologies/examples/ExampleProfile.owl">
  <!ENTITY norms     "http://web.me.com/adouglas/ontologies/examples/ExampleNorms.owl">
  <!ENTITY exceptions     "http://web.me.com/adouglas/ontologies/examples/ExampleExceptions.owl">
]>

<rdf:RDF xml:base="http://web.me.com/adouglas/ontologies/examples/ExampleCatalog.owl">
        <owl:Ontology rdf:about="">
                <owl:versionInfo>
                         $Id: ExampleCatalog.owl V1.2 13/01/10 $
                </owl:versionInfo>

                <rdfs:comment>
                   This ontology represents an NPE-L example catalog.
                </rdfs:comment>

                <owl:imports rdf:resource="http://web.me.com/adouglas/ontologies/catalog.owl"/>
                <owl:imports rdf:resource="http://web.me.com/adouglas/ontologies/catalogprofile.
    owl"/>
                <owl:imports rdf:resource="http://web.me.com/adouglas/ontologies/catalogactivities
    .owl"/>
                <owl:imports rdf:resource="http://web.me.com/adouglas/ontologies/catalognorms.owl"
    />
                <owl:imports rdf:resource="http://web.me.com/adouglas/ontologies/catalogexceptions
    .owl"/>
                <owl:imports rdf:resource="&profile;#"/>
                <owl:imports rdf:resource="&activities;#"/>
                <owl:imports rdf:resource="&norms;#"/>
                <owl:imports rdf:resource="&exceptions;#"/>
        </owl:Ontology>


        <catalog:Catalog rdf:ID="ExampleCatalog">
                <!-- Reference to the Profile -->
                <catalog:presents rdf:resource="&profile;#ExampleProfile"/>
                <!-- Reference to the Activities involved -->
                <catalog:governs rdf:resource="&activities;#ExampleCatalog_Activity"/>
                <!-- Reference to the Norms -->
                <catalog:provides rdf:resource="&norms;#ExampleCatalog_ObliPutInCart"/>
                <!-- Reference to the Exceptions -->
                <catalog:administers rdf:resource="&exceptions;#ExampleCatalog_NoItemsException"/>
        </catalog:Catalog>
</rdf:RDF>
```

A.3.2 ExampleProfile.owl

```
<?xml version='1.0' encoding='ISO-8859-1'?>
```

```
<!DOCTYPE uridef[
  <!ENTITY rdf     "http://www.w3.org/1999/02/22-rdf-syntax-ns">
  <!ENTITY rdfs    "http://www.w3.org/2000/01/rdf-schema">
  <!ENTITY xsd     "http://www.w3.org/2001/XMLSchema">
  <!ENTITY catalog     "http://web.me.com/adouglas/ontologies/examples/ExampleCatalog.owl">
  <!ENTITY roles     "http://web.me.com/adouglas/ontologies/examples/ExampleRoles.owl">
  <!ENTITY norms     "http://web.me.com/adouglas/ontologies/examples/ExampleNorms.owl">
]>

<rdf:RDF xml:base="http://web.me.com/adouglas/ontologies/examples/ExampleProfile.owl">
        <owl:Ontology rdf:about="">
                <owl:versionInfo>
                          $Id: ExampleProfile.owl v1.5 13/01/10 $
                </owl:versionInfo>
                <rdfs:comment>
                          NPE-L example profile.
                </rdfs:comment>
                <owl:imports rdf:resource="http://web.me.com/adouglas/ontologies/catalog.owl"/>
                <owl:imports rdf:resource="http://web.me.com/adouglas/ontologies/catalogprofile.
     owl"/>
                <owl:imports rdf:resource="http://web.me.com/adouglas/ontologies/role.owl"/>
                <owl:imports rdf:resource="http://web.me.com/adouglas/ontologies/catalogactivities
     .owl"/>
                <owl:imports rdf:resource="http://web.me.com/adouglas/ontologies/catalognorms.owl"
     />
                <owl:imports rdf:resource="http://web.me.com/adouglas/ontologies/catalogexceptions
     .owl"/>
                <owl:imports rdf:resource="&catalog;#"/>
                <owl:imports rdf:resource="&roles;#"/>
                <owl:imports rdf:resource="&norms;#"/>
        </owl:Ontology>

<profile:Profile rdf:ID="ExampleProfile">
        <!-- reference to the service specification -->
        <catalog:presentedBy rdf:resource="&catalog;#ExampleCatalog"/>

        <profile:textDescription>
                        The catalog provides an example of how NPE-L might be deployed in a real
     world situation.
        </profile:textDescription>

        <profile:contactInformation>
                <actor:Actor rdf:ID="ExampleCatalog_Contacts">
                <actor:name>Example Owner</actor:name>
                <actor:title> Service Representative </actor:title>
                <actor:email>act@or.com</actor:email>
                </actor:Actor>
        </profile:contactInformation>

        <profile:serviceCategory>
                <role:BusinessCategory rdf:ID="NAICS-category">
                        <role:value>
                                E-tailers
                        </role:value>
                        <role:code>
                                454111
                        </role:code>
                </role:BusinessCategory >
        </profile:serviceCategory>
```

```
        <profile:hasRole rdf:resource="&roles;#ExampleServerRole"/>
        <profile:hasRole rdf:resource="&roles;#ExampleClientRole"/>

        <profile:hasNorm rdf:resource="&norms;#ExampleNorm"/>
</profile:Profile>
</rdf:RDF>
```

A.3.3 ExampleRoles.owl

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<!DOCTYPE uridef[
  <!ENTITY rdf     "http://www.w3.org/1999/02/22-rdf-syntax-ns">
  <!ENTITY rdfs    "http://www.w3.org/2000/01/rdf-schema">
  <!ENTITY xsd     "http://www.w3.org/2001/XMLSchema">
]>

<rdf:RDF xml:base="http://web.me.com/adouglas/ontologies/examples/ExampleRoles.owl">
        <owl:Ontology rdf:about="">
                <owl:versionInfo>
                         $Id: ExampleRoles.owl V1.0 30/11/09 $
                </owl:versionInfo>

                <rdfs:comment>
                   This ontology represents the roles to be used in the example catalog.
                </rdfs:comment>

                <owl:imports rdf:resource="http://web.me.com/adouglas/ontologies/role.owl"/>
        </owl:Ontology>


        <role:Role rdf:ID="Example_ServerRole">
                <role:Type>Server</role:Type>
                <role:BusinessCategory rdf:ID="NAICS-category">
                        <role:value>
                                E-tailers
                        </role:value>
                        <role:code>
                                454111
                        </role:code>
                </role:BusinessCategory >
        </role:Role>

        <role:Role rdf:ID="Example_ClientRole">
                <role:roleIdentifier>
                        user
                </role:roleIdentifier>
                <role:Type>Client</role:Type>
                        <role:BusinessCategory rdf:ID="NAICS-category">
                        <role:value>
                                University
                        </role:value>
                        <role:code>
                                611310
                        </role:code>
                        </role:BusinessCategory >
        </role:Role>
```

```
</rdf:RDF>
```

A.3.4 ExampleActivities.owl

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<!DOCTYPE uridef[
  <!ENTITY rdf     "http://www.w3.org/1999/02/22-rdf-syntax-ns">
  <!ENTITY rdfs    "http://www.w3.org/2000/01/rdf-schema">
  <!ENTITY xsd     "http://www.w3.org/2001/XMLSchema">
  <!ENTITY owl     "http://www.w3.org/2002/07/owl">
  <!ENTITY activities     "http://web.me.com/adouglas/ontologies/examples/ExampleActivities.owl">
]>

<rdf:RDF xml:base="http://web.me.com/adouglas/ontologies/examples/ExampleActivities.owl">
        <owl:Ontology rdf:about="">
                <owl:versionInfo>
                          $Id: ExampleActivities.owl v1.4 13/01/10 $
                </owl:versionInfo>
                <rdfs:comment>
                          Activities governed by norms in the example catalog.
                </rdfs:comment>

                <owl:imports rdf:resource="http://web.me.com/adouglas/ontologies/catalogactivities
     .owl"/>
                <owl:imports rdf:resource="&activities;#"/>
        </owl:Ontology>

        <activity:Activity rdf:ID="ExampleCatalog_Activity">
                <activity:activityName>
                        Item_Purchase
                </activity:activityName>
                <activity:hasInput rdf:ID="ItemCodeInput">
                        <activity:Input>
                                <activity:parameterType rdf:datatype="&xsd;#anyURI">
                                        #ItemCode
                                </activity:parameterType>
                        </activity:Input>
                </activity:hasInput>
                <activity:hasOutput rdf:ID="FrangoOutput">
                        <activity:Output>
                                <activity:parameterType rdf:datatype="&xsd;#anyURI">
                                        #BuyOutputType
                                </activity:parameterType>
                        </activity:Output>
                </activity:hasOutput>
        </activity:Activity>

        <owl:Class rdf:ID="BuyOutputType"/>
        <owl:ObjectProperty rdf:ID="hasFrangoBuyOutputType">
                <rdf:type rdf:resource="&owl;#FunctionalProperty"/>
                <rdfs:domain rdf:resource="#BuyOutputType"/>
                <rdfs:range rdf:resource="&xsd;#string"/>
        </owl:ObjectProperty>

        <owl:Class rdf:ID="Product">
                <rdfs:comment>Top level class for a product</rdfs:comment>
        </owl:Class>
```

```
        <owl:Class rdf:ID="ItemCode"/>

        <owl:Class rdf:ID="Vendible">
                <rdfs:subClassOf rdf:resource="#Product"/>
        </owl:Class>

        <owl:DatatypeProperty rdf:ID="identifier">
                <rdfs:domain rdf:resource="#Vendible"/>
                <rdfs:range rdf:resource="&xsd;#string"/>
        </owl:DatatypeProperty>

        <owl:DatatypeProperty rdf:ID="manufacturer">
                <rdfs:domain rdf:resource="#Vendible"/>
                <rdfs:range rdf:resource="&xsd;#string"/>
        </owl:DatatypeProperty>

        <owl:ObjectProperty rdf:ID="hasItemCode">
                <rdfs:domain rdf:resource="#Vendible"/>
                <rdfs:range rdf:resource="#ItemCode"/>
        </owl:ObjectProperty>
</rdf:RDF>
```

A.3.5 ExampleNorms.owl

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<!DOCTYPE uridef[
  <!ENTITY rdf     "http://www.w3.org/1999/02/22-rdf-syntax-ns">
  <!ENTITY rdfs    "http://www.w3.org/2000/01/rdf-schema">
  <!ENTITY xsd     "http://www.w3.org/2001/XMLSchema">
  <!ENTITY expr     "http://www.daml.org/services/owl-s/1.1/generic/Expression.owl">
  <!ENTITY catalog     "http://web.me.com/adouglas/ontologies/examples/ExampleCatalog.owl">
  <!ENTITY activities     "http://web.me.com/adouglas/ontologies/examples/ExampleActivities.owl">
  <!ENTITY roles     "http://web.me.com/adouglas/ontologies/examples/ExampleRoles.owl">
  <!ENTITY exceptions     "http://web.me.com/adouglas/ontologies/examples/ExampleExceptions.owl">
]>

<rdf:RDF xml:base="http://web.me.com/adouglas/ontologies/examples/ExampleNorms.owl">
        <owl:Ontology rdf:about="">
                <owl:versionInfo>
                         $Id: ExampleNorms.owl V1.1 13/01/10 $
                </owl:versionInfo>

                <rdfs:comment>
                   This ontology represents an example norm.
                </rdfs:comment>

                <owl:imports rdf:resource="http://web.me.com/adouglas/ontologies/catalogprofile.
    owl"/>
                <owl:imports rdf:resource="http://web.me.com/adouglas/ontologies/role.owl"/>
                <owl:imports rdf:resource="http://web.me.com/adouglas/ontologies/catalogactivities
    .owl"/>
                <owl:imports rdf:resource="http://web.me.com/adouglas/ontologies/catalognorms.owl"
    />
                <owl:imports rdf:resource="http://web.me.com/adouglas/ontologies/catalogexceptions
    .owl"/>
                <owl:imports rdf:resource="&activities;#"/>
```

```
            <owl:imports rdf:resource="&roles;#"/>
            <owl:imports rdf:resource="&norms;#"/>
            <owl:imports rdf:resource="&exceptions;#"/>
        </owl:Ontology>


        <norm:Obli rdf:ID="ExampleCatalog_ObliPutInCart">
                <norm:governsActivity rdf:resource="&activities;#ExampleCatalog_Activity" />
                <norm:hasRole rdf:resource="&roles;#Example_ClientRole" />
                <norm:hasInCondition>
                        <expr:SWRL-Condition>
                                <rdfs:label>
                                        in(#Buy_Sequence)
                                </rdfs:label>
                                <rdfs:comment>
                                        Test to see if at the time this norm is applied the agent
    believes it will be in the #Buy_Sequence. This is equivelent to the expression: in_region(
    user, #Buy_Sequence).
                                </rdfs:comment>
                                <expr:expressionLanguage rdf:resource="&expr;#SWRL" />
                                <expr:expressionBody rdf:parseType="Literal">
                                        <swrl:IndividualPropertyAtom>
                                                <swrl:propertyPredicate rdf:resource="#in_region"
    />
                                                <swrl:argument1 rdf:resource="&roles;#
    Example_ClientRole" />
                                                <swrl:argument2 rdf:resource="&activities;#
    ExampleCatalog_Activity_BuySequence" />
                                        </swrl:IndividualPropertyAtom>
                                </expr:expressionBody>
                        </expr:SWRL-Condition>
                </norm:hasInCondition>
                <norm:hasDCondition>
                        <expr:SWRL-Condition>
                                <rdfs:label>
                                        once(#PutItemInCart)
                                </rdfs:label>
                                <rdfs:comment>
                                        Test to see if at the time this norm is active that it
    will be deactivated by the following condition. This is equivelent to the expression: once(
    user, #PutInCart).
                                </rdfs:comment>
                                <expr:expressionLanguage rdf:resource="&expr;#SWRL" />
                                <expr:expressionBody rdf:parseType="Literal">
                                        <swrl:IndividualPropertyAtom>
                                                <swrl:propertyPredicate rdf:resource="#once" />
                                                <swrl:argument1 rdf:resource="&roles;#
    Example_ClientRole" />
                                                <swrl:argument2 rdf:resource="&activities;#
    ExampleCatalog_Activity_PutItemInCart" />
                                        </swrl:IndividualPropertyAtom>
                                </expr:expressionBody>
                        </expr:SWRL-Condition>
                </norm:hasDContition>
                <norm:triggersException rdf:resource="&roles;#ExampleCatalog_NoItemsException" />
        </norm:Obli>

</rdf:RDF>
```

A.3.6 ExampleExceptions.owl

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<!DOCTYPE uridef[
  <!ENTITY rdf     "http://www.w3.org/1999/02/22-rdf-syntax-ns">
  <!ENTITY rdfs    "http://www.w3.org/2000/01/rdf-schema">
  <!ENTITY xsd     "http://www.w3.org/2001/XMLSchema">
  <!ENTITY exceptions     "http://web.me.com/adouglas/ontologies/examples/ExampleExceptions.owl">
]>

<rdf:RDF xml:base="http://web.me.com/adouglas/ontologies/examples/ExampleExceptions.owl">
        <owl:Ontology rdf:about="">
                <owl:versionInfo>
                         $Id: ExampleExceptions.owl V1.1 13/01/10 $
                </owl:versionInfo>

                <rdfs:comment>
                        This ontology represents the exceptions which may be triggered by norms in
     the example catalog.
                </rdfs:comment>

                <owl:imports rdf:resource="http://web.me.com/adouglas/ontologies/catalogexceptions
    .owl"/>
                <owl:imports rdf:resource="&exceptions;#"/>
        </owl:Ontology>


        <exception:Exception rdf:ID="ExampleCatalog_NoItemsException">
                <exception:Recoverable>
                        true
                </exception:Recoverable>
                <exception:hasPenalty>
                        <penalty:Penalty ref:datatype="&xsd;#Integer">
                                25
                        </penalty:Penalty>
                </exception:hasPenalty>
        </exception:Exception>
</rdf:RDF>
```

A.4 NPE-L Test Cases

A.4.1 globalprofile.npel

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE uridef[
  <!ENTITY rdf     "http://www.w3.org/1999/02/22-rdf-syntax-ns">
  <!ENTITY rdfs    "http://www.w3.org/2000/01/rdf-schema">
  <!ENTITY owl     "http://www.w3.org/2002/07/owl">
  <!ENTITY xsd     "http://www.w3.org/2001/XMLSchema">
  <!ENTITY actor "http://www.daml.org/services/owl-s/1.2/ActorDefault.owl">
  <!ENTITY profile "http://web.me.com/adouglas/ontologies/npel/1.3/profile.owl">
  <!ENTITY THIS    "http://web.me.com/adouglas/npel/globalprofile.npel">
]>

<rdf:RDF        xmlns:rdf = "&rdf;#"
```

```
                        xmlns:rdfs = "&rdfs;#"
                        xmlns:xsd = "&xsd;#"
                        xmlns:owl = "&owl;#"
                        xmlns:actor = "&actor;#"
                        xmlns:profile = "&profile;#"
                        xmlns = "&THIS;#"
                        xml:base = "&THIS;">
        <owl:Ontology rdf:about="">
                <owl:versionInfo>
                          $Id: globalprofile.owl v1.1 13/01/10 $
                </owl:versionInfo>
                <rdfs:comment></rdfs:comment>

                <owl:imports rdf:resource="&profile;#"/>
        </owl:Ontology>

<profile:Profile rdf:ID="Profile">
        <catalog:presentedBy rdf:resource="&catalog;#ExampleCatalog"/>

        <profile:textDescription>
                        The catalog provides an example of how NPE-L might be deployed in a real
     world situation.
        </profile:textDescription>

        <profile:contactInformation>
                <actor:Actor rdf:ID="Contact">
                <actor:name>Example Owner</actor:name>
                <actor:title> Service Representative </actor:title>
                <actor:email>act@or.com</actor:email>
                </actor:Actor>
        </profile:contactInformation>

        <profile:serviceCategory>
                <role:BusinessCategory rdf:ID="NAICS-category">
                        <role:value>
                                E-tailers
                        </role:value>
                        <role:code>
                                454111
                        </role:code>
                </role:BusinessCategory >
        </profile:serviceCategory>

        <profile:hasRole rdf:resource="&roles;#Role"/>
        <profile:hasRole rdf:resource="&roles;#Client"/>

        <profile:hasNorm rdf:resource="&norms;#LocationSpoofing"/>
        <profile:hasNorm rdf:resource="&norms;#ExposePersonalDetails"/>
        <profile:hasNorm rdf:resource="&norms;#Login"/>
        <profile:hasNorm rdf:resource="&norms;#ProvideRoot"/>
        <profile:hasNorm rdf:resource="&norms;#UpdateOtherUser"/>
        <profile:hasNorm rdf:resource="&norms;#RequireOWLS"/>
        <profile:hasNorm rdf:resource="&norms;#LegalRequirements"/>
</profile:Profile>
</rdf:RDF>
```

A.4.2 globalcatalog.npel

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE uridef[
  <!ENTITY rdf     "http://www.w3.org/1999/02/22-rdf-syntax-ns">
  <!ENTITY rdfs    "http://www.w3.org/2000/01/rdf-schema">
  <!ENTITY owl     "http://www.w3.org/2002/07/owl">
  <!ENTITY xsd     "http://www.w3.org/2001/XMLSchema">
  <!ENTITY catalog "http://web.me.com/adouglas/ontologies/npel/1.3/catalog.owl">
  <!ENTITY profile "http://web.me.com/adouglas/npel/globalprofile.npel">
  <!ENTITY norms   "http://web.me.com/adouglas/npel/globalnorms.npel">
  <!ENTITY THIS    "http://web.me.com/adouglas/npel/globalcatalog.npel">
]>

<rdf:RDF        xmlns:rdf = "&rdf;#"
                        xmlns:rdfs = "&rdfs;#"
                        xmlns:xsd = "&xsd;#"
                        xmlns:owl = "&owl;#"
                        xmlns:catalog = "&catalog;#"
                        xmlns:profile = "&profile;#"
                        xmlns:norms = "&norms;#"
                        xmlns = "&THIS;#"
                        xml:base = "&THIS;">
        <owl:Ontology rdf:about="">
                <owl:versionInfo>
                         $Id: globalcatalog.npel V1.1 13/01/10 $
                </owl:versionInfo>

                <rdfs:comment></rdfs:comment>

                <owl:imports rdf:resource="&profile;#"/>
                <owl:imports rdf:resource="&norms;#"/>
                <owl:imports rdf:resource="&catalog;#"/>
        </owl:Ontology>


        <catalog:Catalog rdf:ID="GlobalCatalog">
                <!-- Reference to the Profile -->
                <catalog:presents rdf:resource="&profile;#Profile"/>
                <!-- Reference to the Norms -->
                <catalog:provides rdf:resource="&norms;#LocationSpoofing"/>
                <catalog:provides rdf:resource="&norms;#ExposePersonalDetails"/>
                <catalog:provides rdf:resource="&norms;#Login"/>
                <catalog:provides rdf:resource="&norms;#ProvideRoot"/>
                <catalog:provides rdf:resource="&norms;#UpdateOtherUser"/>
                <catalog:provides rdf:resource="&norms;#RequireOWLS"/>
                <catalog:provides rdf:resource="&norms;#LegalRequirements"/>
        </catalog:Catalog>
</rdf:RDF>
```

A.4.3 globalnorms.npel

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE uridef[
  <!ENTITY rdf        "http://www.w3.org/1999/02/22-rdf-syntax-ns">
  <!ENTITY rdfs       "http://www.w3.org/2000/01/rdf-schema">
```

```
  <!ENTITY owl        "http://www.w3.org/2002/07/owl">
  <!ENTITY xsd        "http://www.w3.org/2001/XMLSchema">
  <!ENTITY expr       "http://www.daml.org/services/owl-s/1.2/generic/Expression.owl">
  <!ENTITY swrl           "http://www.w3.org/2003/11/swrl">
  <!ENTITY fol            "http://www.w3.org/Submission/SWRL-FOL/">
  <!ENTITY concepts   "http://web.me.com/adouglas/npel/ontologies/concepts.owl">
  <!ENTITY norm       "http://web.me.com/adouglas/ontologies/npel/1.3/norm.owl">
  <!ENTITY activities "http://web.me.com/adouglas/npel/globalactivities.npel">
  <!ENTITY roles          "http://web.me.com/adouglas/npel/globalroles.npel">
  <!ENTITY exceptions "http://web.me.com/adouglas/npel/globalexceptions.npel">
  <!ENTITY THIS       "http://web.me.com/adouglas/npel/globalnorms.npel">
]>

<rdf:RDF        xmlns:rdf = "&rdf;#"
                        xmlns:rdfs = "&rdfs;#"
                        xmlns:xsd = "&xsd;#"
                        xmlns:owl = "&owl;#"
                        xmlns:expr = "&expr;#"
                        xmlns:swrl = "&swrl;#"
                        xmlns:fol = "&fol;#"
                        xmlns:concepts = "&concepts;#"
                        xmlns:norm = "&norm;#"
                        xmlns:roles = "&roles;#"
                        xmlns:exceptions = "&exceptions;#"
                        xmlns:activities = "&activities;#"
                        xmlns = "&THIS;#"
                        xml:base = "&THIS;">
        <owl:Ontology rdf:about="">
                <owl:versionInfo>
                         $Id: globalnorms.npel V1.1 13/01/10 $
                </owl:versionInfo>

                <rdfs:comment></rdfs:comment>

                <owl:imports rdf:resource="&exceptions;#"/>
                <owl:imports rdf:resource="&roles;#"/>
                <owl:imports rdf:resource="&activities;#"/>
                <owl:imports rdf:resource="&norm;#"/>
                <owl:imports rdf:resource="&concepts;#"/>
        </owl:Ontology>

        <!-- Norm forbiding location spoofing -->
        <norm:Forb rdf:ID="LocationSpoofing">
                <norm:normIdentifier ref:datatype="&xsd;#String">
                        LocationSpoofing
                </norm:normIdentifier>
                <norm:governsActivity rdf:resource="&activities;#Any1To1Activity" />
                <norm:hasRole rdf:resource="&roles;#Role" />
                <norm:hasInCondition>
                        <expr:SWRL-Condition>
                                <rdfs:label>
                                        once(#SigninActivity) and not(equals(@trigger.grounding.
    location, @current.grounding.location))
                                </rdfs:label>
                                <rdfs:comment>
                                        Once logged in if grounding.location does not equal
    trigger.grounding.location.
                                </rdfs:comment>
                                <expr:expressionLanguage rdf:resource="&expr;#SWRL" />
```

```
                        <expr:expressionBody rdf:parseType="Literal">
                                <fol:Assertion>
                                        <fol:And>
                                                <swrl:IndividualPropertyAtom>
                                                        <swrl:propertyPredicate
rdf:resource="&concepts;#Once" />
                                                        <swrl:argument1 rdf:resource="&
roles;#Role" />
                                                        <swrl:argument2 rdf:resource="&
activities;#SigninActivity" />
                                                </swrl:IndividualPropertyAtom>
                                                <swrl:builtinAtom  swrl:builtin="&swrlb;#
notEqual">
                                                <swrl:argument1 rdf:resource="&concepts;#
Trigger_Grounding_Location" />
                                                        <swrl:argument2 rdf:resource="&
concepts;#Current_Grounding_Location" />
                                                    </swrl:builtinAtom>
                                        </fol:And>
                                </fol:Assertion>
                        </expr:expressionBody>
                </expr:SWRL-Condition>
        </norm:hasInCondition>
        <norm:hasDCondition>
                <expr:SWRL-Condition>
                        <rdfs:label>
                                once(#SignOutActivity)
                        </rdfs:label>
                        <rdfs:comment>
                                Once logged out.
                        <expr:expressionLanguage rdf:resource="&expr;#SWRL" />
                        <expr:expressionBody rdf:parseType="Literal">
                                <swrl:IndividualPropertyAtom>
                                        <swrl:propertyPredicate rdf:resource="&concepts;#
Once" />
                                        <swrl:argument1 rdf:resource="&roles;#Role" />
                                        <swrl:argument2 rdf:resource="&activities;#
SignOutActivity" />
                                </swrl:IndividualPropertyAtom>
                        </expr:expressionBody>
                </expr:SWRL-Condition>
        </norm:hasDContition>
        <norm:triggersException rdf:resource="&exceptions;#LocationSpoofingException" />
   </norm:Forb>

   <!-- Norm forbiding the leaking of personal data to users not logged in -->
   <norm:Forb rdf:ID="ExposePersonalDetails">
        <norm:normIdentifier ref:datatype="&xsd;#String">
                ExposePersonalDetails
        </norm:normIdentifier>
        <norm:governsActivity rdf:resource="&activities;#ReturnPersonalDetailsActivity" />
        <norm:hasRole rdf:resource="&roles;#Client" />
        <norm:hasInCondition>
                <expr:SWRL-Condition>
                        <rdfs:label>
                                once(@start) and once(#SignOutActivity)
                        </rdfs:label>
                        <rdfs:comment>
                                If not logged in.
```

```
                        </rdfs:comment>
                        <expr:expressionLanguage rdf:resource="&expr;#SWRL" />
                        <expr:expressionBody rdf:parseType="Literal">
                                <fol:Assertion>
                                        <fol:And>
                                                <swrl:IndividualPropertyAtom>
                                                        <swrl:propertyPredicate
rdf:resource="&concepts;#Once" />
                                                        <swrl:argument1 rdf:resource="&
roles;#Client" />
                                                        <swrl:argument2 rdf:resource="&
concepts;#Start" />
                                                </swrl:IndividualPropertyAtom>
                                                <swrl:IndividualPropertyAtom>
                                                        <swrl:propertyPredicate
rdf:resource="&concepts;#Once" />
                                                        <swrl:argument1 rdf:resource="&
roles;#Client" />
                                                        <swrl:argument2 rdf:resource="&
activities;#SignOutActivity" />
                                                </swrl:IndividualPropertyAtom>
                                        </fol:And>
                                </fol:Assertion>
                        </expr:expressionBody>
                </expr:SWRL-Condition>
        </norm:hasInCondition>
        <norm:hasDCondition>
                <expr:SWRL-Condition>
                        <rdfs:label>
                                once(#SigninActivity)
                        </rdfs:label>
                        <rdfs:comment>
                                Once logged in.
                        <expr:expressionLanguage rdf:resource="&expr;#SWRL" />
                        <expr:expressionBody rdf:parseType="Literal">
                                <swrl:IndividualPropertyAtom>
                                        <swrl:propertyPredicate rdf:resource="&concepts;#
Once" />
                                        <swrl:argument1 rdf:resource="&roles;#Client" />
                                        <swrl:argument2 rdf:resource="&activities;#
SigninActivity" />
                                </swrl:IndividualPropertyAtom>
                        </expr:expressionBody>
                </expr:SWRL-Condition>
        </norm:hasDContition>
        <norm:triggersException rdf:resource="&exceptions;#ExposePersonalDetailsException"
 />
   </norm:Forb>



   <!-- Norm forbiding the root data -->
   <norm:Forb rdf:ID="ProvideRoot">
        <norm:normIdentifier ref:datatype="&xsd;#String">
                ProvideRoot
        </norm:normIdentifier>
        <norm:governsActivity rdf:resource="&activities;#ProvideRootTokenActivity" />
        <norm:hasRole rdf:resource="&roles;#Client" />
        <norm:hasInCondition>
```

```
                    <expr:SWRL-Condition>
                            <rdfs:label>
                                    once(@start)
                            </rdfs:label>
                            <rdfs:comment>
                                    At start.
                            </rdfs:comment>
                            <expr:expressionLanguage rdf:resource="&expr;#SWRL" />
                            <expr:expressionBody rdf:parseType="Literal">
                                    <swrl:IndividualPropertyAtom>
                                            <swrl:propertyPredicate rdf:resource="&concepts;#
Once" />
                                            <swrl:argument1 rdf:resource="&roles;#Client" />
                                            <swrl:argument2 rdf:resource="&concepts;#Start" />
                                    </swrl:IndividualPropertyAtom>
                            </expr:expressionBody>
                    </expr:SWRL-Condition>
            </norm:hasInCondition>
            <norm:hasDCondition>
                    <expr:SWRL-Condition>
                            <rdfs:label>
                                    once(@end)
                            </rdfs:label>
                            <rdfs:comment>
                                    At end.
                            <expr:expressionLanguage rdf:resource="&expr;#SWRL" />
                            <expr:expressionBody rdf:parseType="Literal">
                                    <swrl:IndividualPropertyAtom>
                                            <swrl:propertyPredicate rdf:resource="&concepts;#
Once" />
                                            <swrl:argument1 rdf:resource="&roles;#Client" />
                                            <swrl:argument2 rdf:resource="&concepts;#End" />
                                    </swrl:IndividualPropertyAtom>
                            </expr:expressionBody>
                    </expr:SWRL-Condition>
            </norm:hasDContition>
            <norm:triggersException rdf:resource="&exceptions;#ProvideRootException" />
    </norm:Forb>



    <!-- Norm permitting return of account data following a login -->
    <norm:Perm rdf:ID="Login">
            <norm:normIdentifier ref:datatype="&xsd;#String">
                    Login
            </norm:normIdentifier>
            <norm:governsActivity rdf:resource="&activities;#LoadUserProfileActivity" />
            <norm:hasRole rdf:resource="&roles;#Client" />
            <norm:hasInCondition>
                    <expr:SWRL-Condition>
                            <rdfs:label>
                                    once(#SigninActivity)
                            </rdfs:label>
                            <rdfs:comment>
                                    Once logged in.
                            </rdfs:comment>
                            <expr:expressionLanguage rdf:resource="&expr;#SWRL" />
                            <expr:expressionBody rdf:parseType="Literal">
                                    <swrl:IndividualPropertyAtom>
```

```
                                        <swrl:propertyPredicate rdf:resource="&concepts;#
Once" />
                                        <swrl:argument1 rdf:resource="&roles;#Client" />
                                        <swrl:argument2 rdf:resource="&activities;#
SigninActivity" />
                                </swrl:IndividualPropertyAtom>
                        </expr:expressionBody>
                </expr:SWRL-Condition>
        </norm:hasInCondition>
        <norm:hasDCondition>
                <expr:SWRL-Condition>
                        <rdfs:label>
                                once(#SignOutActivity)
                        </rdfs:label>
                        <rdfs:comment>
                                Once logged out.
                        <expr:expressionLanguage rdf:resource="&expr;#SWRL" />
                        <expr:expressionBody rdf:parseType="Literal">
                                <swrl:IndividualPropertyAtom>
                                        <swrl:propertyPredicate rdf:resource="&concepts;#
Once" />
                                        <swrl:argument1 rdf:resource="&roles;#Client" />
                                        <swrl:argument2 rdf:resource="&activities;#
SignOutActivity" />
                                </swrl:IndividualPropertyAtom>
                        </expr:expressionBody>
                </expr:SWRL-Condition>
        </norm:hasDContition>
        <norm:triggersException rdf:resource="&exceptions;#LoginException" />
   </norm:Perm>



   <!-- Norm requiring owls -->
   <norm:Forb rdf:ID="RequireOWLS">
        <norm:normIdentifier ref:datatype="&xsd;#String">
                RequireOWLS
        </norm:normIdentifier>
        <norm:governsActivity rdf:resource="&activities;#Any1To1Activity" />
        <norm:hasRole rdf:resource="&roles;#Role" />
        <norm:hasInCondition>
                <expr:SWRL-Condition>
                        <rdfs:label>
                                true
                        </rdfs:label>
                        <rdfs:comment>
                                Always active at load.
                        </rdfs:comment>
                        <expr:expressionLanguage rdf:resource="&expr;#SWRL" />
                        <expr:expressionBody rdf:parseType="Literal">
                                <!-- Trivially true -->
                        </expr:expressionBody>
                </expr:SWRL-Condition>
        </norm:hasInCondition>
        <norm:hasDCondition>
                <expr:SWRL-Condition>
                        <rdfs:label>
                                loaded_function(OWLSAPI)
                        </rdfs:label>
```

```
                        <rdfs:comment>
                                Once the OWL-S API is loaded.
                        <expr:expressionLanguage rdf:resource="&expr;#SWRL" />
                        <expr:expressionBody rdf:parseType="Literal">
                                <swrl:IndividualPropertyAtom>
                                        <swrl:propertyPredicate rdf:resource="&concepts;#
Loaded_Function" />
                                        <swrl:argument1 rdf:resource="&roles;#Role" />
                                        <swrl:argument2 rdf:resource="&concepts;#OWLSAPI"
/>
                                </swrl:IndividualPropertyAtom>
                        </expr:expressionBody>
                </expr:SWRL-Condition>
        </norm:hasDContition>
        <norm:triggersException rdf:resource="&exceptions;#RequireOWLSException" />
   </norm:Forb>

   <!-- Norm forbiding the changing of another users information -->
   <norm:Forb rdf:ID="UpdateOtherUser">
        <norm:normIdentifier ref:datatype="&xsd;#String">
                UpdateOtherUser
        </norm:normIdentifier>
        <norm:governsActivity rdf:resource="&activities;#UpdatePersonalDetails" />
        <norm:hasRole rdf:resource="&roles;#Client" />
        <norm:hasInCondition>
                <expr:SWRL-Condition>
                        <rdfs:label>
                                once(#GetUserInfo)
                        </rdfs:label>
                        <rdfs:comment>
                                Load when getting another users info
                        </rdfs:comment>
                        <expr:expressionLanguage rdf:resource="&expr;#SWRL" />
                        <expr:expressionBody rdf:parseType="Literal">
                                <swrl:IndividualPropertyAtom>
                                        <swrl:propertyPredicate rdf:resource="&concepts;#
Once" />
                                        <swrl:argument1 rdf:resource="&roles;#Client" />
                                        <swrl:argument2 rdf:resource="&activities;#
GetUserInfo" />
                                </swrl:IndividualPropertyAtom>
                        </expr:expressionBody>
                </expr:SWRL-Condition>
        </norm:hasInCondition>
        <norm:hasDCondition>
                <expr:SWRL-Condition>
                        <rdfs:label>
                                once(@end)
                        </rdfs:label>
                        <rdfs:comment>
                                Stays to the end
                        <expr:expressionLanguage rdf:resource="&expr;#SWRL" />
                        <expr:expressionBody rdf:parseType="Literal">
                                <swrl:IndividualPropertyAtom>
                                        <swrl:propertyPredicate rdf:resource="&concepts;#
Once" />
                                        <swrl:argument1 rdf:resource="&roles;#Client" />
                                        <swrl:argument2 rdf:resource="&concepts;#End" />
                                </swrl:IndividualPropertyAtom>
```

```
                        </expr:expressionBody>
                    </expr:SWRL-Condition>
            </norm:hasDContition>
            <norm:triggersException rdf:resource="&exceptions;#UpdatePersonalDetailsException"
     />
      </norm:Forb>

      <!-- Forbids payment before any legal requirements have been fulfilled -->
      <norm:Forb rdf:ID="LegalRequirements">
            <norm:normIdentifier ref:datatype="&xsd;#String">
                  LegalRequirements
            </norm:normIdentifier>
            <norm:governsActivity rdf:resource="&activities;#PaymentActivity" />
            <norm:hasRole rdf:resource="&roles;#Client" />
            <norm:hasInCondition>
                  <expr:SWRL-Condition>
                        <rdfs:label>
                              once(#FinalizeOfferActivity)
                        </rdfs:label>
                        <rdfs:comment>
                              Triggered when an offer has been finalized
                        </rdfs:comment>
                        <expr:expressionLanguage rdf:resource="&expr;#SWRL" />
                        <expr:expressionBody rdf:parseType="Literal">
                              <swrl:IndividualPropertyAtom>
                                    <swrl:propertyPredicate rdf:resource="&concepts;#
    Once" />
                                    <swrl:argument1 rdf:resource="&roles;#Role" />
                                    <swrl:argument2 rdf:resource="&activities;#
    FinalizeOfferActivity" />
                              </swrl:IndividualPropertyAtom>
                        </expr:expressionBody>
                  </expr:SWRL-Condition>
            </norm:hasInCondition>
            <norm:hasDCondition>
                  <expr:SWRL-Condition>
                        <rdfs:label>
                              once(#OrderRequirementsActivity)
                        </rdfs:label>
                        <rdfs:comment>
                              Released when any requirements have been met
                        <expr:expressionLanguage rdf:resource="&expr;#SWRL" />
                        <expr:expressionBody rdf:parseType="Literal">
                              <swrl:IndividualPropertyAtom>
                                    <swrl:propertyPredicate rdf:resource="&concepts;#
    Once" />
                                    <swrl:argument1 rdf:resource="&roles;#Role" />
                                    <swrl:argument2 rdf:resource="&activities;#
    OrderRequirementsActivity" />
                              </swrl:IndividualPropertyAtom>
                        </expr:expressionBody>
                  </expr:SWRL-Condition>
            </norm:hasDContition>
            <norm:triggersException rdf:resource="&exceptions;#RequireOWLSException" />
      </norm:Forb>
</rdf:RDF>
```

A.4.4 hardprofile.npel

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE uridef[
  <!ENTITY rdf     "http://www.w3.org/1999/02/22-rdf-syntax-ns">
  <!ENTITY rdfs    "http://www.w3.org/2000/01/rdf-schema">
  <!ENTITY owl     "http://www.w3.org/2002/07/owl">
  <!ENTITY xsd     "http://www.w3.org/2001/XMLSchema">
  <!ENTITY actor "http://www.daml.org/services/owl-s/1.2/ActorDefault.owl">
  <!ENTITY profile "http://web.me.com/adouglas/ontologies/npel/1.3/profile.owl">
  <!ENTITY THIS    "http://web.me.com/adouglas/npel/hardprofile.npel">
]>

<rdf:RDF        xmlns:rdf = "&rdf;#"
                        xmlns:rdfs = "&rdfs;#"
                        xmlns:xsd = "&xsd;#"
                        xmlns:owl = "&owl;#"
                        xmlns:actor = "&actor;#"
                        xmlns:profile = "&profile;#"
                        xmlns = "&THIS;#"
                        xml:base = "&THIS;">
        <owl:Ontology rdf:about="">
                <owl:versionInfo>
                          $Id: hardprofile.owl v1.1 13/01/10 $
                </owl:versionInfo>
                <rdfs:comment></rdfs:comment>

                <owl:imports rdf:resource="&profile;#"/>
        </owl:Ontology>

<profile:Profile rdf:ID="Profile">
        <catalog:presentedBy rdf:resource="&catalog;#ExampleCatalog"/>

        <profile:textDescription>
                        The catalog provides an example of how NPE-L might be deployed in a real
     world situation.
        </profile:textDescription>

        <profile:contactInformation>
                <actor:Actor rdf:ID="Contact">
                <actor:name>Example Owner</actor:name>
                <actor:title> Service Representative </actor:title>
                <actor:email>act@or.com</actor:email>
                </actor:Actor>
        </profile:contactInformation>

        <profile:serviceCategory>
                <role:BusinessCategory rdf:ID="NAICS-category">
                        <role:value>
                                E-tailers
                        </role:value>
                        <role:code>
                                454111
                        </role:code>
                </role:BusinessCategory >
        </profile:serviceCategory>

        <profile:hasRole rdf:resource="&roles;#Role"/>
        <profile:hasRole rdf:resource="&roles;#Searcher"/>
```

```
        <profile:hasRole rdf:resource="&roles;#Client"/>

        <catalog:hasNorm rdf:resource="&norms;#OverGeneralisedSearch"/>
        <catalog:hasNorm rdf:resource="&norms;#RequireSecurity"/>
        <catalog:hasNorm rdf:resource="&norms;#ExtraPayment"/>
        <catalog:hasNorm rdf:resource="&norms;#EmptyWorkflow"/>
        <catalog:hasNorm rdf:resource="&norms;#UnexpectedTermination"/>
        <catalog:hasNorm rdf:resource="&norms;#Fallback"/>
        <catalog:hasNorm rdf:resource="&norms;#OverGeneralisedAddItem"/>
        <catalog:hasNorm rdf:resource="&norms;#RequireShoppingSecurity"/>
        <catalog:hasNorm rdf:resource="&norms;#BookMedicalAppointment"/>
        <catalog:hasNorm rdf:resource="&norms;#RequireEncryption"/>
        <catalog:hasNorm rdf:resource="&norms;#ReturnDeeds"/>
        <catalog:hasNorm rdf:resource="&norms;#RequestFurtherInformation"/>
</profile:Profile>
</rdf:RDF>
```

A.4.5 hardcatalog.npel

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE uridef[
  <!ENTITY rdf     "http://www.w3.org/1999/02/22-rdf-syntax-ns">
  <!ENTITY rdfs    "http://www.w3.org/2000/01/rdf-schema">
  <!ENTITY owl     "http://www.w3.org/2002/07/owl">
  <!ENTITY xsd     "http://www.w3.org/2001/XMLSchema">
  <!ENTITY catalog "http://web.me.com/adouglas/ontologies/npel/1.3/catalog.owl">
  <!ENTITY profile "http://web.me.com/adouglas/npel/hardprofile.npel">
  <!ENTITY norms   "http://web.me.com/adouglas/npel/hardnorms.npel">
  <!ENTITY THIS    "http://web.me.com/adouglas/npel/hardcatalog.npel">
]>

<rdf:RDF        xmlns:rdf = "&rdf;#"
                        xmlns:rdfs = "&rdfs;#"
                        xmlns:xsd = "&xsd;#"
                        xmlns:owl = "&owl;#"
                        xmlns:catalog = "&catalog;#"
                        xmlns:profile = "&profile;#"
                        xmlns:norms = "&norms;#"
                        xmlns = "&THIS;#"
                        xml:base = "&THIS;">
        <owl:Ontology rdf:about="">
                <owl:versionInfo>
                         $Id: hardcatalog.npel V1.1 13/01/10 $
                </owl:versionInfo>

                <rdfs:comment></rdfs:comment>

                <owl:imports rdf:resource="&profile;#"/>
                <owl:imports rdf:resource="&norms;#"/>
                <owl:imports rdf:resource="&catalog;#"/>
        </owl:Ontology>


        <catalog:Catalog rdf:ID="HardCatalog">
                <!-- Reference to the Profile -->
                <catalog:presents rdf:resource="&profile;#ExampleProfile"/>
                <!-- Reference to the Norms -->
```

```
                <catalog:provides rdf:resource="&norms;#OverGeneralisedSearch"/>
                <catalog:provides rdf:resource="&norms;#RequireSecurity"/>
                <catalog:provides rdf:resource="&norms;#ExtraPayment"/>
                <catalog:provides rdf:resource="&norms;#EmptyWorkflow"/>
                <catalog:provides rdf:resource="&norms;#UnexpectedTermination"/>
                <catalog:provides rdf:resource="&norms;#Fallback"/>
                <catalog:provides rdf:resource="&norms;#OverGeneralisedAddItem"/>
                <catalog:provides rdf:resource="&norms;#RequireShoppingSecurity"/>
                <catalog:provides rdf:resource="&norms;#BookMedicalAppointment"/>
                <catalog:provides rdf:resource="&norms;#RequireEncryption"/>
                <catalog:provides rdf:resource="&norms;#ReturnDeeds"/>
                <catalog:provides rdf:resource="&norms;#RequestFurtherInformation"/>

        </catalog:Catalog>
</rdf:RDF>
```

A.4.6 hardnorms.npel

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE uridef[
  <!ENTITY rdf        "http://www.w3.org/1999/02/22-rdf-syntax-ns">
  <!ENTITY rdfs       "http://www.w3.org/2000/01/rdf-schema">
  <!ENTITY owl        "http://www.w3.org/2002/07/owl">
  <!ENTITY xsd        "http://www.w3.org/2001/XMLSchema">
  <!ENTITY expr       "http://www.daml.org/services/owl-s/1.2/generic/Expression.owl">
  <!ENTITY swrl           "http://www.w3.org/2003/11/swrl">
  <!ENTITY fol            "http://www.w3.org/Submission/SWRL-FOL/">
  <!ENTITY norm       "http://web.me.com/adouglas/ontologies/npel/1.3/norm.owl">
  <!ENTITY activities "http://web.me.com/adouglas/npel/hardactivities.npel">
  <!ENTITY concepts   "http://web.me.com/adouglas/npel/ontologies/concepts.owl">
  <!ENTITY roles          "http://web.me.com/adouglas/npel/hardroles.npel">
  <!ENTITY exceptions "http://web.me.com/adouglas/npel/hardexceptions.npel">
  <!ENTITY THIS       "http://web.me.com/adouglas/npel/hardnorms.npel">
]>

<rdf:RDF        xmlns:rdf = "&rdf;#"
                        xmlns:rdfs = "&rdfs;#"
                        xmlns:xsd = "&xsd;#"
                        xmlns:owl = "&owl;#"
                        xmlns:expr = "&expr;#"
                        xmlns:swrl = "&swrl;#"
                        xmlns:fol = "&fol;#"
                        xmlns:norm = "&norm;#"
                        xmlns:concepts = "&concepts;#"
                        xmlns:roles = "&roles;#"
                        xmlns:exceptions = "&exceptions;#"
                        xmlns:activities = "&activities;#"
                        xmlns = "&THIS;#"
                        xml:base = "&THIS;">
        <owl:Ontology rdf:about="">
                <owl:versionInfo>
                         $Id: hardnorms.npel V1.1 13/01/10 $
                </owl:versionInfo>

                <rdfs:comment></rdfs:comment>

                <owl:imports rdf:resource="&exceptions;#"/>
```

```
        <owl:imports rdf:resource="&roles;#"/>
        <owl:imports rdf:resource="&activities;#"/>
        <owl:imports rdf:resource="&norm;#"/>
        <owl:imports rdf:resource="&concepts;#"/>
    </owl:Ontology>



    <!-- Norm preventing generalisation of search terms during shopping activities -->
    <norm:Forb rdf:ID="OverGeneralisedSearch">
        <norm:normIdentifier ref:datatype="&xsd;#String">
            OverGeneralisedSearch
        </norm:normIdentifier>
        <norm:governsActivity rdf:resource="&activities;#GeneralisedSearchActivity" />
        <norm:hasRole rdf:resource="&roles;#Searcher" />
        <norm:hasInCondition>
            <expr:SWRL-Condition>
                <rdfs:label>
                    next(#PutInCartActivity)
                </rdfs:label>
                <rdfs:comment>
                    Test to see if the next step in the workflow is to place
an item in a cart.
                </rdfs:comment>
                <expr:expressionLanguage rdf:resource="&expr;#SWRL" />
                <expr:expressionBody rdf:parseType="Literal">
                    <swrl:IndividualPropertyAtom>
                        <swrl:propertyPredicate rdf:resource="&concepts;#
Next" />
                        <swrl:argument1 rdf:resource="&roles;#Role" />
                        <swrl:argument2 rdf:resource="&activities;#
PutInCartActivity" />
                    </swrl:IndividualPropertyAtom>
                </expr:expressionBody>
            </expr:SWRL-Condition>
        </norm:hasInCondition>
        <norm:hasDCondition>
            <expr:SWRL-Condition>
                <rdfs:label>
                    not(in(#GeneralisedSearchActivity))
                </rdfs:label>
                <rdfs:comment>
                    Norm deactivates if this is not currently a
GeneralisedSearchActivity
                </rdfs:comment>
                <expr:expressionLanguage rdf:resource="&expr;#SWRL" />
                <expr:expressionBody rdf:parseType="Literal">
                    <swrl:builtinAtom  swrl:builtin="&swrlb;#booleanNot">
                    <swrl:IndividualPropertyAtom>
                                <swrl:propertyPredicate rdf:resource="&
concepts;#In" />
                                <swrl:argument1 rdf:resource="&roles;#
Searcher" />
                                <swrl:argument2 rdf:resource="&activities
;#GeneralisedSearchActivity" />
                            </swrl:IndividualPropertyAtom>
                        </swrl:builtinAtom>
                </expr:expressionBody>
            </expr:SWRL-Condition>
```

```
            </norm:hasDContition>
            <norm:triggersException rdf:resource="&exceptions;#OverGeneralisedSearchException"
 />
      </norm:Forb>



      <!-- Norm forbiding any secure actions if not logged in -->
      <norm:Forb rdf:ID="RequireSecurity">
            <norm:normIdentifier ref:datatype="&xsd;#String">
                  RequireSecurity
            </norm:normIdentifier>
            <norm:governsActivity rdf:resource="&activities;#SecureActivity" />
            <norm:hasRole rdf:resource="&roles;#Client" />
            <norm:hasInCondition>
                  <expr:SWRL-Condition>
                        <rdfs:label>
                              once(@start) or once(#SignOutActivity)
                        </rdfs:label>
                        <rdfs:comment>
                              True if in a workflow and signed out
                        </rdfs:comment>
                        <expr:expressionLanguage rdf:resource="&expr;#SWRL" />
                        <expr:expressionBody rdf:parseType="Literal">
                              <fol:Assertion>
                                    <fol:Or>
                                          <swrl:IndividualPropertyAtom>
                                                <swrl:propertyPredicate
rdf:resource="&concepts;#Once" />
                                                <swrl:argument1 rdf:resource="&
roles;#Client" />
                                                <swrl:argument2 rdf:resource="&
concepts;#Start" />
                                          </swrl:IndividualPropertyAtom>
                                          <swrl:IndividualPropertyAtom>
                                                <swrl:propertyPredicate
rdf:resource="&concepts;#Once" />
                                                <swrl:argument1 rdf:resource="&
roles;#Client" />
                                                <swrl:argument2 rdf:resource="&
activities;#SignOutActivity" />
                                          </swrl:IndividualPropertyAtom>
                                    </fol:Or>
                              <fol:Assertion>
                        </expr:expressionBody>
                  </expr:SWRL-Condition>
            </norm:hasInCondition>
            <norm:hasDCondition>
                  <expr:SWRL-Condition>
                        <rdfs:label>
                              once(#SigninActivity)
                        </rdfs:label>
                        <rdfs:comment>
                              Test to see if at the time this norm is active that it
will be deactivated by the following condition. This is equivelent to the expression: once(
user, #PutInCart).
                        </rdfs:comment>
                        <expr:expressionLanguage rdf:resource="&expr;#SWRL" />
                        <expr:expressionBody rdf:parseType="Literal">
```

```
                                <swrl:IndividualPropertyAtom>
                                        <swrl:propertyPredicate rdf:resource="&concepts;#
Once" />
                                        <swrl:argument1 rdf:resource="&roles;#Client" />
                                        <swrl:argument2 rdf:resource="&activities;#
SigninActivity" />
                                </swrl:IndividualPropertyAtom>
                        </expr:expressionBody>
                </expr:SWRL-Condition>
        </norm:hasDContition>
        <norm:triggersException rdf:resource="&exceptions;#RequireSecurityException" />
    </norm:Forb>



    <!-- Norm forbiding unexpected payment steps -->
    <norm:Forb rdf:ID="ExtraPayment">
        <norm:normIdentifier ref:datatype="&xsd;#String">
                ExtraPayment
        </norm:normIdentifier>
        <norm:governsActivity rdf:resource="&activities;#PaymentActivity" />
        <norm:hasRole rdf:resource="&roles;#Client" />
        <norm:hasInCondition>
                <expr:SWRL-Condition>
                        <rdfs:label>
                                once(#PaymentActivity)
                        </rdfs:label>
                        <rdfs:comment>
                                Active if a payment step has already been passed
                        </rdfs:comment>
                        <expr:expressionLanguage rdf:resource="&expr;#SWRL" />
                        <expr:expressionBody rdf:parseType="Literal">
                                <swrl:IndividualPropertyAtom>
                                        <swrl:propertyPredicate rdf:resource="&concepts;#
Once" />
                                        <swrl:argument1 rdf:resource="&roles;#Client" />
                                        <swrl:argument2 rdf:resource="&activities;#
PaymentActivity" />
                                </swrl:IndividualPropertyAtom>
                        </expr:expressionBody>
                </expr:SWRL-Condition>
        </norm:hasInCondition>
        <norm:hasDCondition>
                <expr:SWRL-Condition>
                        <rdfs:label>
                                false
                        </rdfs:label>
                        <rdfs:comment>
                                Once active always active.
                        </rdfs:comment>
                        <expr:expressionLanguage rdf:resource="&expr;#SWRL" />
                        <expr:expressionBody rdf:parseType="Literal">
                                FALSE
                        </expr:expressionBody>
                </expr:SWRL-Condition>
        </norm:hasDContition>
        <norm:triggersException rdf:resource="&exceptions;#ExtraPaymentException" />
    </norm:Forb>
```

```
<!-- Norm obligating that an activity has to be present -->
<norm:Obli rdf:ID="EmptyWorkflow">
        <norm:normIdentifier ref:datatype="&xsd;#String">
                EmptyWorkflow
        </norm:normIdentifier>
        <norm:governsActivity rdf:resource="&activities;#Any1To1Activity" />
        <norm:hasRole rdf:resource="&roles;#Role" />
        <norm:hasInCondition>
                <expr:SWRL-Condition>
                        <rdfs:label>
                                once(@start)
                        </rdfs:label>
                        <rdfs:comment>
                                Activates at the start of the workflow.
                        </rdfs:comment>
                        <expr:expressionLanguage rdf:resource="&expr;#SWRL" />
                        <expr:expressionBody rdf:parseType="Literal">
                                <swrl:IndividualPropertyAtom>
                                        <swrl:propertyPredicate rdf:resource="&concepts;#
Once" />
                                        <swrl:argument1 rdf:resource="&roles;#Client" />
                                        <swrl:argument2 rdf:resource="&concepts;#Start" />
                                </swrl:IndividualPropertyAtom>
                        </expr:expressionBody>
                </expr:SWRL-Condition>
        </norm:hasInCondition>
        <norm:hasDCondition>
                <expr:SWRL-Condition>
                        <rdfs:label>
                                once(@end)
                        </rdfs:label>
                        <rdfs:comment>
                                Deactivates at the end of the workflow.
                        </rdfs:comment>
                        <expr:expressionLanguage rdf:resource="&expr;#SWRL" />
                        <expr:expressionBody rdf:parseType="Literal">
                                <swrl:IndividualPropertyAtom>
                                        <swrl:propertyPredicate rdf:resource="&concepts;#
Once" />
                                        <swrl:argument1 rdf:resource="&roles;#Client" />
                                        <swrl:argument2 rdf:resource="&concepts;#End" />
                                </swrl:IndividualPropertyAtom>
                        </expr:expressionBody>
                </expr:SWRL-Condition>
        </norm:hasDContition>
        <norm:triggersException rdf:resource="&exceptions;#EmptyWorkflowException" />
</norm:Obli>



<!-- Norm obligating a workflow must terminate normally -->
<norm:Obli rdf:ID="UnexpectedTermination">
        <norm:normIdentifier ref:datatype="&xsd;#String">
                UnexpectedTermination
        </norm:normIdentifier>
        <norm:governsActivity rdf:resource="&activities;#FinalActivity" />
        <norm:hasRole rdf:resource="&roles;#Role" />
```

```
            <norm:hasInCondition>
                    <expr:SWRL-Condition>
                            <rdfs:label>
                                    once(@start)
                            </rdfs:label>
                            <rdfs:comment>
                                    Activates at the start of the workflow
                            </rdfs:comment>
                            <expr:expressionLanguage rdf:resource="&expr;#SWRL" />
                            <expr:expressionBody rdf:parseType="Literal">
                                    <swrl:IndividualPropertyAtom>
                                            <swrl:propertyPredicate rdf:resource="&concepts;#
Once" />
                                            <swrl:argument1 rdf:resource="&roles;#Client" />
                                            <swrl:argument2 rdf:resource="&concepts;#Start" />
                                    </swrl:IndividualPropertyAtom>
                            </expr:expressionBody>
                    </expr:SWRL-Condition>
            </norm:hasInCondition>
            <norm:hasDCondition>
                    <expr:SWRL-Condition>
                            <rdfs:label>
                                    once(@end)
                            </rdfs:label>
                            <rdfs:comment>
                                    Deactivates at the end of the workflow.
                            </rdfs:comment>
                            <expr:expressionLanguage rdf:resource="&expr;#SWRL" />
                            <expr:expressionBody rdf:parseType="Literal">
                                    <swrl:IndividualPropertyAtom>
                                            <swrl:propertyPredicate rdf:resource="&concepts;#
Once" />
                                            <swrl:argument1 rdf:resource="&roles;#Client" />
                                            <swrl:argument2 rdf:resource="&concepts;#End" />
                                    </swrl:IndividualPropertyAtom>
                            </expr:expressionBody>
                    </expr:SWRL-Condition>
            </norm:hasDContition>
            <norm:triggersException rdf:resource="&exceptions;#UnexpectedTerminationException"
 />
    </norm:Obli>



    <!-- Norm obligating a fallback for transactional workflows -->
    <norm:Obli rdf:ID="Fallback">
            <norm:normIdentifier ref:datatype="&xsd;#String">
                    Fallback
            </norm:normIdentifier>
            <norm:governsActivity rdf:resource="&activities;#RecoverActivity" />
            <norm:hasRole rdf:resource="&roles;#Client" />
            <norm:hasInCondition>
                    <expr:SWRL-Condition>
                            <rdfs:label>
                                    in(#TransactionActivity)
                            </rdfs:label>
                            <rdfs:comment>
```

```
                                Test to see if at the time this norm is applied the agent
believes it will be in the #Buy_Sequence. This is equivelent to the expression: in_region(
user, #Buy_Sequence).
                        </rdfs:comment>
                        <expr:expressionLanguage rdf:resource="&expr;#SWRL" />
                        <expr:expressionBody rdf:parseType="Literal">
                                <swrl:IndividualPropertyAtom>
                                        <swrl:propertyPredicate rdf:resource="&concepts;#
In" />
                                        <swrl:argument1 rdf:resource="&roles;#Client" />
                                        <swrl:argument2 rdf:resource="&concepts;#
TransactionActivity" />
                                </swrl:IndividualPropertyAtom>
                        </expr:expressionBody>
                </expr:SWRL-Condition>
        </norm:hasInCondition>
        <norm:hasDCondition>
                <expr:SWRL-Condition>
                        <rdfs:label>
                                once(@end)
                        </rdfs:label>
                        <rdfs:comment>
                                Deactivates at the end of the workflow.
                        </rdfs:comment>
                        <expr:expressionLanguage rdf:resource="&expr;#SWRL" />
                        <expr:expressionBody rdf:parseType="Literal">
                                <swrl:IndividualPropertyAtom>
                                        <swrl:propertyPredicate rdf:resource="&concepts;#
Once" />
                                        <swrl:argument1 rdf:resource="&roles;#Client" />
                                        <swrl:argument2 rdf:resource="&concepts;#End" />
                                </swrl:IndividualPropertyAtom>
                        </expr:expressionBody>
                </expr:SWRL-Condition>
        </norm:hasDContition>
        <norm:triggersException rdf:resource="&exceptions;#FallbackException" />
   </norm:Obli>




   <!-- Norm preventing generalisation of items -->
   <norm:Forb rdf:ID="OverGeneralisedAddItem">
        <norm:normIdentifier ref:datatype="&xsd;#String">
                OverGeneralisedAddItem
        </norm:normIdentifier>
        <norm:governsActivity rdf:resource="&activities;#GeneralisedAddItemActivity" />
        <norm:hasRole rdf:resource="&roles;#Searcher" />
        <norm:hasInCondition>
                <expr:SWRL-Condition>
                        <rdfs:label>
                                once(@start)
                        </rdfs:label>
                        <rdfs:comment>
                                Activated at the start of the workflow.
                        </rdfs:comment>
                        <expr:expressionLanguage rdf:resource="&expr;#SWRL" />
                        <expr:expressionBody rdf:parseType="Literal">
                                <swrl:IndividualPropertyAtom>
```

```
                                        <swrl:propertyPredicate rdf:resource="&concepts;#
Once" />
                                        <swrl:argument1 rdf:resource="&roles;#Client" />
                                        <swrl:argument2 rdf:resource="&concepts;#Start" />
                                </swrl:IndividualPropertyAtom>
                        </expr:expressionBody>
                </expr:SWRL-Condition>
        </norm:hasInCondition>
        <norm:hasDCondition>
                <expr:SWRL-Condition>
                        <rdfs:label>
                                once(@end)
                        </rdfs:label>
                        <rdfs:comment>
                                Deactivates at the end of the workflow.
                        </rdfs:comment>
                        <expr:expressionLanguage rdf:resource="&expr;#SWRL" />
                        <expr:expressionBody rdf:parseType="Literal">
                                <swrl:IndividualPropertyAtom>
                                        <swrl:propertyPredicate rdf:resource="&concepts;#
Once" />
                                        <swrl:argument1 rdf:resource="&roles;#Client" />
                                        <swrl:argument2 rdf:resource="&concepts;#End" />
                                </swrl:IndividualPropertyAtom>
                        </expr:expressionBody>
                </expr:SWRL-Condition>
        </norm:hasDContition>
        <norm:triggersException rdf:resource="&exceptions;#OverGeneralisedAddItemException
" />
   </norm:Forb>



   <!-- Norm forbiding any secure shopping actions if not logged in -->
   <norm:Forb rdf:ID="RequireShoppingSecurity">
        <norm:normIdentifier ref:datatype="&xsd;#String">
                RequireSecurity
        </norm:normIdentifier>
        <norm:governsActivity rdf:resource="&activities;#SecureShoppingActivity" />
        <norm:hasRole rdf:resource="&roles;#Client" />
        <norm:hasInCondition>
                <expr:SWRL-Condition>
                        <rdfs:label>
                                once(@start) or once(#SignOutActivity)
                        </rdfs:label>
                        <rdfs:comment>
                                Activates at start of workflow or if signed out.
                        </rdfs:comment>
                        <expr:expressionLanguage rdf:resource="&expr;#SWRL" />
                        <expr:expressionBody rdf:parseType="Literal">
                                <fol:Assertion>
                                        <fol:Or>
                                                <swrl:IndividualPropertyAtom>
                                                        <swrl:propertyPredicate
rdf:resource="&concepts;#Once" />
                                                        <swrl:argument1 rdf:resource="&
roles;#Client" />
                                                        <swrl:argument2 rdf:resource="&
concepts;#Start" />
```

```
                                                    </swrl:IndividualPropertyAtom>
                                                    <swrl:IndividualPropertyAtom>
                                                            <swrl:propertyPredicate
rdf:resource="&concepts;#Once" />
                                                            <swrl:argument1 rdf:resource="&
roles;#Client" />
                                                            <swrl:argument2 rdf:resource="&
activities;#SignOutActivity" />
                                                    </swrl:IndividualPropertyAtom>
                                            </fol:Or>
                                    </fol:Assertion>
                            </expr:expressionBody>
                    </expr:SWRL-Condition>
            </norm:hasInCondition>
            <norm:hasDCondition>
                    <expr:SWRL-Condition>
                            <rdfs:label>
                                    once(#SigninActivity)
                            </rdfs:label>
                            <rdfs:comment>
                                    Deactivated when signed in
                            </rdfs:comment>
                            <expr:expressionLanguage rdf:resource="&expr;#SWRL" />
                            <expr:expressionBody rdf:parseType="Literal">
                                    <swrl:IndividualPropertyAtom>
                                            <swrl:propertyPredicate rdf:resource="&concepts;#
Once" />
                                            <swrl:argument1 rdf:resource="&roles;#Client" />
                                            <swrl:argument2 rdf:resource="&activities;#
SigninActivity" />
                                    </swrl:IndividualPropertyAtom>
                            </expr:expressionBody>
                    </expr:SWRL-Condition>
            </norm:hasDContition>
            <norm:triggersException rdf:resource="&exceptions;#
RequireShoppingSecurityException" />
    </norm:Forb>



    <!-- Norm requiring encryption -->
    <norm:Forb rdf:ID="RequireEncryption">
            <norm:normIdentifier ref:datatype="&xsd;#String">
                    RequireEncryption
            </norm:normIdentifier>
            <norm:governsActivity rdf:resource="&activities;#EncryptedActivity" />
            <norm:hasRole rdf:resource="&roles;#Client" />
            <norm:hasInCondition>
                    <expr:SWRL-Condition>
                            <rdfs:label>
                                    once(@start) or equals(@current.grounding.protocol,http)
                            </rdfs:label>
                            <rdfs:comment>
                                    Activates at start of workflow.
                            </rdfs:comment>
                            <expr:expressionLanguage rdf:resource="&expr;#SWRL" />
                            <expr:expressionBody rdf:parseType="Literal">
                                    <fol:Assertion>
                                            <fol:Or>
```

```
                                                <swrl:IndividualPropertyAtom>
                                                        <swrl:propertyPredicate
rdf:resource="&concepts;#Once" />
                                                        <swrl:argument1 rdf:resource="&
roles;#Client" />
                                                        <swrl:argument2 rdf:resource="&
concepts;#Start" />
                                                </swrl:IndividualPropertyAtom>
                                                <swrl:builtinAtom  swrl:builtin="&swrlb;#
equal">
                                                <swrl:argument1 rdf:resource="&concepts;#
Current_Grounding_Protocol" />
                                                        <swrl:argument2 rdf:resource="&
concepts;#HTTP" />
                                            </swrl:builtinAtom>
                                        </fol:Or>
                                </fol:Assertion>
                        </expr:expressionBody>
                </expr:SWRL-Condition>
        </norm:hasInCondition>
        <norm:hasDCondition>
                <expr:SWRL-Condition>
                        <rdfs:label>
                                equals(@current.grounding.protocol,https)
                        </rdfs:label>
                        <rdfs:comment>
                                Deactiviated if https protocol is in use.
                        </rdfs:comment>
                        <expr:expressionLanguage rdf:resource="&expr;#SWRL" />
                        <expr:expressionBody rdf:parseType="Literal">
                                <swrl:builtinAtom  swrl:builtin="&swrlb;#equal">
                                <swrl:argument1 rdf:resource="&concepts;#
Current_Grounding_Protocol" />
                                        <swrl:argument2 rdf:resource="&concepts;#HTTPS" />
                            </swrl:builtinAtom>
                        </expr:expressionBody>
                </expr:SWRL-Condition>
        </norm:hasDContition>
        <norm:triggersException rdf:resource="&exceptions;#RequireEncryptionException" />
   </norm:Forb>



   <!-- Norm permitting the booking of a medical appointment -->
   <norm:Perm rdf:ID="BookMedicalAppointment">
        <norm:normIdentifier ref:datatype="&xsd;#String">
                BookMedicalAppointment
        </norm:normIdentifier>
        <norm:governsActivity rdf:resource="&activities;#BookMedicalAppointmentActivity" /
>
        <norm:hasRole rdf:resource="&roles;#Client" />
        <norm:hasInCondition>
                <expr:SWRL-Condition>
                        <rdfs:label>
                                once(#FindMedicalLocationActivity)
                        </rdfs:label>
                        <rdfs:comment>
                                Activates once #FindMedicalLocationActivity fires.
                        </rdfs:comment>
```

```
                        <expr:expressionLanguage rdf:resource="&expr;#SWRL" />
                        <expr:expressionBody rdf:parseType="Literal">
                                <swrl:IndividualPropertyAtom>
                                        <swrl:propertyPredicate rdf:resource="&concepts;#
Once" />
                                        <swrl:argument1 rdf:resource="&roles;#Client" />
                                        <swrl:argument2 rdf:resource="&activities;#
FindMedicalLocationActivity" />
                                </swrl:IndividualPropertyAtom>
                        </expr:expressionBody>
                </expr:SWRL-Condition>
        </norm:hasInCondition>
        <norm:hasDCondition>
                <expr:SWRL-Condition>
                        <rdfs:label>
                                once(#BookMedicalAppointmentActivity)
                        </rdfs:label>
                        <rdfs:comment>
                                Deactivates once BookMedicalAppointmentActivity fires.
                        </rdfs:comment>
                        <expr:expressionLanguage rdf:resource="&expr;#SWRL" />
                        <expr:expressionBody rdf:parseType="Literal">
                                <swrl:IndividualPropertyAtom>
                                        <swrl:propertyPredicate rdf:resource="&concepts;#
Once" />
                                        <swrl:argument1 rdf:resource="&roles;#Client" />
                                        <swrl:argument2 rdf:resource="&activities;#
BookMedicalAppointmentActivity" />
                                </swrl:IndividualPropertyAtom>
                        </expr:expressionBody>
                </expr:SWRL-Condition>
        </norm:hasDContition>
        <norm:triggersException rdf:resource="&exceptions;#BookMedicalAppointmentException
" />
   </norm:Perm>



   <!-- Norm requiring that workflows need to be logged in before undertaking secure
activities. -->
   <norm:Forb rdf:ID="RequireLoginForSecure">
        <norm:normIdentifier ref:datatype="&xsd;#String">
                RequireLoginForSecure
        </norm:normIdentifier>
        <norm:governsActivity rdf:resource="&activities;#SecureActivity" />
        <norm:hasRole rdf:resource="&roles;#Client" />
        <norm:hasInCondition>
                <expr:SWRL-Condition>
                        <rdfs:label>
                                inactive(#Login)
                        </rdfs:label>
                        <rdfs:comment>
                                Activates once if the norm Login is inactive.
                        </rdfs:comment>
                        <expr:expressionLanguage rdf:resource="&expr;#SWRL" />
                        <expr:expressionBody rdf:parseType="Literal">
                                <swrl:IndividualPropertyAtom>
                                        <swrl:propertyPredicate rdf:resource="&concepts;#
Inactive" />
```

```
                                        <swrl:argument1 rdf:resource="&roles;#Role" />
                                        <swrl:argument2 rdf:resource="&activities;#Login"
/>
                                </swrl:IndividualPropertyAtom>
                        </expr:expressionBody>
                </expr:SWRL-Condition>
        </norm:hasInCondition>
        <norm:hasDCondition>
                <expr:SWRL-Condition>
                        <rdfs:label>
                                active(#Login)
                        </rdfs:label>
                        <rdfs:comment>
                                Deactivates once the norm login is activated.
                        </rdfs:comment>
                        <expr:expressionLanguage rdf:resource="&expr;#SWRL" />
                        <expr:expressionBody rdf:parseType="Literal">
                                <swrl:IndividualPropertyAtom>
                                        <swrl:propertyPredicate rdf:resource="&concepts;#
Active" />
                                        <swrl:argument1 rdf:resource="&roles;#Role" />
                                        <swrl:argument2 rdf:resource="&activities;#Login"
/>
                                </swrl:IndividualPropertyAtom>
                        </expr:expressionBody>
                </expr:SWRL-Condition>
        </norm:hasDContition>
        <norm:triggersException rdf:resource="&exceptions;#RequireLoginForSecureException"
 />
   </norm:Forb>



   <!-- Norm requiring that if an extra payment cycle is found a reciept should be provided.
-->
   <norm:Forb rdf:ID="RequireRecieptAfterExtraPayment">
        <norm:normIdentifier ref:datatype="&xsd;#String">
                RequireRecieptAfterExtraPayment
        </norm:normIdentifier>
        <norm:governsActivity rdf:resource="&activities;#RecieptActivity" />
        <norm:hasRole rdf:resource="&roles;#Client" />
        <norm:hasInCondition>
                <expr:SWRL-Condition>
                        <rdfs:label>
                                triggered(#ExtraPaymentException)
                        </rdfs:label>
                        <rdfs:comment>
                                Activates once if the norm Login is inactive.
                        </rdfs:comment>
                        <expr:expressionLanguage rdf:resource="&expr;#SWRL" />
                        <expr:expressionBody rdf:parseType="Literal">
                                <swrl:IndividualPropertyAtom>
                                        <swrl:propertyPredicate rdf:resource="&concepts;#
Triggered" />
                                        <swrl:argument1 rdf:resource="&roles;#Role" />
                                        <swrl:argument2 rdf:resource="&exceptions;#
ExtraPaymentException" />
                                </swrl:IndividualPropertyAtom>
                        </expr:expressionBody>
```

```
                </expr:SWRL-Condition>
            </norm:hasInCondition>
            <norm:hasDCondition>
                <expr:SWRL-Condition>
                    <rdfs:label>
                        once(@end)
                    </rdfs:label>
                    <rdfs:comment>
                        Deactivates at end of workflow.
                    </rdfs:comment>
                    <expr:expressionLanguage rdf:resource="&expr;#SWRL" />
                    <expr:expressionBody rdf:parseType="Literal">
                        <swrl:IndividualPropertyAtom>
                            <swrl:propertyPredicate rdf:resource="&concepts;#
Once" />
                            <swrl:argument1 rdf:resource="&roles;#Role" />
                            <swrl:argument2 rdf:resource="&concepts;#End" />
                        </swrl:IndividualPropertyAtom>
                    </expr:expressionBody>
                </expr:SWRL-Condition>
            </norm:hasDContition>
            <norm:triggersException rdf:resource="&exceptions;#
RequireRecieptAfterExtraPaymentException" />
    </norm:Forb>

    <!-- Norm requiring that deeds are returned after finalizing a sale. -->
    <norm:Obli rdf:ID="ReturnDeeds">
            <norm:normIdentifier ref:datatype="&xsd;#String">
                ReturnDeeds
            </norm:normIdentifier>
            <norm:governsActivity rdf:resource="&activities;#TransferDeedsActivity" />
            <norm:hasRole rdf:resource="&roles;#Role" />
            <norm:hasInCondition>
                <expr:SWRL-Condition>
                    <rdfs:label>
                        once(#OfferSequence) and once(#FinaliseOfferActivity) and
once(#PaymentActivity)
                    </rdfs:label>
                    <rdfs:comment>
                        Activates once a complex sale has been finalized.
                    </rdfs:comment>
                    <expr:expressionLanguage rdf:resource="&expr;#SWRL" />
                    <expr:expressionBody rdf:parseType="Literal">
                        <fol:Assertion>
                            <fol:And>
                                <swrl:IndividualPropertyAtom>
                                    <swrl:propertyPredicate
rdf:resource="&concepts;#Once" />
                                    <swrl:argument1 rdf:resource="&
roles;#Role" />
                                    <swrl:argument2 rdf:resource="&
activities;#OfferSequence" />
                                </swrl:IndividualPropertyAtom>
                                <swrl:IndividualPropertyAtom>
                                    <swrl:propertyPredicate
rdf:resource="&concepts;#Once" />
                                    <swrl:argument1 rdf:resource="&
roles;#Role" />
```

```
                                                        <swrl:argument2 rdf:resource="&
activities;#FinaliseOfferActivity" />
                                                </swrl:IndividualPropertyAtom>
                                                <swrl:IndividualPropertyAtom>
                                                        <swrl:propertyPredicate
rdf:resource="&concepts;#Once" />
                                                        <swrl:argument1 rdf:resource="&
roles;#Role" />
                                                        <swrl:argument2 rdf:resource="&
activities;#PaymentActivity" />
                                                </swrl:IndividualPropertyAtom>
                                        </fol:And>
                                </fol:Assertion>
                        </expr:expressionBody>
                </expr:SWRL-Condition>
        </norm:hasInCondition>
        <norm:hasDCondition>
                <expr:SWRL-Condition>
                        <rdfs:label>
                                once(@end)
                        </rdfs:label>
                        <rdfs:comment>
                                Deactivates at end of workflow.
                        </rdfs:comment>
                        <expr:expressionLanguage rdf:resource="&expr;#SWRL" />
                        <expr:expressionBody rdf:parseType="Literal">
                                <swrl:IndividualPropertyAtom>
                                        <swrl:propertyPredicate rdf:resource="&concepts;#
Once" />
                                        <swrl:argument1 rdf:resource="&roles;#Role" />
                                        <swrl:argument2 rdf:resource="&concepts;#End" />
                                </swrl:IndividualPropertyAtom>
                        </expr:expressionBody>
                </expr:SWRL-Condition>
        </norm:hasDContition>
        <norm:triggersException rdf:resource="&exceptions;#TransferDeedsException" />
   </norm:Obli>

   <!-- Norm forbidding requesting further informtion before actor is first provided with
initial information. -->
   <norm:Forb rdf:ID="RequestFurtherInformation">
        <norm:normIdentifier ref:datatype="&xsd;#String">
                RequestFurtherInformation
        </norm:normIdentifier>
        <norm:governsActivity rdf:resource="&activities;#RequestFutherInfoActivity" />
        <norm:hasRole rdf:resource="&roles;#Client" />
        <norm:hasInCondition>
                <expr:SWRL-Condition>
                        <rdfs:label>
                                once(@start)
                        </rdfs:label>
                        <rdfs:comment>
                                Activates at the start of the workflow.
                        </rdfs:comment>
                        <expr:expressionLanguage rdf:resource="&expr;#SWRL" />
                        <expr:expressionBody rdf:parseType="Literal">
                                <swrl:IndividualPropertyAtom>
                                        <swrl:propertyPredicate rdf:resource="&concepts;#
Once" />
```

```
                                        <swrl:argument1 rdf:resource="&roles;#Client" />
                                        <swrl:argument2 rdf:resource="&concepts;#Start" />
                                    </swrl:IndividualPropertyAtom>
                                </expr:expressionBody>
                        </expr:SWRL-Condition>
                </norm:hasInCondition>
                <norm:hasDCondition>
                        <expr:SWRL-Condition>
                                <rdfs:label>
                                        once(#RequestInformationActivity)
                                </rdfs:label>
                                <rdfs:comment>
                                        Deactivates once initial information has been collected.
                                </rdfs:comment>
                                <expr:expressionLanguage rdf:resource="&expr;#SWRL" />
                                <expr:expressionBody rdf:parseType="Literal">
                                        <swrl:IndividualPropertyAtom>
                                                <swrl:propertyPredicate rdf:resource="&concepts;#
    Once" />
                                                <swrl:argument1 rdf:resource="&roles;#Client" />
                                                <swrl:argument2 rdf:resource="&activities;#
    RequestInformationActivity" />
                                        </swrl:IndividualPropertyAtom>
                                </expr:expressionBody>
                        </expr:SWRL-Condition>
                </norm:hasDContition>
                <norm:triggersException rdf:resource="&exceptions;#FurtherInformationException" />
        </norm:Forb>
</rdf:RDF>
```

A.4.7 softprofile.npel

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE uridef[
  <!ENTITY rdf     "http://www.w3.org/1999/02/22-rdf-syntax-ns">
  <!ENTITY rdfs    "http://www.w3.org/2000/01/rdf-schema">
  <!ENTITY owl     "http://www.w3.org/2002/07/owl">
  <!ENTITY xsd     "http://www.w3.org/2001/XMLSchema">
  <!ENTITY actor "http://www.daml.org/services/owl-s/1.2/ActorDefault.owl">
  <!ENTITY profile "http://web.me.com/adouglas/ontologies/npel/1.3/profile.owl">
  <!ENTITY THIS    "http://web.me.com/adouglas/npel/softprofile.npel">
]>

<rdf:RDF        xmlns:rdf = "&rdf;#"
                        xmlns:rdfs = "&rdfs;#"
                        xmlns:xsd = "&xsd;#"
                        xmlns:owl = "&owl;#"
                        xmlns:actor = "&actor;#"
                        xmlns:profile = "&profile;#"
                        xmlns = "&THIS;#"
                        xml:base = "&THIS;">
        <owl:Ontology rdf:about="">
                <owl:versionInfo>
                          $Id: softprofile.owl v1.1 13/01/10 $
                </owl:versionInfo>
                <rdfs:comment></rdfs:comment>
```

```
                <owl:imports rdf:resource="&profile;#"/>
        </owl:Ontology>

<profile:Profile rdf:ID="Profile">
        <catalog:presentedBy rdf:resource="&catalog;#ExampleCatalog"/>

        <profile:textDescription>
                        The catalog provides an example of how NPE-L might be deployed in a real
    world situation.
        </profile:textDescription>

        <profile:contactInformation>
                <actor:Actor rdf:ID="Contact">
                <actor:name>Example Owner</actor:name>
                <actor:title> Service Representative </actor:title>
                <actor:email>act@or.com</actor:email>
                </actor:Actor>
        </profile:contactInformation>

        <profile:serviceCategory>
                <role:BusinessCategory rdf:ID="NAICS-category">
                        <role:value>
                                E-tailers
                        </role:value>
                        <role:code>
                                454111
                        </role:code>
                </role:BusinessCategory >
        </profile:serviceCategory>

        <profile:hasRole rdf:resource="&roles;#Client"/>

        <catalog:hasNorm rdf:resource="&norms;#OverspecializedSearch"/>
        <catalog:hasNorm rdf:resource="&norms;#FinalizeOrder"/>
        <catalog:hasNorm rdf:resource="&norms;#CreditCardRequirement"/>
        <catalog:hasNorm rdf:resource="&norms;#ExtraSecurity"/>
        <catalog:hasNorm rdf:resource="&norms;#UnsupportedControl"/>
        <catalog:hasNorm rdf:resource="&norms;#SpellingError"/>
        <catalog:hasNorm rdf:resource="&norms;#UnsupportedLanguage"/>
        <catalog:hasNorm rdf:resource="&norms;#OverspecializedProcess"/>
        <catalog:hasNorm rdf:resource="&norms;#FinalOffer"/>
        <catalog:hasNorm rdf:resource="&norms;#FinalizeAction"/>
        <catalog:hasNorm rdf:resource="&norms;#StartWithRequest"/>
        <catalog:hasNorm rdf:resource="&norms;#GeneralisedSearch"/>
        <catalog:hasNorm rdf:resource="&norms;#AlterAfterVerification"/>
        <catalog:hasNorm rdf:resource="&norms;#StartWithPersonalData"/>
</profile:Profile>
</rdf:RDF>
```

A.4.8 softcatalog.npel

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE uridef[
  <!ENTITY rdf     "http://www.w3.org/1999/02/22-rdf-syntax-ns">
  <!ENTITY rdfs    "http://www.w3.org/2000/01/rdf-schema">
  <!ENTITY owl     "http://www.w3.org/2002/07/owl">
  <!ENTITY xsd     "http://www.w3.org/2001/XMLSchema">
```

```
  <!ENTITY catalog "http://web.me.com/adouglas/ontologies/npel/1.3/catalog.owl">
  <!ENTITY profile "http://web.me.com/adouglas/npel/hardprofile.npel">
  <!ENTITY norms   "http://web.me.com/adouglas/npel/hardnorms.npel">
  <!ENTITY THIS    "http://web.me.com/adouglas/npel/softcatalog.npel">
]>

<rdf:RDF        xmlns:rdf = "&rdf;#"
                        xmlns:rdfs = "&rdfs;#"
                        xmlns:xsd = "&xsd;#"
                        xmlns:owl = "&owl;#"
                        xmlns:profile = "&profile;#"
                        xmlns:norms = "&norms;#"
                        xmlns:catalog = "&catalog;#"
                        xmlns = "&THIS;#"
                        xml:base = "&THIS;">
        <owl:Ontology rdf:about="">
                <owl:versionInfo>
                         $Id: softcatalog.npel V1.1 13/01/10 $
                </owl:versionInfo>

                <rdfs:comment></rdfs:comment>

                <owl:imports rdf:resource="&profile;#"/>
                <owl:imports rdf:resource="&norms;#"/>
                <owl:imports rdf:resource="&catalog;#"/>
        </owl:Ontology>


        <catalog:Catalog rdf:ID="SoftCatalog">
                <!-- Reference to the Profile -->
                <catalog:presents rdf:resource="&profile;#ExampleProfile"/>
                <!-- Reference to the Norms -->
                <catalog:provides rdf:resource="&norms;#OverspecializedSearch"/>
                <catalog:provides rdf:resource="&norms;#FinalizeOrder"/>
                <catalog:provides rdf:resource="&norms;#CreditCardRequirement"/>
                <catalog:provides rdf:resource="&norms;#ExtraSecurity"/>
                <catalog:provides rdf:resource="&norms;#UnsupportedControl"/>
                <catalog:provides rdf:resource="&norms;#SpellingError"/>
                <catalog:provides rdf:resource="&norms;#UnsupportedLanguage"/>
                <catalog:provides rdf:resource="&norms;#OverspecializedProcess"/>
                <catalog:provides rdf:resource="&norms;#FinalOffer"/>
                <catalog:provides rdf:resource="&norms;#FinalizeAction"/>
                <catalog:provides rdf:resource="&norms;#StartWithRequest"/>
                <catalog:provides rdf:resource="&norms;#GeneralisedSearch"/>
                <catalog:provides rdf:resource="&norms;#AlterAfterVerification"/>
                <catalog:provides rdf:resource="&norms;#StartWithPersonalData"/>

        </catalog:Catalog>
</rdf:RDF>
```

A.4.9 softnorms.npel

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE uridef[
  <!ENTITY rdf        "http://www.w3.org/1999/02/22-rdf-syntax-ns">
  <!ENTITY rdfs       "http://www.w3.org/2000/01/rdf-schema">
  <!ENTITY owl        "http://www.w3.org/2002/07/owl">
```

```
  <!ENTITY xsd        "http://www.w3.org/2001/XMLSchema">
  <!ENTITY expr       "http://www.daml.org/services/owl-s/1.2/generic/Expression.owl">
  <!ENTITY swrl           "http://www.w3.org/2003/11/swrl">
  <!ENTITY fol            "http://www.w3.org/Submission/SWRL-FOL/">
  <!ENTITY concepts   "http://web.me.com/adouglas/npel/ontologies/concepts.owl">
  <!ENTITY norm       "http://web.me.com/adouglas/ontologies/npel/1.3/norm.owl">
  <!ENTITY activities "http://web.me.com/adouglas/npel/softactivities.npel">
  <!ENTITY roles          "http://web.me.com/adouglas/npel/softroles.npel">
  <!ENTITY exceptions "http://web.me.com/adouglas/npel/softexceptions.npel">
  <!ENTITY THIS       "http://web.me.com/adouglas/npel/softnorms.npel">
]>

<rdf:RDF        xmlns:rdf = "&rdf;#"
                        xmlns:rdfs = "&rdfs;#"
                        xmlns:xsd = "&xsd;#"
                        xmlns:owl = "&owl;#"
                        xmlns:expr = "&expr;#"
                        xmlns:swrl = "&swrl;#"
                        xmlns:fol = "&fol;#"
                        xmlns:norm = "&norm;#"
                        xmlns:concepts = "&concepts;#"
                        xmlns:roles = "&roles;#"
                        xmlns:exceptions = "&exceptions;#"
                        xmlns:activities = "&activities;#"
                        xmlns = "&THIS;#"
                        xml:base = "&THIS;">
        <owl:Ontology rdf:about="">
                <owl:versionInfo>
                         $Id: softnorms.npel V1.1 13/01/10 $
                </owl:versionInfo>

                <rdfs:comment></rdfs:comment>

                <owl:imports rdf:resource="&exceptions;#"/>
                <owl:imports rdf:resource="&roles;#"/>
                <owl:imports rdf:resource="&activities;#"/>
                <owl:imports rdf:resource="&norm;#"/>
                <owl:imports rdf:resource="&concepts;#"/>
        </owl:Ontology>


        <!-- Norm demonstrating specialisaion in search -->
        <norm:Obli rdf:ID="OverspecializedSearch">
                <norm:normIdentifier ref:datatype="&xsd;#String">
                        OverspecializedSearch
                </norm:normIdentifier>
                <norm:governsActivity rdf:resource="&activities;#OverspecializedSearchActivity" />
                <norm:hasRole rdf:resource="&roles;#Client" />
                <norm:hasInCondition>
                        <expr:SWRL-Condition>
                                <rdfs:label>
                                        next(#ChoosePaperActivity)
                                </rdfs:label>
                                <rdfs:comment>
                                        Test to see if the next step in the workflow handles a
    less specific data type.
                                </rdfs:comment>
                                <expr:expressionLanguage rdf:resource="&expr;#SWRL" />
                                <expr:expressionBody rdf:parseType="Literal">
```

```
                                <swrl:IndividualPropertyAtom>
                                        <swrl:propertyPredicate rdf:resource="&concepts;#
Next" />
                                        <swrl:argument1 rdf:resource="&roles;#Role" />
                                        <swrl:argument2 rdf:resource="&activities;#
ChoosePaperActivity" />
                                </swrl:IndividualPropertyAtom>
                        </expr:expressionBody>
                </expr:SWRL-Condition>
        </norm:hasInCondition>
        <norm:hasDCondition>
                <expr:SWRL-Condition>
                        <rdfs:label>
                                once(@end)
                        </rdfs:label>
                        <rdfs:comment>
                                Deactivates at the end of the workflow.
                        </rdfs:comment>
                        <expr:expressionLanguage rdf:resource="&expr;#SWRL" />
                        <expr:expressionBody rdf:parseType="Literal">
                                <swrl:IndividualPropertyAtom>
                                        <swrl:propertyPredicate rdf:resource="&concepts;#
Once" />
                                        <swrl:argument1 rdf:resource="&roles;#Role" />
                                        <swrl:argument2 rdf:resource="&concepts;#End" />
                                </swrl:IndividualPropertyAtom>
                        </expr:expressionBody>
                </expr:SWRL-Condition>
        </norm:hasDContition>
        <norm:triggersException rdf:resource="&exceptions;#OverspecializedSearchException"
 />
   </norm:Obli>



   <!-- Norm demonstrating finalisation requirement -->
   <norm:Obli rdf:ID="FinalizeOrder">
        <norm:normIdentifier ref:datatype="&xsd;#String">
                FinalizeOrder
        </norm:normIdentifier>
        <norm:governsActivity rdf:resource="&activities;#FinalizeOrderActivity" />
        <norm:hasRole rdf:resource="&roles;#Client" />
        <norm:hasInCondition>
                <expr:SWRL-Condition>
                        <rdfs:label>
                                once(#PaymentActivity)
                        </rdfs:label>
                        <rdfs:comment>
                                Activates after a payment.
                        </rdfs:comment>
                        <expr:expressionLanguage rdf:resource="&expr;#SWRL" />
                        <expr:expressionBody rdf:parseType="Literal">
                                <swrl:IndividualPropertyAtom>
                                        <swrl:propertyPredicate rdf:resource="&concepts;#
Once" />
                                        <swrl:argument1 rdf:resource="&roles;#Client" />
                                        <swrl:argument2 rdf:resource="&activities;#
PaymentActivity" />
                                </swrl:IndividualPropertyAtom>
```

```
                        </expr:expressionBody>
                </expr:SWRL-Condition>
        </norm:hasInCondition>
        <norm:hasDCondition>
                <expr:SWRL-Condition>
                        <rdfs:label>
                                once(#FinalizeOrderActivity)
                        </rdfs:label>
                        <rdfs:comment>
                                Deactivates when FinalizeOrderActivity is run.
                        </rdfs:comment>
                        <expr:expressionLanguage rdf:resource="&expr;#SWRL" />
                        <expr:expressionBody rdf:parseType="Literal">
                                <swrl:IndividualPropertyAtom>
                                        <swrl:propertyPredicate rdf:resource="&concepts;#
Once" />
                                        <swrl:argument1 rdf:resource="&roles;#Role" />
                                        <swrl:argument2 rdf:resource="&activities;#
FinalizeOrderActivity" />
                                </swrl:IndividualPropertyAtom>
                        </expr:expressionBody>
                </expr:SWRL-Condition>
        </norm:hasDContition>
        <norm:triggersException rdf:resource="&exceptions;#FinalizeOrderException" />
   </norm:Obli>



   <!-- Norm requirement to use credit not debit cards -->
   <norm:Forb rdf:ID="CreditCardRequirement">
        <norm:normIdentifier ref:datatype="&xsd;#String">
                CreditCardRequirement
        </norm:normIdentifier>
        <norm:governsActivity rdf:resource="&activities;#DebitCardActivity" />
        <norm:hasRole rdf:resource="&roles;#Client" />
        <norm:hasInCondition>
                <expr:SWRL-Condition>
                        <rdfs:label>
                                once(@start)
                        </rdfs:label>
                        <rdfs:comment>
                                Activates at the start of the workflow.
                        </rdfs:comment>
                        <expr:expressionLanguage rdf:resource="&expr;#SWRL" />
                        <expr:expressionBody rdf:parseType="Literal">
                                <swrl:IndividualPropertyAtom>
                                        <swrl:propertyPredicate rdf:resource="&concepts;#
Once" />
                                        <swrl:argument1 rdf:resource="&roles;#Role" />
                                        <swrl:argument2 rdf:resource="&concepts;#Start" />
                                </swrl:IndividualPropertyAtom>
                        </expr:expressionBody>
                </expr:SWRL-Condition>
        </norm:hasInCondition>
        <norm:hasDCondition>
                <expr:SWRL-Condition>
                        <rdfs:label>
                                once(@end)
                        </rdfs:label>
```

```
                        <rdfs:comment>
                                Deactivates at the end of the workflow.
                        </rdfs:comment>
                        <expr:expressionLanguage rdf:resource="&expr;#SWRL" />
                        <expr:expressionBody rdf:parseType="Literal">
                                <swrl:IndividualPropertyAtom>
                                        <swrl:propertyPredicate rdf:resource="&concepts;#
Once" />
                                        <swrl:argument1 rdf:resource="&roles;#Role" />
                                        <swrl:argument2 rdf:resource="&concepts;#End" />
                                </swrl:IndividualPropertyAtom>
                        </expr:expressionBody>
                </expr:SWRL-Condition>
        </norm:hasDContition>
        <norm:triggersException rdf:resource="&exceptions;#CreditCardRequirementException"
 />
   </norm:Forb>



   <!-- Norm showing unnecessary security steps -->
   <norm:Forb rdf:ID="ExtraSecurity">
        <norm:normIdentifier ref:datatype="&xsd;#String">
                ExtraSecurity
        </norm:normIdentifier>
        <norm:governsActivity rdf:resource="&activities;#SecurityActivity" />
        <norm:hasRole rdf:resource="&roles;#Client" />
        <norm:hasInCondition>
                <expr:SWRL-Condition>
                        <rdfs:label>
                                in(#SecurityActivity)
                        </rdfs:label>
                        <rdfs:comment>
                                Activates when security is activated.
                        </rdfs:comment>
                        <expr:expressionLanguage rdf:resource="&expr;#SWRL" />
                        <expr:expressionBody rdf:parseType="Literal">
                                <swrl:IndividualPropertyAtom>
                                        <swrl:propertyPredicate rdf:resource="&concepts;#
In" />
                                        <swrl:argument1 rdf:resource="&roles;#Client" />
                                        <swrl:argument2 rdf:resource="&activities;#
SecurityActivity" />
                                </swrl:IndividualPropertyAtom>
                        </expr:expressionBody>
                </expr:SWRL-Condition>
        </norm:hasInCondition>
        <norm:hasDCondition>
                <expr:SWRL-Condition>
                        <rdfs:label>
                                once(@end)
                        </rdfs:label>
                        <rdfs:comment>
                                Deacivates at the end of the workflow.
                        </rdfs:comment>
                        <expr:expressionLanguage rdf:resource="&expr;#SWRL" />
                        <expr:expressionBody rdf:parseType="Literal">
                                <swrl:IndividualPropertyAtom>
```

```
                                        <swrl:propertyPredicate rdf:resource="&concepts;#
Once" />
                                        <swrl:argument1 rdf:resource="&roles;#Role" />
                                        <swrl:argument2 rdf:resource="&concepts;#End" />
                                </swrl:IndividualPropertyAtom>
                        </expr:expressionBody>
                </expr:SWRL-Condition>
        </norm:hasDContition>
        <norm:triggersException rdf:resource="&exceptions;#ExtraSecurityException" />
    </norm:Forb>



    <!-- Norm forbiding unsupported control in workflows -->
    <norm:Forb rdf:ID="UnsupportedControl">
        <norm:normIdentifier ref:datatype="&xsd;#String">
            UnsupportedControl
        </norm:normIdentifier>
        <norm:governsActivity rdf:resource="&activities;#AnyActivity" />
        <norm:hasRole rdf:resource="&roles;#Client" />
        <norm:hasInCondition>
            <expr:SWRL-Condition>
                <rdfs:label>
                    equals(@current.construct,'Split-Join') or equals(@current
.construct,'Any-Order')
                </rdfs:label>
                <rdfs:comment>
                    Activates if the current control is not approved.
                </rdfs:comment>
                <expr:expressionLanguage rdf:resource="&expr;#SWRL" />
                <expr:expressionBody rdf:parseType="Literal">
                    <fol:Assertion>
                        <fol:Or>
                            <swrl:builtinAtom  swrl:builtin="&swrlb;#
equal">
                            <swrl:argument1 rdf:resource="&concepts;#
Current_Construct" />
                                <swrl:argument2 rdf:resource="&
activities;#Split-Join" />
                            </swrl:builtinAtom>
                            <swrl:builtinAtom  swrl:builtin="&swrlb;#equal
">
                            <swrl:argument1 rdf:resource="&concepts;#
Current_Construct" />
                                <swrl:argument2 rdf:resource="&
activities;#Any-Order" />
                            </swrl:builtinAtom>
                        </fol:Or>
                    <fol:Assertion>
                </expr:expressionBody>
            </expr:SWRL-Condition>
        </norm:hasInCondition>
        <norm:hasDCondition>
            <expr:SWRL-Condition>
                <rdfs:label>
                    not(equals(@current.construct,'')) and not(equals(@current
.construct,''))
                </rdfs:label>
                <rdfs:comment>
```

```
                                Deactivates when the current control is approved.
                        </rdfs:comment>
                        <expr:expressionLanguage rdf:resource="&expr;#SWRL" />
                        <expr:expressionBody rdf:parseType="Literal">
                                <fol:Assertion>
                                        <fol:And>
                                                <swrl:builtinAtom  swrl:builtin="&swrlb;#
notEqual">
                                                <swrl:argument1 rdf:resource="&concepts;#
Current_Construct" />
                                                        <swrl:argument2 rdf:resource="&
activities;#Split-Join" />
                                            </swrl:builtinAtom>
                                            <swrl:builtinAtom  swrl:builtin="&swrlb;#
notEqual">
                                                <swrl:argument1 rdf:resource="&concepts;#
Current_Construct" />
                                                        <swrl:argument2 rdf:resource="&
activities;#Any-Order" />
                                            </swrl:builtinAtom>
                                        </fol:And>
                                </fol:Assertion>
                        </expr:expressionBody>
                </expr:SWRL-Condition>
        </norm:hasDContition>
        <norm:triggersException rdf:resource="&exceptions;#UnsupportedControlException" />
    </norm:Forb>



    <!-- Norm detecting spelling mistakes in workflows -->
    <norm:Forb rdf:ID="SpellingError">
            <norm:normIdentifier ref:datatype="&xsd;#String">
                    SpellingError
            </norm:normIdentifier>
            <norm:governsActivity rdf:resource="&activities;#Any1to1Activity" />
            <norm:hasRole rdf:resource="&roles;#Client" />
            <norm:hasInCondition>
                    <expr:SWRL-Condition>
                            <rdfs:label>
                                    not(custom_function(spelling_en,@current.description))
                            </rdfs:label>
                            <rdfs:comment>
                                    Custom spelling function.
                            </rdfs:comment>
                            <expr:expressionLanguage rdf:resource="&expr;#SWRL" />
                            <expr:expressionBody rdf:parseType="Literal">
                                    <fol:Assertion>
                                            <fol:Exists>
                                                    <fol:Var type="&concepts;#Custom_Function"
>function</fol:Var>
                                                    <swrl:builtinAtom  swrl:builtin="&swrlb;#
booleanNot">
                                                    <fol:Atom>
                                                            <fol:slot>
                                                                  <fol:Ind>type</fol:Ind>
                                                                  <fol:Ind wref="&concepts;#
Spelling_Function"/>
                                                                </fol:slot>
```

```
                                                        <fol:slot>
                                                          <fol:Ind>loaded</fol:Ind>
                                                          <fol:Ind wref="&concepts;#
True"/>
                                                        </fol:slot>
                                                        <fol:slot>
                                                          <fol:Ind>object</Ind>
                                                          <fol:Var>function</fol:var>
                                                        </fol:slot>
                                                    </fol:Atom>
                                            </swrl:builtinAtom>
                                        </fol:Exists>
                                </fol:Assertion>
                        </expr:expressionBody>
                </expr:SWRL-Condition>
        </norm:hasInCondition>
        <norm:hasDCondition>
                <expr:SWRL-Condition>
                        <rdfs:label>
                                custom_function(spelling_en,@current.description)
                        </rdfs:label>
                        <rdfs:comment>
                                Custom spelling function.
                        </rdfs:comment>
                        <expr:expressionLanguage rdf:resource="&expr;#SWRL" />
                        <expr:expressionBody rdf:parseType="Literal">
                                <fol:Assertion>
                                        <fol:Exists>
                                                <fol:Var type="&concepts;#Custom_Function"
>function</fol:Var>
                                                <fol:Atom>
                                                        <fol:slot>
                                                              <fol:Ind>type</fol:Ind>
                                                              <fol:Ind wref="&concepts;#
Spelling_Function"/>
                                                            </fol:slot>
                                                            <fol:slot>
                                                              <fol:Ind>loaded</fol:Ind>
                                                              <fol:Ind wref="&concepts;#True"/>
                                                            </fol:slot>
                                                            <fol:slot>
                                                              <fol:Ind>object</Ind>
                                                              <fol:Var>function</fol:var>
                                                            </fol:slot>
                                                        </fol:Atom>
                                                </fol:Exists>
                                        </fol:Assertion>
                                </expr:expressionBody>
                        </expr:SWRL-Condition>
                </norm:hasDContition>
                <norm:triggersException rdf:resource="&exceptions;#SpellingErrorException" />
    </norm:Forb>



    <!-- Norm forbiding unsupported language in workflows -->
    <norm:Forb rdf:ID="UnsupportedLanguage">
            <norm:normIdentifier ref:datatype="&xsd;#String">
```

```
                UnsupportedLanguage
        </norm:normIdentifier>
        <norm:governsActivity rdf:resource="&activities;#Any1to1Activity" />
        <norm:hasRole rdf:resource="&roles;#Client" />
        <norm:hasInCondition>
                <expr:SWRL-Condition>
                        <rdfs:label>
                                not(equals(english,@current.language))
                        </rdfs:label>
                        <rdfs:comment>
                                Activates if the current language is not english.
                        </rdfs:comment>
                        <expr:expressionLanguage rdf:resource="&expr;#SWRL" />
                        <expr:expressionBody rdf:parseType="Literal">
                                <fol:Assertion>
                                        <fol:Exists>
                                                <fol:Var type="&concepts;#Current_Language
">language</fol:Var>
                                                <swrl:builtinAtom  swrl:builtin="&swrlb;#
booleanNot">
                                                <fol:Atom>
                                                            <fol:slot>
                                                              <fol:Ind>code</fol:Ind>
                                                              <fol:Ind wref="&concepts;#EN
"/>
                                                            </fol:slot>
                                                            <fol:slot>
                                                              <fol:Ind>language</Ind>
                                                              <fol:Var>language</fol:var>
                                                            </fol:slot>
                                                        </fol:Atom>
                                            </swrl:builtinAtom>
                                        </fol:Exists>
                                </fol:Assertion>
                        </expr:expressionBody>
                </expr:SWRL-Condition>
        </norm:hasInCondition>
        <norm:hasDCondition>
                <expr:SWRL-Condition>
                        <rdfs:label>
                                equals(english,@current.language)
                        </rdfs:label>
                        <rdfs:comment>
                                Activates if the current language is english.
                        </rdfs:comment>
                        <expr:expressionLanguage rdf:resource="&expr;#SWRL" />
                        <expr:expressionBody rdf:parseType="Literal">
                                <fol:Assertion>
                                        <fol:Exists>
                                                <fol:Var type="&concepts;#Current_Language
">language</fol:Var>
                                                <fol:Atom>
                                                    <fol:slot>
                                                      <fol:Ind>code</fol:Ind>
                                                      <fol:Ind wref="&concepts;#EN"/>
                                                    </fol:slot>
                                                    <fol:slot>
                                                      <fol:Ind>language</Ind>
                                                      <fol:Var>language</fol:var>
```

```
                                                    </fol:slot>
                                                </fol:Atom>
                                            </fol:Exists>
                                    </fol:Assertion>
                            </expr:expressionBody>
                    </expr:SWRL-Condition>
            </norm:hasDContition>
            <norm:triggersException rdf:resource="&exceptions;#UnsupportedLanguageException" /
>
    </norm:Forb>

    <!-- Norm demonstrating specialisaion in processing -->
    <norm:Obli rdf:ID="OverspecializedProcess">
            <norm:normIdentifier ref:datatype="&xsd;#String">
                    OverspecializedProcess
            </norm:normIdentifier>
            <norm:governsActivity rdf:resource="&activities;#OverspecializedProcessActivity" /
>
            <norm:hasRole rdf:resource="&roles;#Client" />
            <norm:hasInCondition>
                    <expr:SWRL-Condition>
                            <rdfs:label>
                                    next(#UseItineraryActivity)
                            </rdfs:label>
                            <rdfs:comment>
                                    Activates if the next step uses the outcome of
OverspecializedProcessActivity
                            </rdfs:comment>
                            <expr:expressionLanguage rdf:resource="&expr;#SWRL" />
                            <expr:expressionBody rdf:parseType="Literal">
                                    <swrl:IndividualPropertyAtom>
                                            <swrl:propertyPredicate rdf:resource="&concepts;#
Next" />
                                            <swrl:argument1 rdf:resource="&roles;#Client" />
                                            <swrl:argument2 rdf:resource="&activities;#
UseItineraryActivity" />
                                    </swrl:IndividualPropertyAtom>
                            </expr:expressionBody>
                    </expr:SWRL-Condition>
            </norm:hasInCondition>
            <norm:hasDCondition>
                    <expr:SWRL-Condition>
                            <rdfs:label>
                                    once(#UseItineraryActivity)
                            </rdfs:label>
                            <rdfs:comment>
                                    Deactivates if the current step uses the outcome of
OverspecializedProcessActivity
                            </rdfs:comment>
                            <expr:expressionLanguage rdf:resource="&expr;#SWRL" />
                            <expr:expressionBody rdf:parseType="Literal">
                                    <swrl:IndividualPropertyAtom>
                                            <swrl:propertyPredicate rdf:resource="&concepts;#
Once" />
                                            <swrl:argument1 rdf:resource="&roles;#Role" />
                                            <swrl:argument2 rdf:resource="&activities;#
UseItineraryActivity" />
                                    </swrl:IndividualPropertyAtom>
                            </expr:expressionBody>
```

```
                </expr:SWRL-Condition>
            </norm:hasDContition>
            <norm:triggersException rdf:resource="&exceptions;#OverspecializedProcessException
" />
    </norm:Obli>



    <!-- Norm demonstrating finalisation requirement -->
    <norm:Obli rdf:ID="FinalizeAction">
            <norm:normIdentifier ref:datatype="&xsd;#String">
                FinalizeAction
            </norm:normIdentifier>
            <norm:governsActivity rdf:resource="&activities;#FinalizeActionActivity" />
            <norm:hasRole rdf:resource="&roles;#Client" />
            <norm:hasInCondition>
                <expr:SWRL-Condition>
                    <rdfs:label>
                        next(@end)
                    </rdfs:label>
                    <rdfs:comment>
                        Activates next but one from the end of the workflow.
                    </rdfs:comment>
                    <expr:expressionLanguage rdf:resource="&expr;#SWRL" />
                    <expr:expressionBody rdf:parseType="Literal">
                        <swrl:IndividualPropertyAtom>
                            <swrl:propertyPredicate rdf:resource="&concepts;#
Next" />
                            <swrl:argument1 rdf:resource="&roles;#Role" />
                            <swrl:argument2 rdf:resource="&concepts;#End" />
                        </swrl:IndividualPropertyAtom>
                    </expr:expressionBody>
                </expr:SWRL-Condition>
            </norm:hasInCondition>
            <norm:hasDCondition>
                <expr:SWRL-Condition>
                    <rdfs:label>
                        once(@end)
                    </rdfs:label>
                    <rdfs:comment>
                        Deactiviates at the end of the workflow
                    </rdfs:comment>
                    <expr:expressionLanguage rdf:resource="&expr;#SWRL" />
                    <expr:expressionBody rdf:parseType="Literal">
                        <swrl:IndividualPropertyAtom>
                            <swrl:propertyPredicate rdf:resource="&concepts;#
Once" />
                            <swrl:argument1 rdf:resource="&roles;#Role" />
                            <swrl:argument2 rdf:resource="&concepts;#End" />
                        </swrl:IndividualPropertyAtom>
                    </expr:expressionBody>
                </expr:SWRL-Condition>
            </norm:hasDContition>
            <norm:triggersException rdf:resource="&exceptions;#FinalizeActionException" />
    </norm:Obli>



    <!-- Norm permitting a final offer -->
```

```
    <norm:Perm rdf:ID="FinalOffer">
            <norm:normIdentifier ref:datatype="&xsd;#String">
                    FinalOffer
            </norm:normIdentifier>
            <norm:governsActivity rdf:resource="&activities;#FinaliseOfferActivity" />
            <norm:hasRole rdf:resource="&roles;#Client" />
            <norm:hasInCondition>
                    <expr:SWRL-Condition>
                            <rdfs:label>
                                    once(#TransactionOfferActivity)
                            </rdfs:label>
                            <rdfs:comment>
                                    Activates once an initial offer has been made
                            </rdfs:comment>
                            <expr:expressionLanguage rdf:resource="&expr;#SWRL" />
                            <expr:expressionBody rdf:parseType="Literal">
                                    <swrl:IndividualPropertyAtom>
                                            <swrl:propertyPredicate rdf:resource="&concepts;#
Once" />
                                            <swrl:argument1 rdf:resource="&roles;#Role" />
                                            <swrl:argument2 rdf:resource="&activities;#
TransactionOfferActivity" />
                                    </swrl:IndividualPropertyAtom>
                            </expr:expressionBody>
                    </expr:SWRL-Condition>
            </norm:hasInCondition>
            <norm:hasDCondition>
                    <expr:SWRL-Condition>
                            <rdfs:label>
                                    once(#FinaliseOfferActivity)
                            </rdfs:label>
                            <rdfs:comment>
                                    Deactivates upon final offer
                            </rdfs:comment>
                            <expr:expressionLanguage rdf:resource="&expr;#SWRL" />
                            <expr:expressionBody rdf:parseType="Literal">
                                    <swrl:IndividualPropertyAtom>
                                            <swrl:propertyPredicate rdf:resource="&concepts;#
Once" />
                                            <swrl:argument1 rdf:resource="&roles;#Role" />
                                            <swrl:argument2 rdf:resource="&activities;#
FinaliseOfferActivity" />
                                    </swrl:IndividualPropertyAtom>
                            </expr:expressionBody>
                    </expr:SWRL-Condition>
            </norm:hasDContition>
            <norm:triggersException rdf:resource="&exceptions;#FinalOfferException" />
    </norm:Perm>



    <!-- Norm permitting a full login process -->
    <norm:Perm rdf:ID="FullLogin">
            <norm:normIdentifier ref:datatype="&xsd;#String">
                    FullLogin
            </norm:normIdentifier>
            <norm:governsActivity rdf:resource="&activities;#FullLoginActivity" />
            <norm:hasRole rdf:resource="&roles;#Client" />
            <norm:hasInCondition>
```

```
                    <expr:SWRL-Condition>
                            <rdfs:label>
                                    once(@start)
                            </rdfs:label>
                            <rdfs:comment>
                                    Activates at start of workflow.
                            </rdfs:comment>
                            <expr:expressionLanguage rdf:resource="&expr;#SWRL" />
                            <expr:expressionBody rdf:parseType="Literal">
                                    <swrl:IndividualPropertyAtom>
                                            <swrl:propertyPredicate rdf:resource="&concepts;#
Once" />
                                            <swrl:argument1 rdf:resource="&roles;#Role" />
                                            <swrl:argument2 rdf:resource="&concepts;#Start" />
                                    </swrl:IndividualPropertyAtom>
                            </expr:expressionBody>
                    </expr:SWRL-Condition>
            </norm:hasInCondition>
            <norm:hasDCondition>
                    <expr:SWRL-Condition>
                            <rdfs:label>
                                    once(@end)
                            </rdfs:label>
                            <rdfs:comment>
                                    Deactivates at end of workflow.
                            </rdfs:comment>
                            <expr:expressionLanguage rdf:resource="&expr;#SWRL" />
                            <expr:expressionBody rdf:parseType="Literal">
                                    <swrl:IndividualPropertyAtom>
                                            <swrl:propertyPredicate rdf:resource="&concepts;#
Once" />
                                            <swrl:argument1 rdf:resource="&roles;#Role" />
                                            <swrl:argument2 rdf:resource="&concepts;#End" />
                                    </swrl:IndividualPropertyAtom>
                            </expr:expressionBody>
                    </expr:SWRL-Condition>
            </norm:hasDContition>
            <norm:triggersException rdf:resource="&exceptions;#FullLoginException" />
    </norm:Perm>



    <!-- Norm permitting a full login process -->
    <norm:Obli rdf:ID="RequestResponsePattern">
            <norm:normIdentifier ref:datatype="&xsd;#String">
                    RequestResponsePattern
            </norm:normIdentifier>
            <norm:governsActivity rdf:resource="&activities;#ResponseActivity" />
            <norm:hasRole rdf:resource="&roles;#Client" />
            <norm:hasInCondition>
                    <expr:SWRL-Condition>
                            <rdfs:label>
                                    once(#RequestActivity)
                            </rdfs:label>
                            <rdfs:comment>
                                    Activates upon a request
                            </rdfs:comment>
                            <expr:expressionLanguage rdf:resource="&expr;#SWRL" />
                            <expr:expressionBody rdf:parseType="Literal">
```

```
                                <swrl:IndividualPropertyAtom>
                                        <swrl:propertyPredicate rdf:resource="&concepts;#
Once" />
                                        <swrl:argument1 rdf:resource="&roles;#Client" />
                                        <swrl:argument2 rdf:resource="&activities;#
RequestActivity" />
                                </swrl:IndividualPropertyAtom>
                        </expr:expressionBody>
                </expr:SWRL-Condition>
        </norm:hasInCondition>
        <norm:hasDCondition>
                <expr:SWRL-Condition>
                        <rdfs:label>
                                once(#ResponseActivity) or once(@end)
                        </rdfs:label>
                        <rdfs:comment>
                                Deactivates upon second request or end of workflow.
                        </rdfs:comment>
                        <expr:expressionLanguage rdf:resource="&expr;#SWRL" />
                        <expr:expressionBody rdf:parseType="Literal">
                                <fol:Assertion>
                                        <fol:Or>
                                                <swrl:IndividualPropertyAtom>
                                                        <swrl:propertyPredicate
rdf:resource="&concepts;#Once" />
                                                        <swrl:argument1 rdf:resource="&
roles;#Client" />
                                                        <swrl:argument2 rdf:resource="&
activities;#ResponseActivity" />
                                                </swrl:IndividualPropertyAtom>
                                                <swrl:IndividualPropertyAtom>
                                                        <swrl:propertyPredicate
rdf:resource="&concepts;#Once" />
                                                        <swrl:argument1 rdf:resource="&
roles;#Role" />
                                                        <swrl:argument2 rdf:resource="&
concepts;#End" />
                                                </swrl:IndividualPropertyAtom>
                                        </fol:Or>
                                </fol:Assertion>
                        </expr:expressionBody>
                </expr:SWRL-Condition>
        </norm:hasDContition>
        <norm:triggersException rdf:resource="&exceptions;#RequestResponsePatternException
" />
    </norm:Perm>

    <!-- Norm forbiding generalised search -->
    <norm:Forb rdf:ID="GeneralisedSearch">
        <norm:normIdentifier ref:datatype="&xsd;#String">
                GeneralisedSearch
        </norm:normIdentifier>
        <norm:governsActivity rdf:resource="&activities;#GeneralisedSearchActivity" />
        <norm:hasRole rdf:resource="&roles;#Searcher" />
        <norm:hasInCondition>
                <expr:SWRL-Condition>
                        <rdfs:label>
                                once(@start)
                        </rdfs:label>
```

```
                        <rdfs:comment>
                                Activates at start of workflow.
                        </rdfs:comment>
                        <expr:expressionLanguage rdf:resource="&expr;#SWRL" />
                        <expr:expressionBody rdf:parseType="Literal">
                                <swrl:IndividualPropertyAtom>
                                        <swrl:propertyPredicate rdf:resource="&concepts;#
Once" />
                                        <swrl:argument1 rdf:resource="&roles;#Role" />
                                        <swrl:argument2 rdf:resource="&concepts;#Start" />
                                </swrl:IndividualPropertyAtom>
                        </expr:expressionBody>
                </expr:SWRL-Condition>
        </norm:hasInCondition>
        <norm:hasDCondition>
                <expr:SWRL-Condition>
                        <rdfs:label>
                                once(@end)
                        </rdfs:label>
                        <rdfs:comment>
                                Deactivates at end of workflow.
                        </rdfs:comment>
                        <expr:expressionLanguage rdf:resource="&expr;#SWRL" />
                        <expr:expressionBody rdf:parseType="Literal">
                                <swrl:IndividualPropertyAtom>
                                        <swrl:propertyPredicate rdf:resource="&concepts;#
Once" />
                                        <swrl:argument1 rdf:resource="&roles;#Role" />
                                        <swrl:argument2 rdf:resource="&concepts;#End" />
                                </swrl:IndividualPropertyAtom>
                        </expr:expressionBody>
                </expr:SWRL-Condition>
        </norm:hasDContition>
        <norm:triggersException rdf:resource="&exceptions;#GeneralisedSearchException" />
    </norm:Forb>

    <!-- Norm forbidding altering items to a list once it's been verified -->
    <norm:Forb rdf:ID="AlterAfterVerification">
            <norm:normIdentifier ref:datatype="&xsd;#String">
                    AlterAfterVerification
            </norm:normIdentifier>
            <norm:governsActivity rdf:resource="&activities;#AddActivity" />
            <norm:hasRole rdf:resource="&roles;#Role" />
            <norm:hasInCondition>
                    <expr:SWRL-Condition>
                            <rdfs:label>
                                    once(#VerifyActivity)
                            </rdfs:label>
                            <rdfs:comment>
                                    Activates upon verification.
                            </rdfs:comment>
                            <expr:expressionLanguage rdf:resource="&expr;#SWRL" />
                            <expr:expressionBody rdf:parseType="Literal">
                                    <swrl:IndividualPropertyAtom>
                                            <swrl:propertyPredicate rdf:resource="&concepts;#
Once" />
                                            <swrl:argument1 rdf:resource="&roles;#Role" />
                                            <swrl:argument2 rdf:resource="&activities;#
VerifyActivity" />
```

```
                                </swrl:IndividualPropertyAtom>
                        </expr:expressionBody>
                </expr:SWRL-Condition>
        </norm:hasInCondition>
        <norm:hasDCondition>
                <expr:SWRL-Condition>
                        <rdfs:label>
                                once(@end)
                        </rdfs:label>
                        <rdfs:comment>
                                Deactivates at end of workflow.
                        </rdfs:comment>
                        <expr:expressionLanguage rdf:resource="&expr;#SWRL" />
                        <expr:expressionBody rdf:parseType="Literal">
                                <swrl:IndividualPropertyAtom>
                                        <swrl:propertyPredicate rdf:resource="&concepts;#
Once" />
                                        <swrl:argument1 rdf:resource="&roles;#Role" />
                                        <swrl:argument2 rdf:resource="&concepts;#End" />
                                </swrl:IndividualPropertyAtom>
                        </expr:expressionBody>
                </expr:SWRL-Condition>
        </norm:hasDContition>
        <norm:triggersException rdf:resource="&exceptions;#PostVerifyException" />
   </norm:Forb>

   <!-- Norm forbidding sending of personal data during initial interactions -->
   <norm:Forb rdf:ID="StartWithPersonalData">
        <norm:normIdentifier ref:datatype="&xsd;#String">
                StartWithPersonalData
        </norm:normIdentifier>
        <norm:governsActivity rdf:resource="&activities;#SendPersonalDataActivity" />
        <norm:hasRole rdf:resource="&roles;#Client" />
        <norm:hasInCondition>
                <expr:SWRL-Condition>
                        <rdfs:label>
                                once(@start)
                        </rdfs:label>
                        <rdfs:comment>
                                Activates at start of workflow.
                        </rdfs:comment>
                        <expr:expressionLanguage rdf:resource="&expr;#SWRL" />
                        <expr:expressionBody rdf:parseType="Literal">
                                <swrl:IndividualPropertyAtom>
                                        <swrl:propertyPredicate rdf:resource="&concepts;#
Once" />
                                        <swrl:argument1 rdf:resource="&roles;#Role" />
                                        <swrl:argument2 rdf:resource="&concepts;#Start" />
                                </swrl:IndividualPropertyAtom>
                        </expr:expressionBody>
                </expr:SWRL-Condition>
        </norm:hasInCondition>
        <norm:hasDCondition>
                <expr:SWRL-Condition>
                        <rdfs:label>
                                once(#AnyActivity)
                        </rdfs:label>
                        <rdfs:comment>
                                Deactivates after any activity
```

```
                        </rdfs:comment>
                        <expr:expressionLanguage rdf:resource="&expr;#SWRL" />
                        <expr:expressionBody rdf:parseType="Literal">
                                <swrl:IndividualPropertyAtom>
                                        <swrl:propertyPredicate rdf:resource="&concepts;#
Once" />
                                        <swrl:argument1 rdf:resource="&roles;#Role" />
                                        <swrl:argument2 rdf:resource="&activities;#
AnyActivity" />
                                </swrl:IndividualPropertyAtom>
                        </expr:expressionBody>
                </expr:SWRL-Condition>
        </norm:hasDContition>
        <norm:triggersException rdf:resource="&exceptions;#MinorPersonalDataException" />
   </norm:Forb>

   <!-- Norm Obligating that users start with a request -->
   <norm:Obli rdf:ID="StartWithRequest">
        <norm:normIdentifier ref:datatype="&xsd;#String">
                StartWithRequest
        </norm:normIdentifier>
        <norm:governsActivity rdf:resource="&activities;#Any1To1Activity" />
        <norm:hasRole rdf:resource="&roles;#Client" />
        <norm:hasInCondition>
                <expr:SWRL-Condition>
                        <rdfs:label>
                                once(@start)
                        </rdfs:label>
                        <rdfs:comment>
                                Activates at start of workflow.
                        </rdfs:comment>
                        <expr:expressionLanguage rdf:resource="&expr;#SWRL" />
                        <expr:expressionBody rdf:parseType="Literal">
                                <swrl:IndividualPropertyAtom>
                                        <swrl:propertyPredicate rdf:resource="&concepts;#
Once" />
                                        <swrl:argument1 rdf:resource="&roles;#Role" />
                                        <swrl:argument2 rdf:resource="&concepts;#Start" />
                                </swrl:IndividualPropertyAtom>
                        </expr:expressionBody>
                </expr:SWRL-Condition>
        </norm:hasInCondition>
        <norm:hasDCondition>
                <expr:SWRL-Condition>
                        <rdfs:label>
                                once(#AnyActivity)
                        </rdfs:label>
                        <rdfs:comment>
                                Deactivates after any activity.
                        </rdfs:comment>
                        <expr:expressionLanguage rdf:resource="&expr;#SWRL" />
                        <expr:expressionBody rdf:parseType="Literal">
                                <swrl:IndividualPropertyAtom>
                                        <swrl:propertyPredicate rdf:resource="&concepts;#
Once" />
                                        <swrl:argument1 rdf:resource="&roles;#Role" />
                                        <swrl:argument2 rdf:resource="&activities;#
AnyActivity" />
                                </swrl:IndividualPropertyAtom>
```

```
                                </expr:expressionBody>
                            </expr:SWRL-Condition>
                        </norm:hasDContition>
                        <norm:triggersException rdf:resource="&exceptions;#RequestResponsePatternException
    " />
        </norm:Obli>
</rdf:RDF>
```

A.5 OWL-S Test Cases

A.5.1 FindExpert.owls

```
<?xml version="1.0" encoding="WINDOWS-1252"?>
<!DOCTYPE uridef [
  <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema">
  <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns">
  <!ENTITY owl "http://www.w3.org/2002/07/owl">
  <!ENTITY xsd "http://www.w3.org/2001/XMLSchema">
  <!ENTITY service "http://www.daml.org/services/owl-s/1.2/Service.owl">
  <!ENTITY profile "http://www.daml.org/services/owl-s/1.2/Profile.owl">
  <!ENTITY process "http://www.daml.org/services/owl-s/1.2/Process.owl">
  <!ENTITY grounding "http://www.daml.org/services/owl-s/1.2/Grounding.owl">
  <!ENTITY list "http://www.daml.org/services/owl-s/1.2/generic/ObjectList.owl">
  <!ENTITY swrl "http://www.w3.org/2003/11/swrl">
  <!ENTITY swrlb "http://www.w3.org/2003/11/swrlb">
  <!ENTITY expr "http://www.daml.org/services/owl-s/1.2/generic/Expression.owl">
  <!ENTITY concepts "http://testcases.freya.local/ontology/concepts.owl">
  <!ENTITY THIS "http://testcases.freya.local/info/findExpert.owls">
]>

<rdf:RDF
        xmlns:rdf="&rdf;#"
        xmlns:xsd="&xsd;#"
        xmlns:rdfs="&rdfs;#"
        xmlns:owl="&owl;#"
        xmlns:expr="&expr;#"
        xmlns:service="&service;#"
        xmlns:grounding="&grounding;#"
    xmlns:process="&process;#"
        xmlns:profile="&profile;#"
    xmlns:list="&list;#"
    xmlns:swrl="&swrl;#"
    xmlns:swrlb="&swrlb;#"
        xmlns:concepts="&concepts;#"
    xmlns="&THIS;#"
        xml:base="&THIS;#">

  <owl:Ontology rdf:about="">
        <owl:versionInfo>
                $ID: findExpert.owls, v 1.4 2011/02/28 akd07r $
        </owl:versionInfo>
        <rdfs:comment>
                TODO
        </rdfs:comment>
        <owl:imports rdf:resource="&service;" />
```

```
      <owl:imports rdf:resource="&process;" />
      <owl:imports rdf:resource="&profile;" />
      <owl:imports rdf:resource="&grounding;" />
      <owl:imports rdf:resource="&concepts;" />
</owl:Ontology>


<!--
=====================================================
=====================================================
      Service Model
=====================================================
=====================================================
-->
<service:Service rdf:ID="FindExpertService">
      <service:presents rdf:resource="#FindExpertProfile"/>
      <service:describedBy rdf:resource="#FindExpert"/>
      <!--<service:supports rdf:resource="#FindExpertServiceGrounding"/>-->
</service:Service>

<!--
=====================================================
=====================================================
      Profile Model
=====================================================
=====================================================
-->
<profile:Profile rdf:ID="FindExpertProfile">
      <profile:serviceName>
              FindExpert_Service
      </profile:serviceName>
      <profile:textDescription>
      TODO
      </profile:textDescription>

      <service:presentedBy rdf:resource="#FindExpertService"/>
      <profile:has_process rdf:resource="#FindExpert"/>

      <profile:hasInput rdf:resource="#FindExpertSearchTerm"/>
      <profile:hasInput rdf:resource="#FindExpertAcademicPaperList"/>
      <profile:hasInput rdf:resource="#FindExpertAcademicPaper"/>
      <profile:hasInput rdf:resource="#FindExpertCreateAccInfo"/>
      <profile:hasInput rdf:resource="#FindExpertSignInData"/>
      <profile:hasInput rdf:resource="#FindExpertAuthor"/>
      <profile:hasInput rdf:resource="#FindExpertMessage"/>

      <profile:hasOutput rdf:resource="#FindExpert"/>
      <profile:hasOutput rdf:resource="#FindExpert"/>
</profile:Profile>

<!--
=====================================================
=====================================================
      Process Model
=====================================================
=====================================================
-->

<!--
```

```
=====================================================
      CompositeProcess: FindExpert
=====================================================
-->

<process:CompositeProcess rdf:ID="FindExpert">
      <rdfs:comment>
              TODO
      </rdfs:comment>

      <process:hasInput>
              <process:Input rdf:ID="FindExpertSearchTerm">
                      <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#SearchTerm</
   process:parameterType>
              </process:Input>
      </process:hasInput>

      <process:hasInput>
              <process:Input rdf:ID="FindExpertAcademicPaperList">
                      <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#
   AcademicPaperList</process:parameterType>
              </process:Input>
      </process:hasInput>

      <process:hasInput>
              <process:Input rdf:ID="FindExpertAcademicPaper">
                      <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#
   AcademicPaper</process:parameterType>
              </process:Input>
      </process:hasInput>

      <process:hasInput>
              <process:Input rdf:ID="FindExpertCreateAccInfo">
                      <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#AcctInfo</
   process:parameterType>
              </process:Input>
      </process:hasInput>

      <process:hasInput>
              <process:Input rdf:ID="FindExpertSignInData">
                      <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#SignInData</
   process:parameterType>
              </process:Input>
      </process:hasInput>

      <process:hasInput>
              <process:Input rdf:ID="FindExpertAuthor">
                      <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#Author</
   process:parameterType>
              </process:Input>
      </process:hasInput>

      <process:hasInput>
              <process:Input rdf:ID="FindExpertMessage">
                      <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#Message</
   process:parameterType>
              </process:Input>
      </process:hasInput>
```

```
   <process:hasOutput>
           <process:Output rdf:ID="FindExpertCreateAcctOutput">
                   <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#AcctID</
process:parameterType>
           </process:Output>
   </process:hasOutput>

   <process:hasOutput>
           <process:Output rdf:ID="FindExpertMsgSendConfirmation">
                   <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#
MsgSendConfirmation</process:parameterType>
           </process:Output>
   </process:hasOutput>

   <process:composedOf>
           <process:sequence>
                   <process:components>
                           <process:ControlConstructList>
                                   <list:first>
                                           <process:Perform rdf:ID="
FindExpertFullSearchRepositoryPerform">
                                                   <process:process rdf:resource="#
FindExpertFullSearchRepository" />
                                                   <process:hasDataFrom>
                                                           <process:InputBinding>
                                                                   <process:toParam
rdf:resource="#SearchRepositorySearchTerm" />
                                                                   <process:valueSource>
                                                                           <process:ValueOf>
                                                                                   <
process:theVar rdf:resource="#FindExpertSearchTerm" />
                                                                                   <
process:fromProcess rdf:resource="&process;#TheParentPerform" />
                                                                           </process:ValueOf>
                                                                   </process:valueSource>
                                                           </process:InputBinding>
                                                   </process:hasDataFrom>
                                           </process:Perform>
                                   </list:first>
                                   <list:rest>
                                           <process:ControlConstructList>
                                   <list:first>
                                           <process:Perform rdf:ID="SignInAlternativesPerform
">
                                           <process:process rdf:resource="#SignInAlternatives
"/>
                                                   <process:hasDataFrom>
                                                   <process:InputBinding>
                                                           <process:toParam rdf:resource="#
FindExpertAlternativesCreateAcctInfo"/>
                                                           <process:valueSource>
                                                                   <process:ValueOf>
                                                                           <process:theVar
rdf:resource="#FindExpertCreateAcctInfo"/>
                                                                           <
process:fromProcess rdf:resource="&process;#TheParentPerform"/>
                                                                   </process:ValueOf>
                                                                   </process:valueSource>
                                                           </process:InputBinding>
```

```
                                                        </process:hasDataFrom>
                                                <process:hasDataFrom>
                                                        <process:InputBinding>
                                                        <process:toParam rdf:resource="#
FindExpertAlternativesSignInData"/>
                                                        <process:valueSource>
                                                                <process:ValueOf>
                                                                        <process:theVar
rdf:resource="#FindExpertSignInData"/>
                                                                        <
process:fromProcess rdf:resource="&process;#TheParentPerform"/>
                                                                </process:ValueOf>
                                                        </process:valueSource>
                                                </process:InputBinding>
                                                </process:hasDataFrom>
                                        </process:Perform>
                                </list:first>
                                                <list:rest>
                                                        <process:ControlConstructList>
                                                                <list:first>
                                                                        <process:Perform
rdf:ID="GetUserInfoPerform">
                                                                                <
process:process rdf:resource="#GetUserInfo" />
                                                                                <
process:hasDataFrom>
                                                                                        <
process:InputBinding>

        <process:toParam rdf:resource="#AuthorUserName" />

        <process:valueSource>

                <process:ValueOf>

                        <process:theVar rdf:resource="#ChosenAuthor" />

                        <process:fromProcess rdf:resource="&process;#
FindExpertFullSearchRepositoryPerform" />

                </process:ValueOf>

        </process:valueSource>
                                                                                        </
process:InputBinding>
                                                                                </
process:hasDataFrom>
                                                                        </process:Perform>
                                                                </list:first>
                                                                <list:rest>
                                                                        <
process:ControlConstructList>
                                                                                <
list:first>
                                                                                        <
process:Perform rdf:ID="SendMessagePerform">

        <process:process rdf:resource="#SendMessage" />
```

```
        <process:hasDataFrom>
                <process:InputBinding>
                        <process:toParam rdf:resource="#SendMessageMessage" />
                        <process:valueSource>
                                <process:ValueOf>
                                        <process:theVar rdf:resource="#FindExpertMessage" />
                                        <process:fromProcess rdf:resource="&process;#
TheParentPerform" />
                                </process:ValueOf>
                        </process:valueSource>
                </process:InputBinding>
        </process:hasDataFrom>
        <process:hasDataFrom>
                <process:InputBinding>
                        <process:toParam rdf:resource="#SendMessageUser" />
                        <process:valueSource>
                                <process:ValueOf>
                                        <process:theVar rdf:resource="#ExpertUserID" />
                                        <process:fromProcess rdf:resource="&process;#
GetUserInfoPerform" />
                                </process:ValueOf>
                        </process:valueSource>
                </process:InputBinding>
        </process:hasDataFrom>
                                                                                </
process:Perform>
                                                                        </
list:first>
                                                                        <list:rest
 rdf:resource="&list;#nil"/>
                                                                </
process:ControlConstructList>
                                                        </list:rest>
                                                </process:ControlConstructList>
                                        </list:rest>
                                </process:ControlConstructList>
                        </list:rest>
```

```
                        </process:ControlConstructList>
                    </process:components>
                </process:sequence>
        </process:composedOf>

        <process:hasResult>
                <process:Result rdf:ID="FindExpertCreateAcctOutputResult">
                <process:inCondition rdf:resource="&expr;#AlwaysTrue"/>
                <process:withOutput>
                        <process:OutputBinding>
                                <process:toParam rdf:resource="#FindExpertCreateAcctOutput"/>
                                <process:valueSource>
                                <process:ValueOf>
                                        <process:theVar rdf:resource="#SignInAlternativesAcctID"/>
                                        <process:fromProcess rdf:resource="#
    SignInAlternativesPerform"/>
                                </process:ValueOf>
                                </process:valueSource>
                        </process:OutputBinding>
                </process:withOutput>
            </process:Result>
        </process:hasResult>
        <process:hasResult>
                <process:Result rdf:ID="FindExpertSendMessageResult">
                <process:inCondition rdf:resource="&expr;#AlwaysTrue"/>
                <process:withOutput>
                        <process:OutputBinding>
                                <process:toParam rdf:resource="#FindExpertMsgSendConfirmation"/>
                                <process:valueSource>
                                <process:ValueOf>
                                        <process:theVar rdf:resource="#SendMessageConfirm"/>
                                        <process:fromProcess rdf:resource="#sendMessagePerform"/>
                                </process:ValueOf>
                                </process:valueSource>
                        </process:OutputBinding>
                </process:withOutput>
            </process:Result>
        </process:hasResult>

</process:CompositeProcess>


<!--
=======================================================
      CompositeProcess: FindExpertFullSearchRepository
=======================================================
-->

<process:CompositeProcess rdf:ID="FindExpertFullSearchRepository">
      <rdfs:comment>
      </rdfs:comment>

      <process:hasInput>
              <process:Input rdf:ID="SearchRepositorySearchTerm">
                      <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#SearchTerm</
    process:parameterType>
              </process:Input>
      </process:hasInput>
```

```
   <process:hasOutput>
           <process:Output rdf:ID="ChosenAuthor">
                   <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#Author</
process:parameterType>
           </process:Output>
   </process:hasOutput>

   <process:composedOf>
           <process:sequence>
                   <process:components>
                           <process:ControlConstructList>
                                   <list:first>
                                           <process:Perform rdf:ID="SearchRepositoryPerform">
                                                   <process:process rdf:resource="
SearchRepository" />
                                                   <process:hasDataFrom>
                                                           <process:InputBinding>
                                                                   <process:toParam
rdf:resource="#AtomicSearchTerm" />
                                                                   <process:valueSource>
                                                                           <process:ValueOf>
                                                                                   <
process:theVar rdf:resource="#SearchRepositorySearchTerm" />
                                                                                   <
process:fromProcess rdf:resource="&process;#TheParentPerform" />
                                                                           </process:ValueOf>
                                                                   </process:valueSource>
                                                           </process:InputBinding>
                                                   </process:hasDataFrom>
                                           </process:Perform>
                                   </list:first>
                                   <list:rest>
                                           <process:ControlConstructList>
                                                   <list:first>
                                                           <process:Perform rdf:ID="
ChoosePaperPerform">
                                                                   <process:process
rdf:resource="#ChoosePaper" />
                                                                   <process:hasDataFrom>
                                                                           <
process:InputBinding>
                                                                                   <
process:toParam rdf:resource="#AtomicPaperList" />
                                                                                   <
process:valueSource>
                                                                                           <
process:ValueOf>

        <process:theVar rdf:resource="#AtomicSearchResult" />

        <process:fromProcess rdf:resource="&process;#SearchRepositoryPerform" />
                                                                                           </
process:ValueOf>
                                                                                   </
process:valueSource>
                                                                           </
process:InputBinding>
                                                                   </process:hasDataFrom>
                                                           </process:Perform>
```

```
                                                    </list:first>
                                                    <list:rest>
                                                            <process:ControlConstructList>
                                                                    <list:first>
                                                                            <process:Perform
rdf:ID="FindLeadAuthorPerform">
                                                                                    <
process:process rdf:resource="#FindLeadAuthor" />
                                                                                    <
process:hasDataFrom>
                                                                                            <
process:InputBinding>

        <process:toParam rdf:resource="#PaperForAuthor" />

        <process:valueSource>

                <process:ValueOf>

                        <process:theVar rdf:resource="#AtomicPaperChoice" />

                        <process:fromProcess rdf:resource="&process;#ChoosePaperPerform" />

                </process:ValueOf>

        </process:valueSource>
                                                                                            </
process:InputBinding>
                                                                                    </
process:hasDataFrom>
                                                                            </process:Perform>
                                                                    </list:first>
                                                                    <list:rest rdf:resource="&
list;#nil"/>
                                                            </process:ControlConstructList>
                                                    </list:rest>
                                            </process:ControlConstructList>
                                    </list:rest>
                            </process:ControlConstructList>
                    </process:components>
            </process:sequence>
    </process:composedOf>

    <process:hasResult>
  <process:Result>
    <process:inCondition rdf:resource="&expr;#AlwaysTrue"/>
    <process:withOutput>
      <process:OutputBinding>
        <process:toParam rdf:resource="#ChosenAuthor"/>
        <process:valueSource>
          <process:ValueOf>
            <process:theVar rdf:resource="#LeadAuthor"/>
            <process:fromProcess rdf:resource="#FindLeadAuthor"/>
          </process:ValueOf>
        </process:valueSource>
      </process:OutputBinding>
    </process:withOutput>
  </process:Result>
</process:hasResult>
```

```
</process:CompositeProcess>



<!--
=====================================================
      CompositeProcess: SignInAlternatives
=====================================================
-->

<process:CompositeProcess rdf:ID="SignInAlternatives">
      <rdfs:comment>
              TODO
      </rdfs:comment>

      <process:hasInput>
      <process:Input rdf:ID="SignInAlternativesCreateAcctInfo">
              <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#AcctInfo</
   process:parameterType>
      </process:Input>
      </process:hasInput>

      <process:hasInput>
      <process:Input rdf:ID="SignInAlternativesSignInData">
              <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#SignInData</
   process:parameterType>
      </process:Input>
      </process:hasInput>

      <process:hasOutput>
      <process:Output rdf:ID="SignInAlternativesAcctID">
              <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#AcctID</
   process:parameterType>
      </process:Output>
      </process:hasOutput>

      <process:composedOf>
      <process:Choice>
              <process:components>
                      <process:ControlConstructBag>
                              <list:first>
                              <process:Perform rdf:ID="CreateAcctSequencePerform">
                                      <process:process rdf:resource="#CreateAcctSequence"/>
                                      <process:hasDataFrom>
                                              <process:InputBinding>
                                                      <process:toParam rdf:resource="#
   CreateAcctSequenceCreateAcctInfo"/>
                                                      <process:valueSource>
                                                      <process:ValueOf>
                                                              <process:theVar rdf:resource="#
   SignInAlternativesCreateAcctInfo"/>
                                                              <process:fromProcess rdf:resource=
   "&process;#TheParentPerform"/>
                                                      </process:ValueOf>
                                                      </process:valueSource>
                                              </process:InputBinding>
                                      </process:hasDataFrom>
                              </process:Perform>
```

```
                        </list:first>
                        <list:rest>
                        <process:ControlConstructBag>
                                <list:first>
                                        <process:Perform rdf:ID="SignInSequencePerform">
                                                <process:process rdf:resource="#
SignInSequence"/>
                                                <process:hasDataFrom>
                                                <process:InputBinding>
                                                        <process:toParam rdf:resource="#
SignInSequenceSignInInfo"/>
                                                        <process:valueSource>
                                                                <process:ValueOf>
                                                                        <process:theVar
rdf:resource="#SignInAlternativesSignInData"/>
                                                                        <
process:fromProcess rdf:resource="&process;#TheParentPerform"/>
                                                                </process:ValueOf>
                                                        </process:valueSource>
                                                </process:InputBinding>
                                                </process:hasDataFrom>
                                        </process:Perform>
                                </list:first>
                                <list:rest rdf:resource="&list;#nil"/>
                        </process:ControlConstructBag>
                        </list:rest>
                </process:ControlConstructBag>
           </process:components>
       </process:Choice>
   </process:composedOf>

   <process:hasResult>
   <process:Result>
           <process:inCondition>
                   <expr:SWRL-Condition rdf:ID="SignInAlternativesAcctExists">
                           <rdfs:label>
                                   hasAcctID(SignInAlternativesSignInData,
SignInAlternativesAcctID)
                           </rdfs:label>
                           <rdfs:comment>
                           If an account already exists, sign-in operation will be
                           performed and returned acct ID willbe used
                           </rdfs:comment>
                           <expr:expressionLanguage rdf:resource="&expr;#SWRL"/>
                           <expr:expressionObject>
                           <swrl:AtomList>
                                   <rdf:first>
                                           <swrl:IndividualPropertyAtom>
                                                   <swrl:propertyPredicate rdf:resource="#
hasAcctID"/>
                                                   <swrl:argument1 rdf:resource="#
SignInAlternativesSignInData"/>
                                                   <swrl:argument2 rdf:resource="#
SignInAlternativesAcctID"/>
                                           </swrl:IndividualPropertyAtom>
                                   </rdf:first>
                                   <rdf:rest rdf:resource="&rdf;#nil"/>
                           </swrl:AtomList>
                           </expr:expressionObject>
```

```
                </expr:SWRL-Condition>
            </process:inCondition>
            <process:withOutput>
                <process:OutputBinding>
                    <process:toParam rdf:resource="#SignInAlternativesAcctID"/>
                    <process:valueSource>
                    <process:ValueOf>
                        <process:theVar rdf:resource="#SignInSequenceAcctID"/>
                        <process:fromProcess rdf:resource="#SignInSequencePerform"
/>
                    </process:ValueOf>
                    </process:valueSource>
                </process:OutputBinding>
            </process:withOutput>
      </process:Result>
      </process:hasResult>

      <process:hasResult>
      <process:Result>
            <process:inCondition>
                <expr:SWRL-Condition rdf:ID="SignInAlternativesNoAcctExists">
                    <rdfs:comment>
                    If an account does not exist a new account will be
                    create by the CreateAcct process and the ID of new
                    account will be used.
                    </rdfs:comment>
                    <expr:expressionLanguage rdf:resource="&expr;#SWRL"/>
                    <expr:expressionObject>
                    <swrl:AtomList>
                        <rdf:first>
                            <swrl:ClassAtom>
                                <swrl:classPredicate rdf:resource="#
NonExistingAcct"/>
                                <swrl:argument1 rdf:resource="#
SignInAlternativesSignInData"/>
                            </swrl:ClassAtom>
                        </rdf:first>
                        <rdf:rest rdf:resource="&rdf;#nil"/>
                    </swrl:AtomList>
                    </expr:expressionObject>
                </expr:SWRL-Condition>
            </process:inCondition>
            <process:withOutput>
                <process:OutputBinding>
                    <process:toParam rdf:resource="#SignInAlternativesAcctID"/>
                    <process:valueSource>
                    <process:ValueOf>
                        <process:theVar rdf:resource="#
CreateAcctSequenceCreateAcctOutput"/>
                        <process:fromProcess rdf:resource="#
CreateAcctSequencePerform"/>
                    </process:ValueOf>
                    </process:valueSource>
                </process:OutputBinding>
            </process:withOutput>
      </process:Result>
      </process:hasResult>
</process:CompositeProcess>
```

```
<!--
=====================================================
      CompositeProcess: SignInSequence
=====================================================
-->

<process:CompositeProcess rdf:ID="SignInSequence">
      <rdfs:comment>
      SignInSequence performs an atomic process SignIn,
      followed by an atomic process LoadUserProfile.
      </rdfs:comment>

      <process:hasInput>
      <process:Input rdf:ID="SignInSequenceSignInInfo">
              <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#SignInData</
  process:parameterType>
      </process:Input>
      </process:hasInput>

      <process:hasOutput>
      <process:Output rdf:ID="SignInSequenceAcctID">
              <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#AcctID</
  process:parameterType>
      </process:Output>
      </process:hasOutput>

      <process:composedOf>
      <process:Sequence>
              <process:components>
                      <process:ControlConstructList>
                              <list:first>
                              <process:Perform rdf:ID="SignInPerform">
                                      <process:process rdf:resource="#SignIn"/>
                                      <process:hasDataFrom>
                                              <process:InputBinding>
                                                      <process:toParam rdf:resource="#SignInInfo
  "/>
                                                      <process:valueSource>
                                                      <process:ValueOf>
                                                              <process:theVar rdf:resource="#
  SignInSequenceSignInInfo"/>
                                                              <process:fromProcess rdf:resource=
  "&process;#TheParentPerform"/>
                                                      </process:ValueOf>
                                                      </process:valueSource>
                                              </process:InputBinding>
                                      </process:hasDataFrom>
                              </process:Perform>
                              </list:first>
                              <list:rest>
                              <process:ControlConstructList>
                                      <list:first>
                                              <process:Perform rdf:ID="LoadUserProfilePerform">
                                                      <process:process rdf:resource="#
  LoadUserProfile"/>
                                              </process:Perform>
                                      </list:first>
```

```
                                        <list:rest rdf:resource="&list;#nil"/>
                                </process:ControlConstructList>
                                </list:rest>
                        </process:ControlConstructList>
                </process:components>
        </process:Sequence>
        </process:composedOf>
</process:CompositeProcess>


<!--
=======================================================
        CompositeProcess: CreateAcctSequence
=======================================================
-->

<process:CompositeProcess rdf:ID="CreateAcctSequence">
        <rdfs:comment>
        CreateAcctSequence performs atomic process
        CreateAcct followed by atomic process LoadUserProfile.
        </rdfs:comment>

        <process:hasInput>
        <process:Input rdf:ID="CreateAcctSequenceCreateAcctInfo">
                <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#AcctInfo</
    process:parameterType>
        </process:Input>
        </process:hasInput>

        <process:hasOutput>
        <process:Output rdf:ID="CreateAcctSequenceCreateAcctOutput">
                <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#AcctID</
    process:parameterType>
        </process:Output>
        </process:hasOutput>

        <process:composedOf>
        <process:Sequence>
                <process:components>
                        <process:ControlConstructList>
                                <list:first>
                                <process:Perform rdf:ID="CreateAcctPerform">
                                        <process:process rdf:resource="#CreateAcct"/>
                                        <process:hasDataFrom>
                                                <process:InputBinding>
                                                        <process:toParam rdf:resource="#
    CreateAcctInfo"/>
                                                        <process:valueSource>
                                                        <process:ValueOf>
                                                                <process:theVar rdf:resource="#
    CreateAcctSequenceCreateAcctInfo"/>
                                                                <process:fromProcess rdf:resource=
    "&process;#TheParentPerform"/>
                                                        </process:ValueOf>
                                                        </process:valueSource>
                                                </process:InputBinding>
                                        </process:hasDataFrom>
                                </process:Perform>
                                </list:first>
```

```
                                <list:rest>
                                <process:ControlConstructList>
                                        <list:first>
                                                <process:Perform rdf:ID="LoadUserProfilePerform1">
                                                        <process:process rdf:resource="#
   LoadUserProfile"/>
                                                </process:Perform>
                                        </list:first>
                                        <list:rest rdf:resource="&list;#nil"/>
                                </process:ControlConstructList>
                                </list:rest>
                        </process:ControlConstructList>
                </process:components>
        </process:Sequence>
        </process:composedOf>
        <process:hasResult>
        <process:Result>
                <process:inCondition rdf:resource="&expr;#AlwaysTrue"/>
                <process:withOutput>
                        <process:OutputBinding>
                                <process:toParam rdf:resource="#CreateAcctSequenceCreateAcctOutput
   "/>
                                <process:valueSource>
                                <process:ValueOf>
                                        <process:theVar rdf:resource="#CreateAcctOutput"/>
                                        <process:fromProcess rdf:resource="#CreateAcctPerform"/>
                                </process:ValueOf>
                                </process:valueSource>
                        </process:OutputBinding>
                </process:withOutput>
        </process:Result>
        </process:hasResult>
</process:CompositeProcess>




<!--
=======================================================
      AtomicProcess: SearchRepository
=======================================================
-->

<process:AtomicProcess rdf:ID="SearchRepository">
      <process:hasInput>
              <process:Input rdf:ID="AtomicSearchTerm">
                      <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#SearchTerm</
   process:parameterType>
              </process:Input>
      </process:hasInput>

      <process:hasOutput>
              <process:Output rdf:ID="AtomicSearchResult">
                      <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#
   AcademicPaperList</process:parameterType>
              </process:Output>
```

```
      </process:hasOutput>
</process:AtomicProcess>


<!--
=======================================================
      AtomicProcess: ChoosePaper
=======================================================
-->

<process:AtomicProcess rdf:ID="ChoosePaper">
      <process:hasInput>
              <process:Input rdf:ID="AtomicPaperList">
                      <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#
   AcademicPaperList</process:parameterType>
              </process:Input>
      </process:hasInput>

      <process:hasOutput>
              <process:Output rdf:ID="AtomicPaperChoice">
                      <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#
   AcademicPaper</process:parameterType>
              </process:Output>
      </process:hasOutput>
</process:AtomicProcess>



<!--
=======================================================
      AtomicProcess: FindLeadAuthor
=======================================================
-->

<process:AtomicProcess rdf:ID="FindLeadAuthor">
      <process:hasInput>
              <process:Input rdf:ID="PaperForAuthor">
                      <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#
   AcademicPaper</process:parameterType>
              </process:Input>
      </process:hasInput>

      <process:hasOutput>
              <process:Output rdf:ID="LeadAuthor">
                      <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#Author</
   process:parameterType>
              </process:Output>
      </process:hasOutput>
</process:AtomicProcess>



<!--
=======================================================
      AtomicProcess: SignIn
=======================================================
-->

<process:AtomicProcess rdf:ID="SignIn">
```

```
      <rdfs:comment>
      Sign in is a process that requires input of signin info.
      </rdfs:comment>

      <process:hasInput>
      <process:Input rdf:ID="SignInInfo">
              <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#SignInData</
   process:parameterType>
      </process:Input>
      </process:hasInput>
</process:AtomicProcess>


<!--
=======================================================
      AtomicProcess: CreateAcct
=======================================================
-->

<process:AtomicProcess rdf:ID="CreateAcct">
      <process:hasInput>
      <process:Input rdf:ID="CreateAcctInfo">
              <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#AcctInfo</
   process:parameterType>
      </process:Input>
      </process:hasInput>

      <process:hasOutput>
      <process:Output rdf:ID="CreateAcctOutput">
              <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#AcctID</
   process:parameterType>
      </process:Output>
      </process:hasOutput>
</process:AtomicProcess>


<!--
=======================================================
      AtomicProcess: LoadUserProfile
=======================================================
-->

<process:AtomicProcess rdf:ID="LoadUserProfile">
      <rdfs:comment>
      LoadUserProfile can only be invoked if a user profile already exists.
      </rdfs:comment>
</process:AtomicProcess>


<!--
=======================================================
      AtomicProcess: GetUserInfo
=======================================================
-->

<process:AtomicProcess rdf:ID="GetUserInfo">
      <process:hasInput>
      <process:Input rdf:ID="AuthorUserName">
```

```
                <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#Author</
    process:parameterType>
        </process:Input>
        </process:hasInput>

        <process:hasOutput>
        <process:Output rdf:ID="ExpertUserID">
                <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#AcctID</
    process:parameterType>
        </process:Output>
        </process:hasOutput>
  </process:AtomicProcess>


  <!--
  ======================================================
        AtomicProcess: SendMessage
  ======================================================
  -->

  <process:AtomicProcess rdf:ID="SendMessage">
        <process:hasInput>
        <process:Input rdf:ID="SendMessageMessage">
                <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#Message</
    process:parameterType>
        </process:Input>
        </process:hasInput>
        <process:hasInput>
        <process:Input rdf:ID="SendMessageUser">
                <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#AcctID</
    process:parameterType>
        </process:Input>
        </process:hasInput>

        <process:hasOutput>
        <process:Output rdf:ID="SendMessageConfirm">
                <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#MsgSendConfirmation<
    /process:parameterType>
        </process:Output>
        </process:hasOutput>
  </process:AtomicProcess>

</rdf:RDF>
```

A.5.2 FilmLocator.owls

```
<?xml version="1.0" encoding="WINDOWS-1252"?>
<!DOCTYPE uridef [
  <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema">
  <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns">
  <!ENTITY owl "http://www.w3.org/2002/07/owl">
  <!ENTITY xsd "http://www.w3.org/2001/XMLSchema">
  <!ENTITY service "http://www.daml.org/services/owl-s/1.2/Service.owl">
  <!ENTITY profile "http://www.daml.org/services/owl-s/1.2/Profile.owl">
  <!ENTITY process "http://www.daml.org/services/owl-s/1.2/Process.owl">
  <!ENTITY grounding "http://www.daml.org/services/owl-s/1.2/Grounding.owl">
  <!ENTITY list "http://www.daml.org/services/owl-s/1.2/generic/ObjectList.owl">
```

```
  <!ENTITY swrl "http://www.w3.org/2003/11/swrl">
  <!ENTITY swrlb "http://www.w3.org/2003/11/swrlb">
  <!ENTITY expr "http://www.daml.org/services/owl-s/1.2/generic/Expression.owl">
  <!ENTITY mo "http://testcases.freya.local/ontology/movieontology.owl">
  <!ENTITY concepts "http://testcases.freya.local/ontology/concepts.owl">
  <!ENTITY THIS "http://testcases.freya.local/info/filmLocator.owls">
]>

<rdf:RDF
        xmlns:rdf="&rdf;#"
        xmlns:xsd="&xsd;#"
        xmlns:rdfs="&rdfs;#"
        xmlns:owl="&owl;#"
        xmlns:expr="&expr;#"
        xmlns:service="&service;#"
        xmlns:grounding="&grounding;#"
    xmlns:process="&process;#"
        xmlns:profile="&profile;#"
    xmlns:list="&list;#"
    xmlns:swrl="&swrl;#"
    xmlns:swrlb="&swrlb;#"
        xmlns:mo="&mo;#"
        xmlns:concepts="&concepts;#"
    xmlns="&THIS;#"
        xml:base="&THIS;#">

  <owl:Ontology rdf:about="">
        <owl:versionInfo>
                $ID: filmLocator.owls, v 1.4 2011/02/28 akd07r $
        </owl:versionInfo>
        <rdfs:comment>
                TODO
        </rdfs:comment>
        <owl:imports rdf:resource="&service;" />
        <owl:imports rdf:resource="&process;" />
        <owl:imports rdf:resource="&profile;" />
        <owl:imports rdf:resource="&grounding;" />
        <owl:imports rdf:resource="&concepts;" />
        <owl:imports rdf:resource="&mo;" />
  </owl:Ontology>

  <!--
  ================================================
  ================================================
        Service Model
  ================================================
  ================================================
  -->
  <service:Service rdf:ID="FilmLocatorService">
        <service:presents rdf:resource="#FilmLocatorProfile"/>
        <service:describedBy rdf:resource="#FilmLocator"/>
        <!--<service:supports rdf:resource="#FilmLocatorServiceGrounding"/>-->
  </service:Service>

  <!--
  ================================================
  ================================================
        Profile Model
  ================================================
```

```
=====================================================
-->
<profile:Profile rdf:ID="FilmLocatorProfile">
      <profile:serviceName>
              FilmLocator_Service
      </profile:serviceName>
      <profile:textDescription>
      TODO
      </profile:textDescription>

      <service:presentedBy rdf:resource="#FilmLocatorService"/>
      <profile:has_process rdf:resource="#FilmLocator"/>

      <profile:hasInput rdf:resource="#FilmLocatorGenre"/>
      <profile:hasInput rdf:resource="#FilmLocatorLocation"/>
      <profile:hasInput rdf:resource="#FilmLocatorCinema"/>
      <profile:hasInput rdf:resource="#FilmLocatorFilm"/>

      <profile:hasOutput rdf:resource="#FilmLocatorReview"/>
      <profile:hasOutput rdf:resource="#FilmLocatorInfo"/>
      <profile:hasOutput rdf:resource="#FilmLocatorMap"/>
</profile:Profile>

<!--
=====================================================
=====================================================
      Process Model
=====================================================
=====================================================
-->

<!--
=====================================================
      CompositeProcess: FilmLocator
=====================================================
-->

<process:CompositeProcess rdf:ID="FilmLocator">
      <rdfs:comment>
              TODO
      </rdfs:comment>

      <process:hasInput>
              <process:Input rdf:ID="FilmLocatorGenre">
                      <process:parameterType rdf:datatype="&xsd;#anyURI">&mo;#Genre</
   process:parameterType>
              </process:Input>
      </process:hasInput>

      <process:hasInput>
              <process:Input rdf:ID="FilmLocatorFilm">
                      <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#Film</
   process:parameterType>
              </process:Input>
      </process:hasInput>

      <process:hasInput>
              <process:Input rdf:ID="FilmLocatorLocation">
```

```
                <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#Location</
process:parameterType>
            </process:Input>
    </process:hasInput>

    <process:hasInput>
            <process:Input rdf:ID="FilmLocatorCinema">
                <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#Cinema</
process:parameterType>
            </process:Input>
    </process:hasInput>


    <process:hasOutput>
            <process:Output rdf:ID="FilmLocatorReview">
                <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#FilmReview</
process:parameterType>
            </process:Output>
    </process:hasOutput>

    <process:hasOutput>
            <process:Output rdf:ID="FilmLocatorInfo">
                <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#FilmInfo</
process:parameterType>
            </process:Output>
    </process:hasOutput>

    <process:hasOutput>
            <process:Output rdf:ID="FilmLocatorMap">
                <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#Map</
process:parameterType>
            </process:Output>
    </process:hasOutput>

    <process:composedOf>
            <process:sequence>
                <process:components>
                    <process:ControlConstructList>
                        <list:first>
                            <process:Perform rdf:ID="FilmFinderPerform">
                                <process:process rdf:resource="#FilmFinder
" />
                                <process:hasDataFrom>
                                    <process:InputBinding>
                                        <process:toParam
rdf:resource="#FilmFinderGenre" />
                                        <process:valueSource>
                                            <process:ValueOf>
                                                <
process:theVar rdf:resource="#FilmLocatorGenre" />
                                                <
process:fromProcess rdf:resource="&process;#TheParentPerform" />
                                            </process:ValueOf>
                                        </process:valueSource>
                                    </process:InputBinding>
                                </process:hasDataFrom>
                            </process:Perform>
                        </list:first>
                        <list:rest>
```

```
                                        <process:ControlConstructList>
                                <list:first>
                                        <process:Perform rdf:ID="FindLocalCinemaPerform">
                                        <process:process rdf:resource="#FindLocalCinema"/>
                                                <process:hasDataFrom>
                                                <process:InputBinding>
                                                        <process:toParam rdf:resource="#
FindLocalCinemaLocation"/>
                                                        <process:valueSource>
                                                                <process:ValueOf>
                                                                        <process:theVar
rdf:resource="#FilmLocatorLocation"/>
                                                                        <
process:fromProcess rdf:resource="&process;#TheParentPerform"/>
                                                                </process:ValueOf>
                                                                </process:valueSource>
                                                        </process:InputBinding>
                                                        </process:hasDataFrom>
                                        </process:Perform>
                                </list:first>
                                                <list:rest>
                                                        <process:ControlConstructList>
                                                                <list:first>
                                                                        <process:Perform
rdf:ID="GetFilmInformationPerform">
                                                                                <
process:process rdf:resource="#GetFilmInformation" />
                                                                                <
process:hasDataFrom>
                                                                                        <
process:InputBinding>

        <process:toParam rdf:resource="#GetFilmInformationCinema" />

        <process:valueSource>

                <process:ValueOf>

                        <process:theVar rdf:resource="#FindLocalCinemaCinema" />

                        <process:fromProcess rdf:resource="&THIS;#FindLocalCinemaPerform" />

                </process:ValueOf>

        </process:valueSource>
                                                                                        </
process:InputBinding>
                                                                                </
process:hasDataFrom>
                                                                                <
process:hasDataFrom>
                                                                                        <
process:InputBinding>

        <process:toParam rdf:resource="#GetFilmInformationCinemaFilm" />

        <process:valueSource>

                <process:ValueOf>
```

```
                        <process:theVar rdf:resource="#FilmLocatorFilm" />

                        <process:fromProcess rdf:resource="&process;#TheParentPerform" />

                </process:ValueOf>

        </process:valueSource>
                                                                                </
process:InputBinding>
                                                                        </
process:hasDataFrom>
                                                                        </process:Perform>
                                                                </list:first>
                                                                <list:rest>
                                                                        <process:Perform
rdf:ID="GetCinemaInfoPerform">
                                                                        <
process:process rdf:resource="#GetCinemaInfo" />
                                                                        <
process:hasDataFrom>
                                                                                <
process:InputBinding>

        <process:toParam rdf:resource="#GetCinemaInfoCinema" />

        <process:valueSource>

                <process:ValueOf>

                        <process:theVar rdf:resource="#FilmLocatorCinema" />

                        <process:fromProcess rdf:resource="&process;#TheParentPerform" />

                </process:ValueOf>

        </process:valueSource>
                                                                                </
process:InputBinding>
                                                                        </
process:hasDataFrom>
                                                                        </process:Perform>
                                                                </list:rest>
                                                        </process:ControlConstructList>
                                                </list:rest>
                                        </process:ControlConstructList>
                                </list:rest>
                        </process:ControlConstructList>
                </process:components>
        </process:sequence>
   </process:composedOf>

   <process:hasResult>
        <process:Result rdf:ID="FilmLocatorFilmReviewResult">
        <process:inCondition rdf:resource="&expr;#AlwaysTrue"/>
        <process:withOutput>
                <process:OutputBinding>
                        <process:toParam rdf:resource="#FilmLocatorReview"/>
                        <process:valueSource>
```

```
                                <process:ValueOf>
                                        <process:theVar rdf:resource="#FilmReviewForGenre"/>
                                        <process:fromProcess rdf:resource="#FilmFinderPerform"/>
                                </process:ValueOf>
                                </process:valueSource>
                        </process:OutputBinding>
                </process:withOutput>
            </process:Result>
        </process:hasResult>
        <process:hasResult>
                <process:Result rdf:ID="FilmLocatorFilmInfoResult">
                <process:inCondition rdf:resource="&expr;#AlwaysTrue"/>
                <process:withOutput>
                        <process:OutputBinding>
                                <process:toParam rdf:resource="#FilmLocatorInfOutput"/>
                                <process:valueSource>
                                <process:ValueOf>
                                        <process:theVar rdf:resource="#GetFilmInformationFull"/>
                                        <process:fromProcess rdf:resource="#
    GetFilmInformationPerform"/>
                                </process:ValueOf>
                                </process:valueSource>
                        </process:OutputBinding>
                </process:withOutput>
            </process:Result>
        </process:hasResult>
        <process:hasResult>
                <process:Result rdf:ID="FilmLocatorMapResult">
                <process:inCondition rdf:resource="&expr;#AlwaysTrue"/>
                <process:withOutput>
                        <process:OutputBinding>
                                <process:toParam rdf:resource="#FilmLocatorMap"/>
                                <process:valueSource>
                                <process:ValueOf>
                                        <process:theVar rdf:resource="#GetFilmInformationFull"/>
                                        <process:fromProcess rdf:resource="#GetCinemaInfoPerform"/
    >
                                </process:ValueOf>
                                </process:valueSource>
                        </process:OutputBinding>
                </process:withOutput>
            </process:Result>
        </process:hasResult>

</process:CompositeProcess>


<!--
=======================================================
      CompositeProcess: FilmFinder
=======================================================
-->

<process:CompositeProcess rdf:ID="FilmFinder">
      <rdfs:comment>
      </rdfs:comment>

      <process:hasInput>
              <process:Input rdf:ID="FilmFinderGenre">
```

```
                <process:parameterType rdf:datatype="&xsd;#anyURI">&mo;#Genre</
process:parameterType>
            </process:Input>
    </process:hasInput>

    <process:hasOutput>
            <process:Output rdf:ID="FilmReviewForGenre">
                <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#FilmReview</
process:parameterType>
            </process:Output>
    </process:hasOutput>

    <process:composedOf>
            <process:sequence>
                <process:components>
                    <process:ControlConstructList>
                        <list:first>
                            <process:Perform rdf:ID="RandomFilmForGenrePerform
">
                                <process:process rdf:resource="
RandomFilmForGenre" />
                                <process:hasDataFrom>
                                    <process:InputBinding>
                                        <process:toParam
rdf:resource="#RandomFilmForGenreGenre" />
                                        <process:valueSource>
                                            <process:ValueOf>
                                                <
process:theVar rdf:resource="#FilmFinderGenre" />
                                                <
process:fromProcess rdf:resource="&process;#TheParentPerform" />
                                            </process:ValueOf>
                                        </process:valueSource>
                                    </process:InputBinding>
                                </process:hasDataFrom>
                            </process:Perform>
                        </list:first>
                        <list:rest>
                            <process:Perform rdf:ID="GetFilmReviewPerform">
                                <process:process rdf:resource="#
GetFilmReview" />
                                <process:hasDataFrom>
                                    <process:InputBinding>
                                        <process:toParam
rdf:resource="#GetFilmReviewFilm" />
                                        <process:valueSource>
                                            <process:ValueOf>
                                                <
process:theVar rdf:resource="#RandomFilmForGenreFilm" />
                                                <
process:fromProcess rdf:resource="&process;#RandomFilmForGenrePerform" />
                                            </process:ValueOf>
                                        </process:valueSource>
                                    </process:InputBinding>
                                </process:hasDataFrom>
                            </process:Perform>
                        </list:rest>
                    </process:ControlConstructList>
                </process:components>
```

```
              </process:sequence>
      </process:composedOf>

      <process:hasResult>
    <process:Result>
      <process:inCondition rdf:resource="&expr;#AlwaysTrue"/>
      <process:withOutput>
        <process:OutputBinding>
          <process:toParam rdf:resource="#FilmReviewForGenre"/>
          <process:valueSource>
            <process:ValueOf>
              <process:theVar rdf:resource="#GetFilmReviewReview"/>
              <process:fromProcess rdf:resource="#GetFilmReview"/>
            </process:ValueOf>
          </process:valueSource>
        </process:OutputBinding>
      </process:withOutput>
    </process:Result>
  </process:hasResult>

</process:CompositeProcess>

<!--
=======================================================
      AtomicProcess: RandomFilmForGenre
=======================================================
-->

<process:AtomicProcess rdf:ID="RandomFilmForGenre">
      <process:hasInput>
      <process:Input rdf:ID="RandomFilmForGenreGenre">
              <process:parameterType rdf:datatype="&xsd;#anyURI">&mo;#Genre</
   process:parameterType>
      </process:Input>
      </process:hasInput>

      <process:hasOutput>
      <process:Output rdf:ID="RandomFilmForGenreFilm">
              <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#Film</
   process:parameterType>
      </process:Output>
      </process:hasOutput>
</process:AtomicProcess>

<!--
=======================================================
      AtomicProcess: GetFilmReview
=======================================================
-->

<process:AtomicProcess rdf:ID="GetFilmReview">
      <process:hasInput>
      <process:Input rdf:ID="GetFilmReviewFilm">
              <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#Film</
   process:parameterType>
      </process:Input>
      </process:hasInput>

      <process:hasOutput>
```

```
      <process:Output rdf:ID="GetFilmReviewReview">
              <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#FilmReview</
   process:parameterType>
      </process:Output>
      </process:hasOutput>
</process:AtomicProcess>

<!--
=======================================================
      AtomicProcess: FindLocalCinema
=======================================================
-->

<process:AtomicProcess rdf:ID="FindLocalCinema">
      <process:hasInput>
      <process:Input rdf:ID="FindLocalCinemaLocation">
              <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#Location</
   process:parameterType>
      </process:Input>
      </process:hasInput>

      <process:hasOutput>
      <process:Output rdf:ID="FindLocalCinemaCinema">
              <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#Cinema</
   process:parameterType>
      </process:Output>
      </process:hasOutput>
</process:AtomicProcess>

<!--
=======================================================
      AtomicProcess: GetFilmInformation
=======================================================
-->

<process:AtomicProcess rdf:ID="GetFilmInformation">
      <process:hasInput>
      <process:Input rdf:ID="GetFilmInformationCinema">
              <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#Cinema</
   process:parameterType>
      </process:Input>
      </process:hasInput>

      <process:hasInput>
      <process:Input rdf:ID="GetFilmInformationFilm">
              <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#Film</
   process:parameterType>
      </process:Input>
      </process:hasInput>

      <process:hasOutput>
      <process:Output rdf:ID="GetFilmInformationFull">
              <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#FilmInfo</
   process:parameterType>
      </process:Output>
      </process:hasOutput>
</process:AtomicProcess>
```

```
  <!--
  ======================================================
        AtomicProcess: GetCinemaInfo
  ======================================================
  -->

  <process:AtomicProcess rdf:ID="GetCinemaInfo">
        <process:hasInput>
        <process:Input rdf:ID="GetCinemaInfoCinema">
                <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#Cinema</
     process:parameterType>
        </process:Input>
        </process:hasInput>

        <process:hasOutput>
        <process:Output rdf:ID="GetCinemaInfoMap">
                <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#Map</
     process:parameterType>
                </process:Output>
        </process:hasOutput>
  </process:AtomicProcess>

</rdf:RDF>
```

A.5.3 BookMedicalAppointment.owls

```
<?xml version="1.0" encoding="WINDOWS-1252"?>
<!DOCTYPE uridef [
  <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema">
  <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns">
  <!ENTITY owl "http://www.w3.org/2002/07/owl">
  <!ENTITY xsd "http://www.w3.org/2001/XMLSchema">
  <!ENTITY service "http://www.daml.org/services/owl-s/1.2/Service.owl">
  <!ENTITY profile "http://www.daml.org/services/owl-s/1.2/Profile.owl">
  <!ENTITY process "http://www.daml.org/services/owl-s/1.2/Process.owl">
  <!ENTITY grounding "http://www.daml.org/services/owl-s/1.2/Grounding.owl">
  <!ENTITY list "http://www.daml.org/services/owl-s/1.2/generic/ObjectList.owl">
  <!ENTITY swrl "http://www.w3.org/2003/11/swrl">
  <!ENTITY swrlb "http://www.w3.org/2003/11/swrlb">
  <!ENTITY expr "http://www.daml.org/services/owl-s/1.2/generic/Expression.owl">
  <!ENTITY foaf "http://xmlns.com/foaf/spec/">
  <!ENTITY concepts "http://testcases.freya.local/ontology/concepts.owl">
  <!ENTITY THIS "http://testcases.freya.local/service/bookMedicalAppointment.owls">
]>

<rdf:RDF
        xmlns:rdf="&rdf;#"
        xmlns:xsd="&xsd;#"
        xmlns:rdfs="&rdfs;#"
        xmlns:owl="&owl;#"
        xmlns:expr="&expr;#"
        xmlns:service="&service;#"
        xmlns:grounding="&grounding;#"
    xmlns:process="&process;#"
        xmlns:profile="&profile;#"
    xmlns:list="&list;#"
    xmlns:swrl="&swrl;#"
```

```
  xmlns:swrlb="&swrlb;#"
      xmlns:foaf="&foaf;#"
      xmlns:concepts="&concepts;#"
  xmlns="&THIS;#"
      xml:base="&THIS;#">

<owl:Ontology rdf:about="">
      <owl:versionInfo>
              $ID: bookMedicalAppointment.owls, v 1.4 2011/02/28 akd07r $
      </owl:versionInfo>
      <rdfs:comment>
              TODO
      </rdfs:comment>
      <owl:imports rdf:resource="&service;" />
      <owl:imports rdf:resource="&process;" />
      <owl:imports rdf:resource="&profile;" />
      <owl:imports rdf:resource="&grounding;" />
      <owl:imports rdf:resource="&concepts;" />
      <owl:imports rdf:resource="http://127.0.0.1/ontology/portal.owl" />
      <owl:imports rdf:resource="http://127.0.0.1/ontology/support.owl" />
</owl:Ontology>

<!--
=======================================================
=======================================================
      Service Model
=======================================================
=======================================================
-->
<service:Service rdf:ID="BookMedicalAppointmentService">
      <service:presents rdf:resource="#BookMedicalAppointmentProfile"/>
      <service:describedBy rdf:resource="#BookMedicalAppointment"/>
      <!--<service:supports rdf:resource="#BookMedicalAppointmentServiceGrounding"/>-->
</service:Service>

<!--
=======================================================
=======================================================
      Profile Model
=======================================================
=======================================================
-->
<profile:Profile rdf:ID="BookMedicalAppointmentProfile">
      <profile:serviceName>
              BookMedicalAppointment_Service
      </profile:serviceName>
      <profile:textDescription>
      TODO
      </profile:textDescription>

      <service:presentedBy rdf:resource="#BookMedicalAppointmentService"/>
      <profile:has_process rdf:resource="#BookMedicalAppointment"/>

      <profile:hasInput rdf:resource="#BookMedicalAppointmentLocation"/>
      <profile:hasInput rdf:resource="#BookMedicalAppointmentSearchTerm"/>
      <profile:hasInput rdf:resource="#BookMedicalAppointmentTime"/>

      <profile:hasOutput rdf:resource="#BookMedicalAppointmentTimes"/>
      <profile:hasOutput rdf:resource="#BookMedicalAppointmentAppointment"/>
```

```
      <profile:hasOutput rdf:resource="#BookMedicalAppointmentAppointmentGuide"/>
</profile:Profile>

<!--
=======================================================
=======================================================
      Process Model
=======================================================
=======================================================
-->

<!--
=======================================================
      CompositeProcess: BookMedicalAppointment
=======================================================
-->

<process:CompositeProcess rdf:ID="BookMedicalAppointment">
      <process:hasInput>
              <process:Input rdf:ID="BookMedicalAppointmentLocation">
              <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#Location</
   process:parameterType>
              </process:Input>
      </process:hasInput>

      <process:hasInput>
      <process:Input rdf:ID="BookMedicalAppointmentSearchTerm">
              <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#SearchTerm</
   process:parameterType>
      </process:Input>
      </process:hasInput>

      <process:hasInput>
      <process:Input rdf:ID="BookMedicalAppointmentTime">
              <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#AppointmentTime</
   process:parameterType>
      </process:Input>
      </process:hasInput>

      <process:hasOutput>
      <process:Output rdf:ID="BookMedicalAppointmentTimes">
              <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#AppointmentTimeList<
   /process:parameterType>
      </process:Output>
      </process:hasOutput>

      <process:hasOutput>
      <process:Output rdf:ID="BookMedicalAppointmentAppointment">
              <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#MedicalAppointment</
   process:parameterType>
      </process:Output>
      </process:hasOutput>

      <process:hasOutput>
      <process:Output rdf:ID="BookMedicalAppointmentAppointmentGuide">
              <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#AppointmentGuide</
   process:parameterType>
      </process:Output>
      </process:hasOutput>
```

```
    <process:composedOf>
    <process:Sequence>
            <process:components>
                    <process:ControlConstructList>
                            <list:first>
                            <process:Perform rdf:ID="LocalPCTPerform">
                                    <process:process rdf:resource="#LocalPCT"/>
                                    <process:hasDataFrom>
                                            <process:InputBinding>
                                                    <process:toParam rdf:resource="#
LocalPCTLocation"/>
                                                    <process:valueSource>
                                                    <process:ValueOf>
                                                            <process:theVar rdf:resource="#
BookMedicalAppointmentLocation"/>
                                                            <process:fromProcess rdf:resource=
"&process;#TheParentPerform"/>
                                                    </process:ValueOf>
                                                    </process:valueSource>
                                            </process:InputBinding>
                                    </process:hasDataFrom>
                            </process:Perform>
                            </list:first>
                            <list:rest>
                            <process:ControlConstructList>
                                            <list:first>
                                            <process:Perform rdf:ID="FindCareSequencePerform">
                                                    <process:process rdf:resource="#
FindCareSequence"/>
                                                    <process:hasDataFrom>
                                                            <process:InputBinding>
                                                                    <process:toParam
rdf:resource="#FindCareSequenceSearchTerm"/>
                                                                    <process:valueSource>
                                                                    <process:ValueOf>
                                                                            <process:theVar
rdf:resource="#BookMedicalAppointmentSearchTerm"/>
                                                                            <
process:fromProcess rdf:resource="&process;#TheParentPerform"/>
                                                                    </process:ValueOf>
                                                                    </process:valueSource>
                                                            </process:InputBinding>
                                                    </process:hasDataFrom>
                                                            <process:hasDataFrom>
                                                            <process:InputBinding>
                                                                    <process:toParam
rdf:resource="#FindCareSequencePCT"/>
                                                                    <process:valueSource>
                                                                    <process:ValueOf>
                                                                            <process:theVar
rdf:resource="#LocalPCTPCT"/>
                                                                            <
process:fromProcess rdf:resource="&THIS;#LocalPCTPerform"/>
                                                                    </process:ValueOf>
                                                                    </process:valueSource>
                                                            </process:InputBinding>
                                                    </process:hasDataFrom>
                                            </process:Perform>
```

```
                                        </list:first>
                                        <list:rest>
                                        <process:ControlConstructList>
                                                        <list:first>
                                                        <process:Perform rdf:ID="
RequestFreeSlotsPerform">
                                                                <process:process
rdf:resource="#RequestFreeSlots"/>
                                                                <process:hasDataFrom>
                                                                        <
process:InputBinding>
                                                                                <
process:toParam rdf:resource="#RequestFreeSlotsDepartment"/>
                                                                                <
process:valueSource>
                                                                                <
process:ValueOf>
                                                                                        <
process:theVar rdf:resource="#FindCareSequenceCareOrganization"/>
                                                                                        <
process:fromProcess rdf:resource="&THIS;#FindCareSequencePerform"/>
                                                                                </
process:ValueOf>
                                                                                </
process:valueSource>
                                                                        </
process:InputBinding>
                                                                </process:hasDataFrom>
                                                        </process:Perform>
                                                        </list:first>
                                                        <list:rest>
                                                        <process:ControlConstructList>
                                                                <list:first>
                                                                <process:Perform
rdf:ID="BookAppointmentPerform">
                                                                        <
process:process rdf:resource="#BookAppointment"/>
                                                                        <
process:hasDataFrom>
                                                                                <
process:InputBinding>

        <process:toParam rdf:resource="#BookAppointmentHospitalDepartment"/>

        <process:valueSource>

        <process:ValueOf>

                <process:theVar rdf:resource="#FindCareSequenceCareOrganization"/>

                <process:fromProcess rdf:resource="&THIS;#FindCareSequencePerform"/>

        </process:ValueOf>

        </process:valueSource>
                                                                                </
process:InputBinding>
                                                                        </
process:hasDataFrom>
```

```
        <process:hasDataFrom>
                                                                                <
process:InputBinding>
        <process:toParam rdf:resource="#BookAppointmentTime"/>
        <process:valueSource>
        <process:ValueOf>
                <process:theVar rdf:resource="#BookMedicalAppointmentTime"/>
                <process:fromProcess rdf:resource="&process;#TheParentPerform"/>
        </process:ValueOf>
        </process:valueSource>
                                                                                </
process:InputBinding>
                                                                        </
process:hasDataFrom>
                                                                    </process:Perform>
                                                                    </list:first>
                                                                    <list:rest>
                                                                    <
process:ControlConstructList>
                                                                                <
list:first>
                                                                                <
process:Perform rdf:ID="GetInstructionsPerform">
        <process:process rdf:resource="#GetInstructions"/>
        <process:hasDataFrom>
                <process:InputBinding>
                        <process:toParam rdf:resource="#GetInstructionsOrganization"/>
                        <process:valueSource>
                        <process:ValueOf>
                                <process:theVar rdf:resource="#
FindCareSequenceCareOrganization"/>
                                <process:fromProcess rdf:resource="&THIS;#
FindCareSequencePerform"/>
                        </process:ValueOf>
                        </process:valueSource>
                </process:InputBinding>
        </process:hasDataFrom>
                                                                                </
process:Perform>
```

```
                                                                                              </
    list:first>
                                                                                              <
    list:rest rdf:resource="&list;#nil"/>
                                                                                      </
    process:ControlConstructList>
                                                                              </list:rest>
                                                                      </
    process:ControlConstructList>
                                                              </list:rest>
                                                      </process:ControlConstructList>
                                              </list:rest>
                                      </process:ControlConstructList>
                              </list:rest>
                      </process:ControlConstructList>
              </process:components>
      </process:Sequence>
      </process:composedOf>

      <process:hasResult>
      <process:Result>
              <process:inCondition rdf:resource="&expr;#AlwaysTrue"/>
              <process:withOutput>
                      <process:OutputBinding>
                              <process:toParam rdf:resource="#BookMedicalAppointmentTimes"/>
                              <process:valueSource>
                              <process:ValueOf>
                                      <process:theVar rdf:resource="#RequestFreeSlotsTimeList"/>
                                      <process:fromProcess rdf:resource="#
    RequestFreeSlotsPerform"/>
                              </process:ValueOf>
                              </process:valueSource>
                      </process:OutputBinding>
              </process:withOutput>
      </process:Result>
      </process:hasResult>

      <process:hasResult>
      <process:Result>
              <process:inCondition rdf:resource="&expr;#AlwaysTrue"/>
              <process:withOutput>
                      <process:OutputBinding>
                              <process:toParam rdf:resource="#BookMedicalAppointmentAppointment"
    />
                              <process:valueSource>
                              <process:ValueOf>
                                      <process:theVar rdf:resource="#BookAppointmentAppointment"
    />
                                      <process:fromProcess rdf:resource="BookAppointmentPerform"
    />
                              </process:ValueOf>
                              </process:valueSource>
                      </process:OutputBinding>
              </process:withOutput>
      </process:Result>
      </process:hasResult>

      <process:hasResult>
      <process:Result>
```

```
            <process:inCondition rdf:resource="&expr;#AlwaysTrue"/>
            <process:withOutput>
                    <process:OutputBinding>
                            <process:toParam rdf:resource="#
   BookMedicalAppointmentAppointmentGuide"/>
                            <process:valueSource>
                            <process:ValueOf>
                                    <process:theVar rdf:resource="#GetInstructionsInstructions
   "/>
                                    <process:fromProcess rdf:resource="GetInstructionsPerform"
   />
                            </process:ValueOf>
                            </process:valueSource>
                    </process:OutputBinding>
            </process:withOutput>
      </process:Result>
      </process:hasResult>

</process:CompositeProcess>


<!--
=======================================================
      CompositeProcess: FindCareSequence
=======================================================
-->

<process:CompositeProcess rdf:ID="FindCareSequence">
      <process:hasInput>
      <process:Input rdf:ID="FindCareSequenceSearchTerm">
              <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#SearchTerm</
   process:parameterType>
      </process:Input>
      </process:hasInput>

      <process:hasInput>
      <process:Input rdf:ID="FindCareSequencePCT">
              <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#PCT</
   process:parameterType>
      </process:Input>
      </process:hasInput>

      <process:hasOutput>
      <process:Output rdf:ID="FindCareSequenceCareOrganization">
              <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#CareOrganization</
   process:parameterType>
      </process:Output>
      </process:hasOutput>

      <process:composedOf>
      <process:Sequence>
              <process:components>
                      <process:ControlConstructList>
                      <list:first>
                      <process:Perform rdf:ID="LookUpConditionPerform">
                              <process:process rdf:resource="#LookUpCondition"/>
                              <process:hasDataFrom>
                                      <process:InputBinding>
```

```
                                                <process:toParam rdf:resource="#
LookUpConditionSearchTerm"/>
                                                <process:valueSource>
                                                <process:ValueOf>
                                                        <process:theVar rdf:resource="#
FindCareSequenceSearchTerm"/>
                                                        <process:fromProcess rdf:resource=
"&process;#TheParentPerform"/>
                                                </process:ValueOf>
                                                </process:valueSource>
                                        </process:InputBinding>
                                </process:hasDataFrom>
                        </process:Perform>
                        </list:first>
                        <list:rest>
                        <process:Perform rdf:ID="RequestLocationsPerform">
                                <process:process rdf:resource="#RequestLocations"/>
                                <process:hasDataFrom>
                                        <process:InputBinding>
                                                <process:toParam rdf:resource="#
RequestLocationsPCT"/>
                                                <process:valueSource>
                                                <process:ValueOf>
                                                        <process:theVar rdf:resource="#
FindCareSequencePCT"/>
                                                        <process:fromProcess rdf:resource=
"&process;#TheParentPerform"/>
                                                </process:ValueOf>
                                                </process:valueSource>
                                        </process:InputBinding>
                                </process:hasDataFrom>
                                                <process:hasDataFrom>
                                        <process:InputBinding>
                                                <process:toParam rdf:resource="#
RequestLocationsMedicalCondition"/>
                                                <process:valueSource>
                                                <process:ValueOf>
                                                        <process:theVar rdf:resource="#
LookUpConditionCondition"/>
                                                        <process:fromProcess rdf:resource=
"&THIS;#LookUpConditionPerform"/>
                                                </process:ValueOf>
                                                </process:valueSource>
                                        </process:InputBinding>
                                </process:hasDataFrom>
                        </process:Perform>
                        </list:rest>
                </process:ControlConstructList>
        </process:components>
   </process:Sequence>
   </process:composedOf>

   <process:hasResult>
   <process:Result>
        <process:inCondition rdf:resource="&expr;#AlwaysTrue"/>
        <process:withOutput>
                <process:OutputBinding>
                        <process:toParam rdf:resource="#FindCareSequenceCareOrganization"/
>
```

```
                                <process:valueSource>
                                <process:ValueOf>
                                        <process:theVar rdf:resource="#
   RequestLocationsCareOrganization"/>
                                        <process:fromProcess rdf:resource="#
   RequestLocationsPerform"/>
                                </process:ValueOf>
                                </process:valueSource>
                        </process:OutputBinding>
                </process:withOutput>
        </process:Result>
        </process:hasResult>
</process:CompositeProcess>

<!--
=======================================================
      AtomicProcess: LocalPCT
=======================================================
-->

<process:AtomicProcess rdf:ID="LocalPCT">
      <process:hasInput>
              <process:Input rdf:ID="LocalPCTLocation">
                      <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#Location</
   process:parameterType>
              </process:Input>
      </process:hasInput>
      <process:hasOutput>
      <process:Output rdf:ID="LocalPCTPCT">
              <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#PCT</
   process:parameterType>
      </process:Output>
      </process:hasOutput>
</process:AtomicProcess>

<!--
=======================================================
      AtomicProcess: LookUpCondition
=======================================================
-->

<process:AtomicProcess rdf:ID="LookUpCondition">
      <process:hasInput>
              <process:Input rdf:ID="LookUpConditionSearchTerm">
                      <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#SearchTerm</
   process:parameterType>
              </process:Input>
      </process:hasInput>
      <process:hasOutput>
      <process:Output rdf:ID="LookUpConditionCondition">
              <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#MedicalCondition</
   process:parameterType>
      </process:Output>
      </process:hasOutput>
</process:AtomicProcess>

<!--
=======================================================
      AtomicProcess: RequestLocations
```

```
=====================================================
-->

<process:AtomicProcess rdf:ID="RequestLocations">
      <process:hasInput>
              <process:Input rdf:ID="RequestLocationsPCT">
                      <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#PCT</
   process:parameterType>
              </process:Input>
      </process:hasInput>
      <process:hasInput>
              <process:Input rdf:ID="RequestLocationsMedicalCondition">
                      <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#
   MedicalCondition</process:parameterType>
              </process:Input>
      </process:hasInput>
      <process:hasOutput>
      <process:Output rdf:ID="RequestLocationsCareOrganization">
              <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#CareOrganization</
   process:parameterType>
      </process:Output>
      </process:hasOutput>
</process:AtomicProcess>

<!--
=====================================================
      AtomicProcess: RequestFreeSlots
=====================================================
-->

<process:AtomicProcess rdf:ID="RequestFreeSlots">
      <process:hasInput>
              <process:Input rdf:ID="RequestFreeSlotsDepartment">
                      <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#
   HospitalDepartment</process:parameterType>
              </process:Input>
      </process:hasInput>
      <process:hasOutput>
      <process:Output rdf:ID="RequestFreeSlotsTimeList">
              <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#AppointmentTimeList<
   /process:parameterType>
      </process:Output>
      </process:hasOutput>
</process:AtomicProcess>

<!--
=====================================================
      AtomicProcess: BookAppointment
=====================================================
-->

<process:AtomicProcess rdf:ID="BookAppointment">
      <process:hasInput>
              <process:Input rdf:ID="BookAppointmentTime">
                      <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#
   AppointmentTime</process:parameterType>
              </process:Input>
      </process:hasInput>
      <process:hasInput>
```

```
                <process:Input rdf:ID="BookAppointmentHospitalDepartment">
                        <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#
    HospitalDepartment</process:parameterType>
                </process:Input>
        </process:hasInput>
        <process:hasOutput>
        <process:Output rdf:ID="BookAppointmentAppointment">
                <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#MedicalAppointment</
    process:parameterType>
        </process:Output>
        </process:hasOutput>
  </process:AtomicProcess>

  <!--
  ========================================================
        AtomicProcess: GetInstructions
  ========================================================
  -->

  <process:AtomicProcess rdf:ID="GetInstructions">
        <process:hasInput>
                <process:Input rdf:ID="GetInstructionsOrganization">
                        <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#
    CareOrganization</process:parameterType>
                </process:Input>
        </process:hasInput>
        <process:hasOutput>
        <process:Output rdf:ID="GetInstructionsInstructions">
                <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#AppointmentGuide</
    process:parameterType>
        </process:Output>
        </process:hasOutput>
  </process:AtomicProcess>

</rdf:RDF>
```

A.5.4 GenerateReportFromItinery.owls

```
<?xml version="1.0" encoding="WINDOWS-1252"?>
<!DOCTYPE uridef [
  <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema">
  <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns">
  <!ENTITY owl "http://www.w3.org/2002/07/owl">
  <!ENTITY xsd "http://www.w3.org/2001/XMLSchema">
  <!ENTITY service "http://www.daml.org/services/owl-s/1.2/Service.owl">
  <!ENTITY profile "http://www.daml.org/services/owl-s/1.2/Profile.owl">
  <!ENTITY process "http://www.daml.org/services/owl-s/1.2/Process.owl">
  <!ENTITY grounding "http://www.daml.org/services/owl-s/1.2/Grounding.owl">
  <!ENTITY list "http://www.daml.org/services/owl-s/1.2/generic/ObjectList.owl">
  <!ENTITY swrl "http://www.w3.org/2003/11/swrl">
  <!ENTITY swrlb "http://www.w3.org/2003/11/swrlb">
  <!ENTITY expr "http://www.daml.org/services/owl-s/1.2/generic/Expression.owl">
  <!ENTITY concepts "http://testcases.freya.local/ontology/concepts.owl">
  <!ENTITY THIS "http://testcases.freya.local/service/generateReportForItinerary.owls">
]>

<rdf:RDF
```

```
      xmlns:rdf="&rdf;#"
      xmlns:xsd="&xsd;#"
      xmlns:rdfs="&rdfs;#"
      xmlns:owl="&owl;#"
      xmlns:expr="&expr;#"
      xmlns:service="&service;#"
      xmlns:grounding="&grounding;#"
  xmlns:process="&process;#"
      xmlns:profile="&profile;#"
  xmlns:list="&list;#"
  xmlns:swrl="&swrl;#"
  xmlns:swrlb="&swrlb;#"
      xmlns:concepts="&concepts;#"
  xmlns="&THIS;#"
      xml:base="&THIS;#">

<owl:Ontology rdf:about="">
      <owl:versionInfo>
              $ID: generateReportForItiniery.owls, v 1.4 2011/02/28 akd07r $
      </owl:versionInfo>
      <rdfs:comment>
              TODO
      </rdfs:comment>
      <owl:imports rdf:resource="&service;" />
      <owl:imports rdf:resource="&process;" />
      <owl:imports rdf:resource="&profile;" />
      <owl:imports rdf:resource="&grounding;" />
      <owl:imports rdf:resource="&concepts;" />
      <owl:imports rdf:resource="http://127.0.0.1/ontology/portal.owl" />
      <owl:imports rdf:resource="http://127.0.0.1/ontology/support.owl" />
</owl:Ontology>

<!--
================================================
================================================
      Service Model
================================================
================================================
-->
<service:Service rdf:ID="GenerateReportForItineraryService">
      <service:presents rdf:resource="#GenerateReportForItineraryProfile"/>
      <service:describedBy rdf:resource="#GenerateReportForItinerary"/>
      <!--<service:supports rdf:resource="#GenerateReportForItineraryServiceGrounding"/>-->
</service:Service>

<!--
================================================
================================================
      Profile Model
================================================
================================================
-->
<profile:Profile rdf:ID="GenerateReportForItineraryProfile">
      <profile:serviceName>
              GenerateReportForItinerary_Service
      </profile:serviceName>
      <profile:textDescription>
      TODO
      </profile:textDescription>
```

```
      <service:presentedBy rdf:resource="#GenerateReportForItineraryService"/>
      <profile:has_process rdf:resource="#GenerateReportForItinerary"/>

      <profile:hasInput rdf:resource="#GenerateReportForItineraryCreateAccInfo"/>
      <profile:hasInput rdf:resource="#GenerateReportForItinerarySignInData"/>
      <profile:hasInput rdf:resource="#GenerateReportForItineraryFlight"/>
      <profile:hasInput rdf:resource="#GenerateReportForItineraryTransfer"/>
      <profile:hasInput rdf:resource="#GenerateReportForItineraryHotel"/>

      <profile:hasOutput rdf:resource="#GenerateReportForItineraryAcctID"/>
      <profile:hasOutput rdf:resource="#GenerateReportForItineraryReport"/>
</profile:Profile>

<!--
=======================================================
=======================================================
      Process Model
=======================================================
=======================================================
-->

<!--
=======================================================
      CompositeProcess: GenerateReportForItinerary
=======================================================
-->

<process:CompositeProcess rdf:ID="GenerateReportForItinerary">
      <process:hasInput>
            <process:Input rdf:ID="GenerateReportForItineraryCreateAccInfo">
                  <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#AcctInfo</
   process:parameterType>
            </process:Input>
      </process:hasInput>

      <process:hasInput>
            <process:Input rdf:ID="GenerateReportForItinerarySignInData">
                  <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#SignInData</
   process:parameterType>
            </process:Input>
      </process:hasInput>

      <process:hasInput>
            <process:Input rdf:ID="GenerateReportForItineraryFlight">
                  <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#Flight</
   process:parameterType>
            </process:Input>
      </process:hasInput>

      <process:hasInput>
            <process:Input rdf:ID="GenerateReportForItineraryTransfer">
                  <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#Transfer</
   process:parameterType>
            </process:Input>
      </process:hasInput>

      <process:hasInput>
            <process:Input rdf:ID="GenerateReportForItineraryHotel">
```

```
                <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#Hotel</
process:parameterType>
           </process:Input>
   </process:hasInput>

   <process:hasOutput>
           <process:Output rdf:ID="GenerateReportForItineraryAcctID">
           <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#AcctID</
process:parameterType>
       </process:Output>
   </process:hasOutput>

   <process:hasOutput>
           <process:Output rdf:ID="GenerateReportForItineraryReport">
           <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#ItineraryReport</
process:parameterType>
       </process:Output>
   </process:hasOutput>

   <process:composedOf>
   <process:Sequence>
           <process:components>
                   <process:ControlConstructList>
                           <list:first>
                                   <process:Perform rdf:ID="SignInAlternativesPerform
">
                           <process:process rdf:resource="#SignInAlternatives"/>
                                   <process:hasDataFrom>
                                   <process:InputBinding>
                                   <process:toParam rdf:resource="#
SignInAlternativesCreateAcctInfo"/>
                                   <process:valueSource>
                                           <process:ValueOf>
                                                   <process:theVar rdf:resource="#
GenerateReportForItineraryAcctInfo"/>
                                                   <process:fromProcess rdf:resource=
"&process;#TheParentPerform"/>
                                           </process:ValueOf>
                                           </process:valueSource>
                                   </process:InputBinding>
                                   </process:hasDataFrom>
                                <process:hasDataFrom>
                                   <process:InputBinding>
                                   <process:toParam rdf:resource="#
SignInAlternativesSignInData"/>
                                   <process:valueSource>
                                           <process:ValueOf>
                                                   <process:theVar rdf:resource="#
GenerateReportForItinerarySignInData"/>
                                                   <process:fromProcess rdf:resource=
"&process;#TheParentPerform"/>
                                           </process:ValueOf>
                                   </process:valueSource>
                                </process:InputBinding>
                                   </process:hasDataFrom>
                           </process:Perform>
                           </list:first>
                           <list:rest>
                           <process:ControlConstructList>
```

```
                                        <list:first>
                                        <process:Perform rdf:ID="CreateItineraryPerform">
                                                <process:process rdf:resource="#
CreateItinerary"/>
                                                <process:hasDataFrom>
                                                        <process:InputBinding>
                                                                <process:toParam
rdf:resource="#CreateItineraryAcctID"/>
                                                                <process:valueSource>
                                                                <process:ValueOf>
                                                                        <process:theVar
rdf:resource="#SignInAlternativesAcctID"/>
                                                                        <
process:fromProcess rdf:resource="&THIS;#SignInAlternativesPerform"/>
                                                                </process:ValueOf>
                                                                </process:valueSource>
                                                        </process:InputBinding>
                                                </process:hasDataFrom>
                                        </process:Perform>
                                        </list:first>
                                        <list:rest>
                                        <process:ControlConstructList>
                                                        <list:first>
                                                        <process:Perform rdf:ID="
TravelItemIterationPerform">
                                                                <process:process
rdf:resource="#TravelItemIteration"/>
                                                                <process:hasDataFrom>
                                                                        <
process:InputBinding>
                                                                                <
process:toParam rdf:resource="#TravelItemIterationTravelItinerary"/>
                                                                                <
process:valueSource>
                                                                                <
process:ValueOf>
                                                                                        <
process:theVar rdf:resource="#CreateItineraryItinerary"/>
                                                                                        <
process:fromProcess rdf:resource="&THIS;#CreateItineraryPerform"/>
                                                                                </
process:ValueOf>
                                                                                </
process:valueSource>
                                                                        </
process:InputBinding>
                                                                </process:hasDataFrom>
                                                                        <
process:hasDataFrom>
                                                                        <
process:InputBinding>
                                                                                <
process:toParam rdf:resource="#TravelItemIterationFlight"/>
                                                                                <
process:valueSource>
                                                                                <
process:ValueOf>
                                                                                        <
process:theVar rdf:resource="#GenerateReportForItineraryFlight"/>
```

```
                                                                                <
process:fromProcess rdf:resource="&process;#TheParentPerform"/>
                                                                                </
process:ValueOf>
                                                                                </
process:valueSource>
                                                                        </
process:InputBinding>
                                                                </process:hasDataFrom>
                                                                        <
process:hasDataFrom>
                                                                        <
process:InputBinding>
                                                                                <
process:toParam rdf:resource="#TravelItemIterationTransfer"/>
                                                                                <
process:valueSource>
                                                                                <
process:ValueOf>
                                                                                <
process:theVar rdf:resource="#GenerateReportForItineraryTransfer"/>
                                                                                <
process:fromProcess rdf:resource="&process;#TheParentPerform"/>
                                                                                </
process:ValueOf>
                                                                                </
process:valueSource>
                                                                        </
process:InputBinding>
                                                                </process:hasDataFrom>
                                                                        <
process:hasDataFrom>
                                                                        <
process:InputBinding>
                                                                                <
process:toParam rdf:resource="#TravelItemIterationHotel"/>
                                                                                <
process:valueSource>
                                                                                <
process:ValueOf>
                                                                                <
process:theVar rdf:resource="#GenerateReportForItineraryHotel"/>
                                                                                <
process:fromProcess rdf:resource="&process;#TheParentPerform"/>
                                                                                </
process:ValueOf>
                                                                                </
process:valueSource>
                                                                        </
process:InputBinding>
                                                                </process:hasDataFrom>
                                                        </process:Perform>
                                                        </list:first>
                                                        <list:rest>
                                                        <process:ControlConstructList>
                                                                <list:first>
                                                                <process:Perform
rdf:ID="VerifyItineraryPerform">
```

```
                                                                    <
process:process rdf:resource="#VerifyItinerary"/>
                                                                    <
process:hasDataFrom>
                                                                        <
process:InputBinding>
        <process:toParam rdf:resource="#VerifyItineraryTravelItinerary"/>
        <process:valueSource>
        <process:ValueOf>
                <process:theVar rdf:resource="#TravelItemIterationItinerary"/>
                <process:fromProcess rdf:resource="&THIS;#TravelItemIterationPerform"/>
        </process:ValueOf>
        </process:valueSource>
                                                                        </
process:InputBinding>
                                                                    </
process:hasDataFrom>
                                                                </process:Perform>
                                                                </list:first>
                                                                <list:rest>
                                                                <
process:ControlConstructList>
                                                                    <
list:first>
                                                                    <
process:Perform rdf:ID="GenerateReportPerform">
        <process:process rdf:resource="#GenerateReport"/>
        <process:hasDataFrom>
                <process:InputBinding>
                        <process:toParam rdf:resource="#GenerateReportItinerary"/>
                        <process:valueSource>
                        <process:ValueOf>
                                <process:theVar rdf:resource="#
VerifyItineraryVerificationReport"/>
                                <process:fromProcess rdf:resource="&THIS;#
VerifyItineraryPerform"/>
                        </process:ValueOf>
                        </process:valueSource>
                </process:InputBinding>
        </process:hasDataFrom>
```

```
                                                                                                    </
process:Perform>
                                                                                                    </
list:first>
                                                                                                    <
list:rest rdf:resource="&list;#nil"/>
                                                                                                </
process:ControlConstructList>
                                                                                            </list:rest>
                                                                                    </
process:ControlConstructList>
                                                                            </list:rest>
                                                                    </process:ControlConstructList>
                                                            </list:rest>
                                                    </process:ControlConstructList>
                                            </list:rest>
                                    </process:ControlConstructList>
                            </process:components>
   </process:Sequence>
   </process:composedOf>

   <process:hasResult>
   <process:Result>
           <process:inCondition rdf:resource="&expr;#AlwaysTrue"/>
           <process:withOutput>
                   <process:OutputBinding>
                           <process:toParam rdf:resource="#GenerateReportForItineraryAcctID"/
>
                           <process:valueSource>
                           <process:ValueOf>
                                   <process:theVar rdf:resource="#SignInAlternativesAcctID"/>
                                   <process:fromProcess rdf:resource="#
SignInAlternativesPerform"/>
                           </process:ValueOf>
                           </process:valueSource>
                   </process:OutputBinding>
           </process:withOutput>
   </process:Result>
   </process:hasResult>

   <process:hasResult>
   <process:Result>
           <process:inCondition rdf:resource="&expr;#AlwaysTrue"/>
           <process:withOutput>
                   <process:OutputBinding>
                           <process:toParam rdf:resource="#GenerateReportForItineraryReport"/
>
                           <process:valueSource>
                           <process:ValueOf>
                                   <process:theVar rdf:resource="#GenerateReportReport"/>
                                   <process:fromProcess rdf:resource="GenerateReportPerform"/
>
                           </process:ValueOf>
                           </process:valueSource>
                   </process:OutputBinding>
           </process:withOutput>
   </process:Result>
   </process:hasResult>
```

```
  </process:CompositeProcess>

<!--
  ======================================================
        CompositeProcess: TravelItemIteration
  ======================================================
  -->

  <process:CompositeProcess rdf:ID="TravelItemIteration">
        <process:hasInput>
                <process:Input rdf:ID="TravelItemIterationTravelItinerary">
                        <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#
     TravelItinerary</process:parameterType>
                </process:Input>
        </process:hasInput>

        <process:hasInput>
                <process:Input rdf:ID="TravelItemIterationFlight">
                        <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#Flight</
     process:parameterType>
                </process:Input>
        </process:hasInput>

        <process:hasInput>
                <process:Input rdf:ID="TravelItemIterationTransfer">
                        <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#Transfer</
     process:parameterType>
                </process:Input>
        </process:hasInput>

        <process:hasInput>
                <process:Input rdf:ID="TravelItemIterationHotel">
                        <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#Hotel</
     process:parameterType>
                </process:Input>
        </process:hasInput>

        <process:hasOutput>
                <process:Output rdf:ID="TravelItemIterationItinerary">
                <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#TravelItinerary</
     process:parameterType>
            </process:Output>
        </process:hasOutput>

        <process:composedOf>
        <process:Iterate>
                <process:components>
                        <process:ControlConstructBag>
                                <list:first>
                                <process:Perform rdf:ID="AddTravelItemPerform">
                                        <process:process rdf:resource="#AddTravelItem"/>
                                        <process:hasDataFrom>
                                                <process:InputBinding>
                                                        <process:toParam rdf:resource="#
     AddTravelItemTravelItinerary"/>
                                                        <process:valueSource>
                                                        <process:ValueOf>
                                                                <process:theVar rdf:resource="#
     TravelItemIterationTravelItinerary"/>
```

```
                                                        <process:fromProcess rdf:resource=
"&process;#TheParentPerform"/>
                                                </process:ValueOf>
                                                </process:valueSource>
                                        </process:InputBinding>
                                </process:hasDataFrom>
                                                <process:hasDataFrom>
                                        <process:InputBinding>
                                                <process:toParam rdf:resource="#
AddTravelItemFlight"/>
                                                <process:valueSource>
                                                <process:ValueOf>
                                                        <process:theVar rdf:resource="#
TravelItemIterationFlight"/>
                                                        <process:fromProcess rdf:resource=
"&process;#TheParentPerform"/>
                                                </process:ValueOf>
                                                </process:valueSource>
                                        </process:InputBinding>
                                </process:hasDataFrom>
                                                <process:hasDataFrom>
                                        <process:InputBinding>
                                                <process:toParam rdf:resource="#
AddTravelItemTransfer"/>
                                                <process:valueSource>
                                                <process:ValueOf>
                                                        <process:theVar rdf:resource="#
TravelItemIterationTransfer"/>
                                                        <process:fromProcess rdf:resource=
"&process;#TheParentPerform"/>
                                                </process:ValueOf>
                                                </process:valueSource>
                                        </process:InputBinding>
                                </process:hasDataFrom>
                                                <process:hasDataFrom>
                                        <process:InputBinding>
                                                <process:toParam rdf:resource="#
AddTravelItemHotel"/>
                                                <process:valueSource>
                                                <process:ValueOf>
                                                        <process:theVar rdf:resource="#
TravelItemIterationHotel"/>
                                                        <process:fromProcess rdf:resource=
"&process;#TheParentPerform"/>
                                                </process:ValueOf>
                                                </process:valueSource>
                                        </process:InputBinding>
                                </process:hasDataFrom>
                        </process:Perform>
                        </list:first>
                        <list:rest rdf:resource="&list;#nil"/>
                </process:ControlConstructBag>
        </process:components>
    </process:Iterate>
  </process:composedOf>

  <process:hasResult>
  <process:Result>
        <process:inCondition rdf:resource="&expr;#AlwaysTrue"/>
```

```
            <process:withOutput>
                    <process:OutputBinding>
                            <process:toParam rdf:resource="#TravelItemIterationItinerary"/>
                            <process:valueSource>
                            <process:ValueOf>
                                    <process:theVar rdf:resource="#AddTravelItemItinerary"/>
                                    <process:fromProcess rdf:resource="#AddTravelItemPerform"/
  >
                            </process:ValueOf>
                            </process:valueSource>
                    </process:OutputBinding>
            </process:withOutput>
      </process:Result>
      </process:hasResult>

</process:CompositeProcess>

<!--
=======================================================
      CompositeProcess: AddTravelItem
=======================================================
-->

<process:CompositeProcess rdf:ID="AddTravelItem">
      <process:hasInput>
              <process:Input rdf:ID="AddTravelItemTravelItinerary">
                      <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#
  TravelItinerary</process:parameterType>
              </process:Input>
      </process:hasInput>

      <process:hasInput>
              <process:Input rdf:ID="AddTravelItemFlight">
                      <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#Flight</
  process:parameterType>
              </process:Input>
      </process:hasInput>

      <process:hasInput>
              <process:Input rdf:ID="AddTravelItemTransfer">
                      <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#Transfer</
  process:parameterType>
              </process:Input>
      </process:hasInput>

      <process:hasInput>
              <process:Input rdf:ID="AddTravelItemHotel">
                      <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#Hotel</
  process:parameterType>
              </process:Input>
      </process:hasInput>

      <process:hasOutput>
              <process:Output rdf:ID="AddTravelItemItinerary">
              <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#TravelItinerary</
  process:parameterType>
          </process:Output>
      </process:hasOutput>
```

```
   <process:composedOf>
   <process:Choice>
           <process:components>
                   <process:ControlConstructBag>
                           <list:first>
                           <process:Perform rdf:ID="AddFlightPerform">
                                   <process:process rdf:resource="#AddFlight"/>
                                   <process:hasDataFrom>
                                           <process:InputBinding>
                                                   <process:toParam rdf:resource="#
AddFlightFlight"/>
                                                   <process:valueSource>
                                                   <process:ValueOf>
                                                           <process:theVar rdf:resource="#
AddTravelItemFlight"/>
                                                           <process:fromProcess rdf:resource=
"&process;#TheParentPerform"/>
                                                   </process:ValueOf>
                                                   </process:valueSource>
                                           </process:InputBinding>
                                   </process:hasDataFrom>
                           </process:Perform>
                           </list:first>
                           <list:rest>
                           <process:ControlConstructBag>
                                   <list:first>
                                           <process:Perform rdf:ID="AddTranfserPerform">
                                                   <process:process rdf:resource="#
AddTransfer"/>
                                                   <process:hasDataFrom>
                                                   <process:InputBinding>
                                                           <process:toParam rdf:resource="#
AddTransferTransfer"/>
                                                           <process:valueSource>
                                                                   <process:ValueOf>
                                                                           <process:theVar
rdf:resource="#AddTravelItemTransfer"/>
                                                                           <
process:fromProcess rdf:resource="&process;#TheParentPerform"/>
                                                                   </process:ValueOf>
                                                           </process:valueSource>
                                                   </process:InputBinding>
                                                   </process:hasDataFrom>
                                           </process:Perform>
                                   </list:first>
                                   <list:rest>
                                                   <process:ControlConstructBag>
                                           <list:first>
                                                   <process:Perform rdf:ID="
AddHotelPerform">
                                                           <process:process
rdf:resource="#AddHotel"/>
                                                           <process:hasDataFrom>
                                                           <process:InputBinding>
                                                                   <process:toParam
rdf:resource="#AddHotelHotel"/>
                                                                   <
process:valueSource>
```

```
                                                                                <
process:ValueOf>
                                                                                        <
process:theVar rdf:resource="#AddTravelItemHotel"/>
                                                                                        <
process:fromProcess rdf:resource="&process;#TheParentPerform"/>
                                                                                <
process:ValueOf>
                                                                        <
process:valueSource>
                                                                                </process:InputBinding>
                                                                                </process:hasDataFrom>
                                                                        </process:Perform>
                                                                </list:first>
                                                                <list:rest rdf:resource="&list;#nil"/>
                                                        </process:ControlConstructBag>
                                                                <list:rest>
                                        </process:ControlConstructBag>
                                        </list:rest>
                                </process:ControlConstructBag>
                        </process:components>
                </process:Choice>
        </process:composedOf>

        <process:hasResult>
        <process:Result>
                <process:inCondition rdf:resource="&expr;#AlwaysTrue"/>
                <process:withOutput>
                        <process:OutputBinding>
                                <process:toParam rdf:resource="#AddTravelItemItinerary"/>
                                <process:valueSource>
                                <process:ValueOf>
                                        <process:theVar rdf:resource="#AddFlightItinerary"/>
                                        <process:fromProcess rdf:resource="#AddFlightPerform"/>
                                </process:ValueOf>
                                </process:valueSource>
                        </process:OutputBinding>
                </process:withOutput>
        </process:Result>
        </process:hasResult>

        <process:hasResult>
        <process:Result>
                <process:inCondition rdf:resource="&expr;#AlwaysTrue"/>
                <process:withOutput>
                        <process:OutputBinding>
                                <process:toParam rdf:resource="#AddTravelItemItinerary"/>
                                <process:valueSource>
                                <process:ValueOf>
                                        <process:theVar rdf:resource="#AddTransferItinerary"/>
                                        <process:fromProcess rdf:resource="#AddTransferPerform"/>
                                </process:ValueOf>
                                </process:valueSource>
                        </process:OutputBinding>
                </process:withOutput>
        </process:Result>
        </process:hasResult>

        <process:hasResult>
```

```
      <process:Result>
              <process:inCondition rdf:resource="&expr;#AlwaysTrue"/>
              <process:withOutput>
                      <process:OutputBinding>
                              <process:toParam rdf:resource="#AddTravelItemItinerary"/>
                              <process:valueSource>
                              <process:ValueOf>
                                      <process:theVar rdf:resource="#AddHotelItinerary"/>
                                      <process:fromProcess rdf:resource="#AddHotelPerform"/>
                              </process:ValueOf>
                              </process:valueSource>
                      </process:OutputBinding>
              </process:withOutput>
      </process:Result>
      </process:hasResult>

</process:CompositeProcess>

<!--
=======================================================
      CompositeProcess: SignInAlternatives
=======================================================
-->

<process:CompositeProcess rdf:ID="SignInAlternatives">
      <rdfs:comment>
              TODO
      </rdfs:comment>

      <process:hasInput>
      <process:Input rdf:ID="SignInAlternativesCreateAcctInfo">
              <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#AcctInfo</
  process:parameterType>
      </process:Input>
      </process:hasInput>

      <process:hasInput>
      <process:Input rdf:ID="SignInAlternativesSignInData">
              <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#SignInData</
  process:parameterType>
      </process:Input>
      </process:hasInput>

      <process:hasOutput>
      <process:Output rdf:ID="SignInAlternativesAcctID">
              <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#AcctID</
  process:parameterType>
      </process:Output>
      </process:hasOutput>

      <process:composedOf>
      <process:Choice>
              <process:components>
                      <process:ControlConstructBag>
                              <list:first>
                              <process:Perform rdf:ID="CreateAcctSequencePerform">
                                      <process:process rdf:resource="#CreateAcctSequence"/>
                                      <process:hasDataFrom>
                                              <process:InputBinding>
```

```
                                                <process:toParam rdf:resource="#
CreateAcctSequenceCreateAcctInfo"/>
                                                <process:valueSource>
                                                <process:ValueOf>
                                                        <process:theVar rdf:resource="#
SignInAlternativesCreateAcctInfo"/>
                                                        <process:fromProcess rdf:resource=
"&process;#TheParentPerform"/>
                                                </process:ValueOf>
                                                </process:valueSource>
                                        </process:InputBinding>
                                </process:hasDataFrom>
                        </process:Perform>
                        </list:first>
                        <list:rest>
                        <process:ControlConstructBag>
                                <list:first>
                                        <process:Perform rdf:ID="SignInSequencePerform">
                                                <process:process rdf:resource="#
SignInSequence"/>
                                                <process:hasDataFrom>
                                                <process:InputBinding>
                                                        <process:toParam rdf:resource="#
SignInSequenceSignInInfo"/>
                                                        <process:valueSource>
                                                                <process:ValueOf>
                                                                        <process:theVar
rdf:resource="#SignInAlternativesSignInData"/>
                                                                        <
process:fromProcess rdf:resource="&process;#TheParentPerform"/>
                                                                </process:ValueOf>
                                                        </process:valueSource>
                                                </process:InputBinding>
                                                </process:hasDataFrom>
                                        </process:Perform>
                                </list:first>
                                <list:rest rdf:resource="&list;#nil"/>
                        </process:ControlConstructBag>
                        </list:rest>
                </process:ControlConstructBag>
        </process:components>
    </process:Choice>
  </process:composedOf>

  <process:hasResult>
  <process:Result>
        <process:inCondition>
                <expr:SWRL-Condition rdf:ID="SignInAlternativesAcctExists">
                        <rdfs:label>
                                hasAcctID(SignInAlternativesSignInData,
SignInAlternativesAcctID)
                        </rdfs:label>
                        <rdfs:comment>
                        If an account already exists, sign-in operation will be
                        performed and returned acct ID willbe used
                        </rdfs:comment>
                        <expr:expressionLanguage rdf:resource="&expr;#SWRL"/>
                        <expr:expressionObject>
                        <swrl:AtomList>
```

```
                                        <rdf:first>
                                                <swrl:IndividualPropertyAtom>
                                                        <swrl:propertyPredicate rdf:resource="#
hasAcctID"/>
                                                        <swrl:argument1 rdf:resource="#
SignInAlternativesSignInData"/>
                                                        <swrl:argument2 rdf:resource="#
SignInAlternativesAcctID"/>
                                                </swrl:IndividualPropertyAtom>
                                        </rdf:first>
                                        <rdf:rest rdf:resource="&rdf;#nil"/>
                                </swrl:AtomList>
                                </expr:expressionObject>
                        </expr:SWRL-Condition>
                </process:inCondition>
                <process:withOutput>
                        <process:OutputBinding>
                                <process:toParam rdf:resource="#SignInAlternativesAcctID"/>
                                <process:valueSource>
                                <process:ValueOf>
                                        <process:theVar rdf:resource="#SignInSequenceAcctID"/>
                                        <process:fromProcess rdf:resource="#SignInSequencePerform"
/>
                                </process:ValueOf>
                                </process:valueSource>
                        </process:OutputBinding>
                </process:withOutput>
        </process:Result>
        </process:hasResult>

        <process:hasResult>
        <process:Result>
                <process:inCondition>
                        <expr:SWRL-Condition rdf:ID="SignInAlternativesNoAcctExists">
                                <rdfs:comment>
                                If an account does not exist a new account will be
                                create by the CreateAcct process and the ID of new
                                account will be used.
                                </rdfs:comment>
                                <expr:expressionLanguage rdf:resource="&expr;#SWRL"/>
                                <expr:expressionObject>
                                <swrl:AtomList>
                                        <rdf:first>
                                                <swrl:ClassAtom>
                                                        <swrl:classPredicate rdf:resource="#
NonExistingAcct"/>
                                                        <swrl:argument1 rdf:resource="#
SignInAlternativesSignInData"/>
                                                </swrl:ClassAtom>
                                        </rdf:first>
                                        <rdf:rest rdf:resource="&rdf;#nil"/>
                                </swrl:AtomList>
                                </expr:expressionObject>
                        </expr:SWRL-Condition>
                </process:inCondition>
                <process:withOutput>
                        <process:OutputBinding>
                                <process:toParam rdf:resource="#SignInAlternativesAcctID"/>
                                <process:valueSource>
```

```
                                <process:ValueOf>
                                        <process:theVar rdf:resource="#
   CreateAcctSequenceCreateAcctOutput"/>
                                        <process:fromProcess rdf:resource="#
   CreateAcctSequencePerform"/>
                                </process:ValueOf>
                                </process:valueSource>
                        </process:OutputBinding>
                </process:withOutput>
        </process:Result>
        </process:hasResult>
</process:CompositeProcess>



<!--
=======================================================
      CompositeProcess: SignInSequence
=======================================================
-->

<process:CompositeProcess rdf:ID="SignInSequence">
      <rdfs:comment>
      SignInSequence performs an atomic process SignIn,
      followed by an atomic process LoadUserProfile.
      </rdfs:comment>

      <process:hasInput>
      <process:Input rdf:ID="SignInSequenceSignInInfo">
              <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#SignInData</
   process:parameterType>
      </process:Input>
      </process:hasInput>

      <process:hasOutput>
      <process:Output rdf:ID="SignInSequenceAcctID">
              <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#AcctID</
   process:parameterType>
      </process:Output>
      </process:hasOutput>

      <process:composedOf>
      <process:Sequence>
              <process:components>
                      <process:ControlConstructList>
                              <list:first>
                              <process:Perform rdf:ID="SignInPerform">
                                      <process:process rdf:resource="#SignIn"/>
                                      <process:hasDataFrom>
                                              <process:InputBinding>
                                                      <process:toParam rdf:resource="#SignInInfo
   "/>
                                                      <process:valueSource>
                                                      <process:ValueOf>
                                                              <process:theVar rdf:resource="#
   SignInSequenceSignInInfo"/>
                                                              <process:fromProcess rdf:resource=
   "&process;#TheParentPerform"/>
                                                      </process:ValueOf>
```

```
                                                    </process:valueSource>
                                            </process:InputBinding>
                                    </process:hasDataFrom>
                            </process:Perform>
                            </list:first>
                            <list:rest>
                            <process:ControlConstructList>
                                    <list:first>
                                            <process:Perform rdf:ID="LoadUserProfilePerform">
                                                    <process:process rdf:resource="#
  LoadUserProfile"/>
                                            </process:Perform>
                                    </list:first>
                                    <list:rest rdf:resource="&list;#nil"/>
                            </process:ControlConstructList>
                            </list:rest>
                    </process:ControlConstructList>
            </process:components>
      </process:Sequence>
      </process:composedOf>
</process:CompositeProcess>


<!--
=======================================================
      CompositeProcess: CreateAcctSequence
=======================================================
-->

<process:CompositeProcess rdf:ID="CreateAcctSequence">
      <rdfs:comment>
      CreateAcctSequence performs atomic process
      CreateAcct followed by atomic process LoadUserProfile.
      </rdfs:comment>

      <process:hasInput>
      <process:Input rdf:ID="CreateAcctSequenceCreateAcctInfo">
              <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#AcctInfo</
  process:parameterType>
      </process:Input>
      </process:hasInput>

      <process:hasOutput>
      <process:Output rdf:ID="CreateAcctSequenceCreateAcctOutput">
              <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#AcctID</
  process:parameterType>
      </process:Output>
      </process:hasOutput>

      <process:composedOf>
      <process:Sequence>
              <process:components>
                      <process:ControlConstructList>
                              <list:first>
                              <process:Perform rdf:ID="CreateAcctPerform">
                                      <process:process rdf:resource="#CreateAcct"/>
                                      <process:hasDataFrom>
                                              <process:InputBinding>
```

```
                                                    <process:toParam rdf:resource="#
    CreateAcctInfo"/>
                                                    <process:valueSource>
                                                    <process:ValueOf>
                                                            <process:theVar rdf:resource="#
    CreateAcctSequenceCreateAcctInfo"/>
                                                            <process:fromProcess rdf:resource=
    "&process;#TheParentPerform"/>
                                                    </process:ValueOf>
                                                    </process:valueSource>
                                            </process:InputBinding>
                                    </process:hasDataFrom>
                            </process:Perform>
                            </list:first>
                            <list:rest>
                            <process:ControlConstructList>
                                    <list:first>
                                            <process:Perform rdf:ID="LoadUserProfilePerform1">
                                                    <process:process rdf:resource="#
    LoadUserProfile"/>
                                            </process:Perform>
                                    </list:first>
                                    <list:rest rdf:resource="&list;#nil"/>
                            </process:ControlConstructList>
                            </list:rest>
                    </process:ControlConstructList>
            </process:components>
        </process:Sequence>
        </process:composedOf>
        <process:hasResult>
        <process:Result>
                <process:inCondition rdf:resource="&expr;#AlwaysTrue"/>
                <process:withOutput>
                        <process:OutputBinding>
                                <process:toParam rdf:resource="#CreateAcctSequenceCreateAcctOutput
    "/>
                                <process:valueSource>
                                <process:ValueOf>
                                        <process:theVar rdf:resource="#CreateAcctOutput"/>
                                        <process:fromProcess rdf:resource="#CreateAcctPerform"/>
                                </process:ValueOf>
                                </process:valueSource>
                        </process:OutputBinding>
                </process:withOutput>
        </process:Result>
        </process:hasResult>
</process:CompositeProcess>

<!--
=====================================================
      AtomicProcess: SignIn
=====================================================
-->

<process:AtomicProcess rdf:ID="SignIn">
      <rdfs:comment>
      Sign in is a process that requires input of signin info.
      </rdfs:comment>
```

```
      <process:hasInput>
      <process:Input rdf:ID="SignInInfo">
              <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#SignInData</
   process:parameterType>
      </process:Input>
      </process:hasInput>
</process:AtomicProcess>


<!--
=======================================================
      AtomicProcess: CreateAcct
=======================================================
-->

<process:AtomicProcess rdf:ID="CreateAcct">
      <process:hasInput>
      <process:Input rdf:ID="CreateAcctInfo">
              <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#AcctInfo</
   process:parameterType>
      </process:Input>
      </process:hasInput>

      <process:hasOutput>
      <process:Output rdf:ID="CreateAcctOutput">
              <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#AcctID</
   process:parameterType>
      </process:Output>
      </process:hasOutput>
</process:AtomicProcess>


<!--
=======================================================
      AtomicProcess: LoadUserProfile
=======================================================
-->

<process:AtomicProcess rdf:ID="LoadUserProfile">
      <rdfs:comment>
      LoadUserProfile can only be invoked if a user profile already exists.
      </rdfs:comment>
</process:AtomicProcess>

<!--
=======================================================
      AtomicProcess: GenerateReport
=======================================================
-->

<process:AtomicProcess rdf:ID="GenerateReport">
      <process:hasInput>
              <process:Input rdf:ID="GenerateReportItinerary">
                      <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#
   VerificationReport</process:parameterType>
              </process:Input>
      </process:hasInput>
      <process:hasOutput>
      <process:Output rdf:ID="GenerateReportReport">
```

```
            <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#VerificationReport</
   process:parameterType>
      </process:Output>
      </process:hasOutput>
</process:AtomicProcess>

<!--
=====================================================
      AtomicProcess: VerifyItinerary
=====================================================
-->

<process:AtomicProcess rdf:ID="VerifyItinerary">
      <process:hasInput>
            <process:Input rdf:ID="VerifyItineraryTravelItinerary">
                  <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#
   TravelItinerary</process:parameterType>
            </process:Input>
      </process:hasInput>
      <process:hasOutput>
      <process:Output rdf:ID="VerifyItineraryVerificationReport">
            <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#VerificationReport</
   process:parameterType>
      </process:Output>
      </process:hasOutput>
</process:AtomicProcess>

<!--
=====================================================
      AtomicProcess: AddHotel
=====================================================
-->

<process:AtomicProcess rdf:ID="AddHotel">
      <process:hasInput>
            <process:Input rdf:ID="AddHotelHotel">
                  <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#Hotel</
   process:parameterType>
            </process:Input>
      </process:hasInput>
      <process:hasOutput>
      <process:Output rdf:ID="AddHotelItinerary">
            <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#TravelItinerary</
   process:parameterType>
      </process:Output>
      </process:hasOutput>
</process:AtomicProcess>

<!--
=====================================================
      AtomicProcess: AddTransfer
=====================================================
-->

<process:AtomicProcess rdf:ID="AddTransfer">
      <process:hasInput>
            <process:Input rdf:ID="AddTransferTransfer">
                  <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#Transfer</
   process:parameterType>
```

```
                </process:Input>
        </process:hasInput>
        <process:hasOutput>
        <process:Output rdf:ID="AddTransferItinerary">
                <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#TravelItinerary</
    process:parameterType>
        </process:Output>
        </process:hasOutput>
  </process:AtomicProcess>

  <!--
  ========================================================
        AtomicProcess: AddFlight
  ========================================================
  -->

  <process:AtomicProcess rdf:ID="AddFlight">
        <process:hasInput>
                <process:Input rdf:ID="AddFlightFilght">
                        <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#Flight</
    process:parameterType>
                </process:Input>
        </process:hasInput>
        <process:hasOutput>
        <process:Output rdf:ID="AddFlightItinerary">
                <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#TravelItinerary</
    process:parameterType>
        </process:Output>
        </process:hasOutput>
  </process:AtomicProcess>

  <!--
  ========================================================
        AtomicProcess: CreateItinerary
  ========================================================
  -->

  <process:AtomicProcess rdf:ID="CreateItinerary">
        <process:hasInput>
                <process:Input rdf:ID="CreateItineraryAcctID">
                        <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#AcctID</
    process:parameterType>
                </process:Input>
        </process:hasInput>
        <process:hasOutput>
        <process:Output rdf:ID="CreateItineraryItinerary">
                <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#TravelItinerary</
    process:parameterType>
        </process:Output>
        </process:hasOutput>
  </process:AtomicProcess>

</rdf:RDF>
```

A.5.5 BuyGenericItem.owls

```
<?xml version="1.0" encoding="WINDOWS-1252"?>
```

```
<!DOCTYPE uridef [
  <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema">
  <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns">
  <!ENTITY owl "http://www.w3.org/2002/07/owl">
  <!ENTITY xsd "http://www.w3.org/2001/XMLSchema">
  <!ENTITY service "http://www.daml.org/services/owl-s/1.2/Service.owl">
  <!ENTITY profile "http://www.daml.org/services/owl-s/1.2/Profile.owl">
  <!ENTITY process "http://www.daml.org/services/owl-s/1.2/Process.owl">
  <!ENTITY grounding "http://www.daml.org/services/owl-s/1.2/Grounding.owl">
  <!ENTITY list "http://www.daml.org/services/owl-s/1.2/generic/ObjectList.owl">
  <!ENTITY swrl "http://www.w3.org/2003/11/swrl">
  <!ENTITY swrlb "http://www.w3.org/2003/11/swrlb">
  <!ENTITY expr "http://www.daml.org/services/owl-s/1.2/generic/Expression.owl">
  <!ENTITY concepts "http://testcases.freya.local/ontology/concepts.owl">
  <!ENTITY THIS "http://testcases.freya.local/purchasing/buyGenericItem.owls">
]>

<rdf:RDF
        xmlns:rdf="&rdf;#"
        xmlns:xsd="&xsd;#"
        xmlns:rdfs="&rdfs;#"
        xmlns:owl="&owl;#"
        xmlns:expr="&expr;#"
        xmlns:service="&service;#"
        xmlns:grounding="&grounding;#"
    xmlns:process="&process;#"
        xmlns:profile="&profile;#"
    xmlns:list="&list;#"
    xmlns:swrl="&swrl;#"
    xmlns:swrlb="&swrlb;#"
        xmlns:concepts="&concepts;#"
    xmlns="&THIS;#"
        xml:base="&THIS;#">

  <owl:Ontology rdf:about="">
        <owl:versionInfo>
                $ID: buyGenericItem.owls, v 1.4 2011/02/28 akd07r $
        </owl:versionInfo>
        <rdfs:comment>
                TODO
        </rdfs:comment>
        <owl:imports rdf:resource="&service;" />
        <owl:imports rdf:resource="&process;" />
        <owl:imports rdf:resource="&profile;" />
        <owl:imports rdf:resource="&grounding;" />
        <owl:imports rdf:resource="&concepts;" />
        <owl:imports rdf:resource="http://127.0.0.1/ontology/portal.owl" />
        <owl:imports rdf:resource="http://127.0.0.1/ontology/support.owl" />
  </owl:Ontology>

  <!--
  ////////////////////////////////////////////////////
  ////////////////////////////////////////////////////
        Service Model
  ////////////////////////////////////////////////////
  ////////////////////////////////////////////////////
  -->
  <service:Service rdf:ID="BuyGenericItemService">
        <service:presents rdf:resource="#BuyGenericItemProfile"/>
```

```
      <service:describedBy rdf:resource="#BuyGenericItem"/>
      <!--<service:supports rdf:resource="#BuyGenericItemServiceGrounding"/>-->
</service:Service>

<!--
=====================================================
=====================================================
      Profile Model
=====================================================
=====================================================
-->
<profile:Profile rdf:ID="BuyGenericItemProfile">
      <profile:serviceName>
              BuyGenericItem_Service
      </profile:serviceName>
      <profile:textDescription>
      TODO
      </profile:textDescription>

      <service:presentedBy rdf:resource="#BuyGenericItemService"/>
      <profile:has_process rdf:resource="#BuyGenericItem"/>

      <profile:hasInput rdf:resource="#BuyGenericItemSearchTerm"/>
      <profile:hasInput rdf:resource="#BuyGenericItemCreateAccInfo"/>
      <profile:hasInput rdf:resource="#BuyGenericItemSignInData"/>
      <profile:hasInput rdf:resource="#BuyGenericItemCreditCard"/>
      <profile:hasInput rdf:resource="#BuyGenericItemDeliveryAddress"/>
      <profile:hasInput rdf:resource="#BuyGenericItemPackagingSelection"/>
      <profile:hasInput rdf:resource="#BuyGenericItemDeliveryTypeSelection"/>

      <profile:hasOutput rdf:resource="#BuyGenericItemShipmentDetails"/>
      <profile:hasOutput rdf:resource="#BuyGenericItemAcctID"/>
</profile:Profile>

<!--
=====================================================
=====================================================
      Process Model
=====================================================
=====================================================
-->

<!--
=====================================================
      CompositeProcess: BuyGenericItem
=====================================================
-->

<process:CompositeProcess rdf:ID="BuyGenericItem">
      <rdfs:comment>
      </rdfs:comment>

      <process:hasInput>
              <process:Input rdf:ID="BuyGenericItemSearchTerm">
                      <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#SearchTerm</
   process:parameterType>
              </process:Input>
      </process:hasInput>
```

```
   <process:hasInput>
           <process:Input rdf:ID="BuyGenericItemCreateAccInfo">
                   <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#AcctInfo</
process:parameterType>
           </process:Input>
   </process:hasInput>

   <process:hasInput>
           <process:Input rdf:ID="BuyGenericItemSignInData">
                   <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#SignInData</
process:parameterType>
           </process:Input>
   </process:hasInput>

   <process:hasInput>
           <process:Input rdf:ID="BuyGenericItemCreditCard">
                   <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#CreditCard</
process:parameterType>
           </process:Input>
   </process:hasInput>

   <process:hasInput>
           <process:Input rdf:ID="BuyGenericItemDeliveryAddress">
                   <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#
PropertyAddress</process:parameterType>
           </process:Input>
   </process:hasInput>

   <process:hasInput>
           <process:Input rdf:ID="BuyGenericItemPackagingSelection">
                   <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#
PackagingType</process:parameterType>
           </process:Input>
   </process:hasInput>

   <process:hasInput>
           <process:Input rdf:ID="BuyGenericItemDeliveryTypeSelection">
                   <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#DeliveryType
</process:parameterType>
           </process:Input>
   </process:hasInput>

   <process:hasOutput>
           <process:Output rdf:ID="BuyGenericItemShipmentDetails">
           <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#Shipment</
process:parameterType>
       </process:Output>
   </process:hasOutput>

   <process:hasOutput>
           <process:Output rdf:ID="BuyGenericItemAcctID">
           <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#AcctID</
process:parameterType>
       </process:Output>
   </process:hasOutput>


   <process:composedOf>
           <process:sequence>
```

```
                    <process:components>
                            <process:ControlConstructList>
                                    <list:first>
                                            <process:Perform rdf:ID="LocateItemPerform">
                                                    <process:process rdf:resource="LocateItem"
 />
                                                    <process:hasDataFrom>
                                                            <process:InputBinding>
                                                                    <process:toParam
rdf:resource="#LocateItemSearch" />
                                                                    <process:valueSource>
                                                                            <process:ValueOf>
                                                                                    <
process:theVar rdf:resource="#BuyGenericItemSearchTerm" />
                                                                                    <
process:fromProcess rdf:resource="&process;#TheParentPerform" />
                                                                            </process:ValueOf>
                                                                    </process:valueSource>
                                                            </process:InputBinding>
                                                    </process:hasDataFrom>
                                            </process:Perform>
                                    </list:first>
                                    <list:rest>
                                            <process:Perform rdf:ID="BuyItemPerform">
                                                    <process:process rdf:resource="#BuyItem" /
>
                                                    <process:hasDataFrom>
                                                            <process:InputBinding>
                                                                    <process:toParam
rdf:resource="#BuyItemItem" />
                                                                    <process:valueSource>
                                                                            <process:ValueOf>
                                                                                    <
process:theVar rdf:resource="#LocateItemResult" />
                                                                                    <
process:fromProcess rdf:resource="&THIS;#LocateItemPerform" />
                                                                            </process:ValueOf>
                                                                    </process:valueSource>
                                                            </process:InputBinding>
                                                    </process:hasDataFrom>
                                                    <process:hasDataFrom>
                                                            <process:InputBinding>
                                                                    <process:toParam
rdf:resource="#BuyItemCreateAccInfo" />
                                                                    <process:valueSource>
                                                                            <process:ValueOf>
                                                                                    <
process:theVar rdf:resource="#BuyGenericItemCreateAccInfo" />
                                                                                    <
process:fromProcess rdf:resource="&process;#TheParentPerform" />
                                                                            </process:ValueOf>
                                                                    </process:valueSource>
                                                            </process:InputBinding>
                                                    </process:hasDataFrom>
                                                    <process:hasDataFrom>
                                                            <process:InputBinding>
                                                                    <process:toParam
rdf:resource="#BuyItemSignInData" />
                                                                    <process:valueSource>
```

```
                                                                        <process:ValueOf>
                                                                                <
process:theVar rdf:resource="#BuyGenericItemSignInData" />
                                                                                <
process:fromProcess rdf:resource="&process;#TheParentPerform" />
                                                                        </process:ValueOf>
                                                                </process:valueSource>
                                                        </process:InputBinding>
                                                </process:hasDataFrom>
                                                <process:hasDataFrom>
                                                        <process:InputBinding>
                                                                <process:toParam
rdf:resource="#BuyItemCreditCard" />
                                                                <process:valueSource>
                                                                        <process:ValueOf>
                                                                                <
process:theVar rdf:resource="#BuyGenericItemCreditCard" />
                                                                                <
process:fromProcess rdf:resource="&process;#TheParentPerform" />
                                                                        </process:ValueOf>
                                                                </process:valueSource>
                                                        </process:InputBinding>
                                                </process:hasDataFrom>
                                                <process:hasDataFrom>
                                                        <process:InputBinding>
                                                                <process:toParam
rdf:resource="#BuyItemDeliveryAddress" />
                                                                <process:valueSource>
                                                                        <process:ValueOf>
                                                                                <
process:theVar rdf:resource="#BuyGenericItemDeliveryAddress" />
                                                                                <
process:fromProcess rdf:resource="&process;#TheParentPerform" />
                                                                        </process:ValueOf>
                                                                </process:valueSource>
                                                        </process:InputBinding>
                                                </process:hasDataFrom>
                                                <process:hasDataFrom>
                                                        <process:InputBinding>
                                                                <process:toParam
rdf:resource="#BuyItemPackagingSelection" />
                                                                <process:valueSource>
                                                                        <process:ValueOf>
                                                                                <
process:theVar rdf:resource="#BuyGenericItemPackagingSelection" />
                                                                                <
process:fromProcess rdf:resource="&process;#TheParentPerform" />
                                                                        </process:ValueOf>
                                                                </process:valueSource>
                                                        </process:InputBinding>
                                                </process:hasDataFrom>
                                                <process:hasDataFrom>
                                                        <process:InputBinding>
                                                                <process:toParam
rdf:resource="#BuyItemDeliveryTypeSelection" />
                                                                <process:valueSource>
                                                                        <process:ValueOf>
                                                                                <
process:theVar rdf:resource="#BuyGenericItemDeliveryTypeSelection" />
```

```
                                                                                    <
    process:fromProcess rdf:resource="&process;#TheParentPerform" />
                                                                                  </process:ValueOf>
                                                                              </process:valueSource>
                                                                          </process:InputBinding>
                                                                      </process:hasDataFrom>
                                                                  </process:Perform>
                                                              </list:rest>
                                                          </process:ControlConstructList>
                                                      </process:components>
                                                  </process:sequence>
                                              </process:composedOf>


                                              <process:hasResult>
                                            <process:Result>
                                              <process:inCondition rdf:resource="&expr;#AlwaysTrue"/>
                                              <process:withOutput>
                                                <process:OutputBinding>
                                                  <process:toParam rdf:resource="#BuyGenericItemShipmentDetails"/>
                                                  <process:valueSource>
                                                    <process:ValueOf>
                                                      <process:theVar rdf:resource="#BuyItemShipmentDetails"/>
                                                      <process:fromProcess rdf:resource="#BuyItemPerform"/>
                                                    </process:ValueOf>
                                                  </process:valueSource>
                                                </process:OutputBinding>
                                              </process:withOutput>
                                            </process:Result>
                                          </process:hasResult>

                                              <process:hasResult>
                                            <process:Result>
                                              <process:inCondition rdf:resource="&expr;#AlwaysTrue"/>
                                              <process:withOutput>
                                                <process:OutputBinding>
                                                  <process:toParam rdf:resource="#BuyGenericItemAcctID"/>
                                                  <process:valueSource>
                                                    <process:ValueOf>
                                                      <process:theVar rdf:resource="#BuyItemAcctID"/>
                                                      <process:fromProcess rdf:resource="#BuyItemPerform"/>
                                                    </process:ValueOf>
                                                  </process:valueSource>
                                                </process:OutputBinding>
                                              </process:withOutput>
                                            </process:Result>
                                          </process:hasResult>

                                        </process:CompositeProcess>

                                        <!--
                                        ======================================================
                                              CompositeProcess: BuyItem
                                        ======================================================
                                        -->

                                        <process:CompositeProcess rdf:ID="BuyItem">
                                              <rdfs:comment>
                                              </rdfs:comment>
```

```
    <process:hasInput>
            <process:Input rdf:ID="BuyItemItem">
                    <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#Item</
process:parameterType>
            </process:Input>
    </process:hasInput>

    <process:hasInput>
            <process:Input rdf:ID="BuyItemCreateAccInfo">
                    <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#AcctInfo</
process:parameterType>
            </process:Input>
    </process:hasInput>

    <process:hasInput>
            <process:Input rdf:ID="BuyItemSignInData">
                    <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#SignInData</
process:parameterType>
            </process:Input>
    </process:hasInput>

    <process:hasInput>
            <process:Input rdf:ID="BuyItemCreditCard">
                    <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#CreditCard</
process:parameterType>
            </process:Input>
    </process:hasInput>

    <process:hasInput>
            <process:Input rdf:ID="BuyItemDeliveryAddress">
                    <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#
PropertyAddress</process:parameterType>
            </process:Input>
    </process:hasInput>

    <process:hasInput>
            <process:Input rdf:ID="BuyItemPackagingSelection">
                    <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#
PackagingType</process:parameterType>
            </process:Input>
    </process:hasInput>

    <process:hasInput>
            <process:Input rdf:ID="BuyItemDeliveryTypeSelection">
                    <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#DeliveryType
</process:parameterType>
            </process:Input>
    </process:hasInput>

    <process:hasOutput>
            <process:Output rdf:ID="BuyItemShipmentDetails">
            <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#Shipment</
process:parameterType>
        </process:Output>
    </process:hasOutput>

    <process:hasOutput>
            <process:Output rdf:ID="BuyItemAcctID">
```

```
          <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#AcctID</
process:parameterType>
       </process:Output>
   </process:hasOutput>


   <process:composedOf>
          <process:sequence>
                 <process:components>
                        <process:ControlConstructList>
                               <list:first>
                                      <process:Perform rdf:ID="BuySequencePerform">
                                             <process:process rdf:resource="BuySequence
" />
                                             <process:hasDataFrom>
                                                    <process:InputBinding>
                                                           <process:toParam
rdf:resource="#BuySequenceItem" />
                                                           <process:valueSource>
                                                                  <process:ValueOf>
                                                                         <
process:theVar rdf:resource="#BuyItemItem" />
                                                                         <
process:fromProcess rdf:resource="&process;#TheParentPerform" />
                                                                  </process:ValueOf>
                                                           </process:valueSource>
                                                    </process:InputBinding>
                                             </process:hasDataFrom>
                                             <process:hasDataFrom>
                                                    <process:InputBinding>
                                                           <process:toParam
rdf:resource="#BuySequenceAccInfo" />
                                                           <process:valueSource>
                                                                  <process:ValueOf>
                                                                         <
process:theVar rdf:resource="#BuyItemAccInfo" />
                                                                         <
process:fromProcess rdf:resource="&process;#TheParentPerform" />
                                                                  </process:ValueOf>
                                                           </process:valueSource>
                                                    </process:InputBinding>
                                             </process:hasDataFrom>
                                             <process:hasDataFrom>
                                                    <process:InputBinding>
                                                           <process:toParam
rdf:resource="#BuySequenceSignInData" />
                                                           <process:valueSource>
                                                                  <process:ValueOf>
                                                                         <
process:theVar rdf:resource="#BuyItemSignInData" />
                                                                         <
process:fromProcess rdf:resource="&process;#TheParentPerform" />
                                                                  </process:ValueOf>
                                                           </process:valueSource>
                                                    </process:InputBinding>
                                             </process:hasDataFrom>
                                             <process:hasDataFrom>
                                                    <process:InputBinding>
```

```
                                                                <process:toParam
rdf:resource="#BuySequenceCreditCard" />
                                                                <process:valueSource>
                                                                        <process:ValueOf>
                                                                                <
process:theVar rdf:resource="#BuyItemCreditCard" />
                                                                                <
process:fromProcess rdf:resource="&process;#TheParentPerform" />
                                                                        </process:ValueOf>
                                                                </process:valueSource>
                                                        </process:InputBinding>
                                                </process:hasDataFrom>
                                        </process:Perform>
                                </list:first>
                                <list:rest>
                                        <process:ControlConstructList>
                                                <list:first>
                                                        <process:Perform rdf:ID="
SpecifyDeliveryDetailsPerform">
                                                                <process:process
rdf:resource="#SpecifyDeliveryDetails" />
                                                                <process:hasDataFrom>
                                                                        <
process:InputBinding>
                                                                                <
process:toParam rdf:resource="#DeliveryAddress" />
                                                                                <
process:valueSource>
                                                                                        <
process:ValueOf>

        <process:theVar rdf:resource="#BuyItemDeliveryAddress" />

        <process:fromProcess rdf:resource="&process;#TheParentPerform" />
                                                                                        </
process:ValueOf>
                                                                                </
process:valueSource>
                                                                        </
process:InputBinding>
                                                                </process:hasDataFrom>
                                                                <process:hasDataFrom>
                                                                        <
process:InputBinding>
                                                                                <
process:toParam rdf:resource="#PackagingSelection" />
                                                                                <
process:valueSource>
                                                                                        <
process:ValueOf>

        <process:theVar rdf:resource="#BuyItemPackagingSelection" />

        <process:fromProcess rdf:resource="&process;#TheParentPerform" />
                                                                                        </
process:ValueOf>
                                                                                </
process:valueSource>
```

```
                                                                    </
process:InputBinding>
                                                                </process:hasDataFrom>
                                                                <process:hasDataFrom>
                                                                    <
process:InputBinding>
                                                                        <
process:toParam rdf:resource="#DeliveryTypeSelection" />
                                                                        <
process:valueSource>
                                                                            <
process:ValueOf>

        <process:theVar rdf:resource="#BuyItemDeliveryTypeSelection" />

        <process:fromProcess rdf:resource="&process;#TheParentPerform" />
                                                                            </
process:ValueOf>
                                                                        </
process:valueSource>
                                                                    </
process:InputBinding>
                                                                </process:hasDataFrom>
                                                            </process:Perform>
                                                        </list:first>
                                                        <list:rest>
                                                            <process:ControlConstructList>
                                                                <list:first>
                                                                    <process:Perform
rdf:ID="FinaliseOrderPerform">
                                                                        <
process:process rdf:resource="#FinaliseOrder" />
                                                                        <
process:hasDataFrom>
                                                                            <
process:InputBinding>

        <process:toParam rdf:resource="#FinaliseOrderAcctID" />

        <process:valueSource>

                <process:ValueOf>

                        <process:theVar rdf:resource="#BuySequenceAcctID" />

                        <process:fromProcess rdf:resource="&THIS;#BuySequencePerform" />

                </process:ValueOf>

        </process:valueSource>
                                                                            </
process:InputBinding>
                                                                        </
process:hasDataFrom>
                                                                    </process:Perform>
                                                                </list:first>
                                                                <list:rest rdf:resource="&
list;#nil"/>
                                                            </process:ControlConstructList>
```

```
                                                    </list:rest>
                                                </process:ControlConstructList>
                                            </list:rest>
                                        </process:ControlConstructList>
                                    </process:components>
                                </process:sequence>
                            </process:composedOf>


                            <process:hasResult>
                        <process:Result>
                            <process:inCondition rdf:resource="&expr;#AlwaysTrue"/>
                            <process:withOutput>
                                <process:OutputBinding>
                                    <process:toParam rdf:resource="#BuyItemShipmentDetails"/>
                                    <process:valueSource>
                                        <process:ValueOf>
                                            <process:theVar rdf:resource="#ShipmentDetails"/>
                                            <process:fromProcess rdf:resource="#FinaliseOrderPerform"/>
                                        </process:ValueOf>
                                    </process:valueSource>
                                </process:OutputBinding>
                            </process:withOutput>
                        </process:Result>
                    </process:hasResult>

                            <process:hasResult>
                        <process:Result>
                            <process:inCondition rdf:resource="&expr;#AlwaysTrue"/>
                            <process:withOutput>
                                <process:OutputBinding>
                                    <process:toParam rdf:resource="#BuyItemAcctID"/>
                                    <process:valueSource>
                                        <process:ValueOf>
                                            <process:theVar rdf:resource="#BuySequenceAcctID"/>
                                            <process:fromProcess rdf:resource="#BuySequencePerform"/>
                                        </process:ValueOf>
                                    </process:valueSource>
                                </process:OutputBinding>
                            </process:withOutput>
                        </process:Result>
                    </process:hasResult>

                </process:CompositeProcess>


                <!--
                ========================================================
                      CompositeProcess: BuySequence
                ========================================================
                -->

                <process:CompositeProcess rdf:ID="BuySequence">
                      <rdfs:comment>
                      </rdfs:comment>

                      <process:hasInput>
                              <process:Input rdf:ID="BuySequenceItem">
```

```
                <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#SearchTerm</
process:parameterType>
            </process:Input>
   </process:hasInput>

   <process:hasInput>
            <process:Input rdf:ID="BuySequenceCreateAccInfo">
                <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#AcctInfo</
process:parameterType>
            </process:Input>
   </process:hasInput>

   <process:hasInput>
            <process:Input rdf:ID="BuySequenceSignInData">
                <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#SignInData</
process:parameterType>
            </process:Input>
   </process:hasInput>

   <process:hasInput>
            <process:Input rdf:ID="BuySequenceCreditCard">
                <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#CreditCard</
process:parameterType>
            </process:Input>
   </process:hasInput>

   <process:hasOutput>
            <process:Output rdf:ID="BuySequenceAcctID">
            <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#AcctID</
process:parameterType>
        </process:Output>
   </process:hasOutput>


   <process:composedOf>
            <process:sequence>
                <process:components>
                    <process:ControlConstructList>
                        <list:first>
                            <process:Perform rdf:ID="PutInCartPerform">
                                <process:process rdf:resource="#PutInCart"
/>
                                <process:hasDataFrom>
                                    <process:InputBinding>
                                        <process:toParam
rdf:resource="#PutInCartItem"/>
                                            <
process:valueSource>
                                            <process:ValueOf>
                                                <
process:theVar rdf:resource="#BuySequenceItem"/>
                                                <
process:fromProcess rdf:resource="&process;#TheParentPerform"/>
                                            </process:ValueOf>
                                        </process:valueSource>
                                    </process:InputBinding>
                                </process:hasDataFrom>
                            </process:Perform>
                        </list:first>
```

```
                                <list:rest>
                                        <process:ControlConstructList>
                                                <list:first>
                                                        <process:Perform rdf:ID="
SignInAlternativesPerform">
                                        <process:process rdf:resource="#SignInAlternatives
"/>
                                                <process:hasDataFrom>
                                                <process:InputBinding>
                                                        <process:toParam rdf:resource="#
SignInAlternativesCreateAcctInfo"/>
                                                        <process:valueSource>
                                                                <process:ValueOf>
                                                                        <process:theVar
rdf:resource="#BuySequenceCreateAcctInfo"/>
                                                                        <
process:fromProcess rdf:resource="&process;#TheParentPerform"/>
                                                                </process:ValueOf>
                                                                </process:valueSource>
                                                        </process:InputBinding>
                                                        </process:hasDataFrom>
                                                <process:hasDataFrom>
                                                        <process:InputBinding>
                                                        <process:toParam rdf:resource="#
SignInAlternativesSignInData"/>
                                                        <process:valueSource>
                                                                <process:ValueOf>
                                                                        <process:theVar
rdf:resource="#BuySequenceSignInData"/>
                                                                        <
process:fromProcess rdf:resource="&process;#TheParentPerform"/>
                                                                </process:ValueOf>
                                                        </process:valueSource>
                                                </process:InputBinding>
                                                </process:hasDataFrom>
                                        </process:Perform>
                                                </list:first>
                                                <list:rest>
                                                        <process:ControlConstructList>
                                                                <list:first>
                                                                        <process:Perform
rdf:ID="SpecifyPaymentMethodPerform">
                                                                                <
process:process rdf:resource="#SpecifyPaymentMethod" />
                                                                                <
process:hasDataFrom>
                                                                                        <
process:InputBinding>
        <process:toParam rdf:resource="#SpecifyPaymentMethodCreditCard" />
        <process:valueSource>
                <process:ValueOf>
                        <process:theVar rdf:resource="#BuySequenceCreditCard" />
                        <process:fromProcess rdf:resource="&process;#TheParentPerform" />
```

```
                </process:ValueOf>

          </process:valueSource>
                                                                                </
   process:InputBinding>
                                                                              </
   process:hasDataFrom>
                                                                        </process:Perform>
                                                                    </list:first>
                                                                    <list:rest rdf:resource="&
   list;#nil"/>
                                                              </process:ControlConstructList>
                                                         </list:rest>
                                                    </process:ControlConstructList>
                                               </list:rest>
                                          </process:ControlConstructList>
                                     </process:components>
                </process:sequence>
      </process:composedOf>

      <process:hasResult>
    <process:Result>
      <process:inCondition rdf:resource="&expr;#AlwaysTrue"/>
      <process:withOutput>
        <process:OutputBinding>
          <process:toParam rdf:resource="#BuySequenceAcctID"/>
          <process:valueSource>
            <process:ValueOf>
              <process:theVar rdf:resource="#SignInAlternativesAcctID"/>
              <process:fromProcess rdf:resource="#SignInAlternativesPerform"/>
            </process:ValueOf>
          </process:valueSource>
        </process:OutputBinding>
      </process:withOutput>
    </process:Result>
  </process:hasResult>

</process:CompositeProcess>

<!--
=======================================================
      CompositeProcess: SignInAlternatives
=======================================================
-->

<process:CompositeProcess rdf:ID="SignInAlternatives">
      <rdfs:comment>
              TODO
      </rdfs:comment>

      <process:hasInput>
      <process:Input rdf:ID="SignInAlternativesCreateAcctInfo">
              <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#AcctInfo</
   process:parameterType>
      </process:Input>
      </process:hasInput>

      <process:hasInput>
```

```
   <process:Input rdf:ID="SignInAlternativesSignInData">
           <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#SignInData</
process:parameterType>
   </process:Input>
   </process:hasInput>

   <process:hasOutput>
   <process:Output rdf:ID="SignInAlternativesAcctID">
           <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#AcctID</
process:parameterType>
   </process:Output>
   </process:hasOutput>

   <process:composedOf>
   <process:Choice>
           <process:components>
                   <process:ControlConstructBag>
                           <list:first>
                           <process:Perform rdf:ID="CreateAcctSequencePerform">
                                   <process:process rdf:resource="#CreateAcctSequence"/>
                                   <process:hasDataFrom>
                                           <process:InputBinding>
                                                   <process:toParam rdf:resource="#
CreateAcctSequenceCreateAcctInfo"/>
                                                   <process:valueSource>
                                                   <process:ValueOf>
                                                           <process:theVar rdf:resource="#
SignInAlternativesCreateAcctInfo"/>
                                                           <process:fromProcess rdf:resource=
"&process;#TheParentPerform"/>
                                                   </process:ValueOf>
                                                   </process:valueSource>
                                           </process:InputBinding>
                                   </process:hasDataFrom>
                           </process:Perform>
                           </list:first>
                           <list:rest>
                           <process:ControlConstructBag>
                                   <list:first>
                                           <process:Perform rdf:ID="SignInSequencePerform">
                                                   <process:process rdf:resource="#
SignInSequence"/>
                                                   <process:hasDataFrom>
                                                   <process:InputBinding>
                                                           <process:toParam rdf:resource="#
SignInSequenceSignInInfo"/>
                                                           <process:valueSource>
                                                                   <process:ValueOf>
                                                                           <process:theVar
rdf:resource="#SignInAlternativesSignInData"/>
                                                                           <
process:fromProcess rdf:resource="&process;#TheParentPerform"/>
                                                                   </process:ValueOf>
                                                           </process:valueSource>
                                                   </process:InputBinding>
                                                   </process:hasDataFrom>
                                           </process:Perform>
                                   </list:first>
                                   <list:rest rdf:resource="&list;#nil"/>
```

```
                        </process:ControlConstructBag>
                        </list:rest>
                </process:ControlConstructBag>
        </process:components>
    </process:Choice>
</process:composedOf>

<process:hasResult>
<process:Result>
        <process:inCondition>
                <expr:SWRL-Condition rdf:ID="SignInAlternativesAcctExists">
                        <rdfs:label>
                                hasAcctID(SignInAlternativesSignInData,
SignInAlternativesAcctID)
                        </rdfs:label>
                        <rdfs:comment>
                        If an account already exists, sign-in operation will be
                        performed and returned acct ID willbe used
                        </rdfs:comment>
                        <expr:expressionLanguage rdf:resource="&expr;#SWRL"/>
                        <expr:expressionObject>
                        <swrl:AtomList>
                                <rdf:first>
                                        <swrl:IndividualPropertyAtom>
                                                <swrl:propertyPredicate rdf:resource="#
hasAcctID"/>
                                                <swrl:argument1 rdf:resource="#
SignInAlternativesSignInData"/>
                                                <swrl:argument2 rdf:resource="#
SignInAlternativesAcctID"/>
                                        </swrl:IndividualPropertyAtom>
                                </rdf:first>
                                <rdf:rest rdf:resource="&rdf;#nil"/>
                        </swrl:AtomList>
                        </expr:expressionObject>
                </expr:SWRL-Condition>
        </process:inCondition>
        <process:withOutput>
                <process:OutputBinding>
                        <process:toParam rdf:resource="#SignInAlternativesAcctID"/>
                        <process:valueSource>
                        <process:ValueOf>
                                <process:theVar rdf:resource="#SignInSequenceAcctID"/>
                                <process:fromProcess rdf:resource="#SignInSequencePerform"
/>
                        </process:ValueOf>
                        </process:valueSource>
                </process:OutputBinding>
        </process:withOutput>
</process:Result>
</process:hasResult>

<process:hasResult>
<process:Result>
        <process:inCondition>
                <expr:SWRL-Condition rdf:ID="SignInAlternativesNoAcctExists">
                        <rdfs:comment>
                        If an account does not exist a new account will be
                        create by the CreateAcct process and the ID of new
```

```
                                account will be used.
                                </rdfs:comment>
                                <expr:expressionLanguage rdf:resource="&expr;#SWRL"/>
                                <expr:expressionObject>
                                <swrl:AtomList>
                                        <rdf:first>
                                                <swrl:ClassAtom>
                                                        <swrl:classPredicate rdf:resource="#
   NonExistingAcct"/>
                                                        <swrl:argument1 rdf:resource="#
   SignInAlternativesSignInData"/>
                                                </swrl:ClassAtom>
                                        </rdf:first>
                                        <rdf:rest rdf:resource="&rdf;#nil"/>
                                </swrl:AtomList>
                                </expr:expressionObject>
                        </expr:SWRL-Condition>
                </process:inCondition>
                <process:withOutput>
                        <process:OutputBinding>
                                <process:toParam rdf:resource="#SignInAlternativesAcctID"/>
                                <process:valueSource>
                                <process:ValueOf>
                                        <process:theVar rdf:resource="#
   CreateAcctSequenceCreateAcctOutput"/>
                                        <process:fromProcess rdf:resource="#
   CreateAcctSequencePerform"/>
                                </process:ValueOf>
                                </process:valueSource>
                        </process:OutputBinding>
                </process:withOutput>
        </process:Result>
        </process:hasResult>
</process:CompositeProcess>



<!--
=======================================================
      CompositeProcess: SignInSequence
=======================================================
-->

<process:CompositeProcess rdf:ID="SignInSequence">
      <rdfs:comment>
      SignInSequence performs an atomic process SignIn,
      followed by an atomic process LoadUserProfile.
      </rdfs:comment>

      <process:hasInput>
      <process:Input rdf:ID="SignInSequenceSignInInfo">
              <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#SignInData</
   process:parameterType>
      </process:Input>
      </process:hasInput>

      <process:hasOutput>
      <process:Output rdf:ID="SignInSequenceAcctID">
```

```
            <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#AcctID</
   process:parameterType>
      </process:Output>
      </process:hasOutput>

      <process:composedOf>
      <process:Sequence>
              <process:components>
                      <process:ControlConstructList>
                              <list:first>
                              <process:Perform rdf:ID="SignInPerform">
                                      <process:process rdf:resource="#SignIn"/>
                                      <process:hasDataFrom>
                                              <process:InputBinding>
                                                      <process:toParam rdf:resource="#SignInInfo
   "/>
                                                      <process:valueSource>
                                                      <process:ValueOf>
                                                              <process:theVar rdf:resource="#
   SignInSequenceSignInInfo"/>
                                                              <process:fromProcess rdf:resource=
   "&process;#TheParentPerform"/>
                                                      </process:ValueOf>
                                                      </process:valueSource>
                                              </process:InputBinding>
                                      </process:hasDataFrom>
                              </process:Perform>
                              </list:first>
                              <list:rest>
                              <process:ControlConstructList>
                                      <list:first>
                                              <process:Perform rdf:ID="LoadUserProfilePerform">
                                                      <process:process rdf:resource="#
   LoadUserProfile"/>
                                              </process:Perform>
                                      </list:first>
                                      <list:rest rdf:resource="&list;#nil"/>
                              </process:ControlConstructList>
                              </list:rest>
                      </process:ControlConstructList>
              </process:components>
      </process:Sequence>
      </process:composedOf>
</process:CompositeProcess>


<!--
=======================================================
      CompositeProcess: CreateAcctSequence
=======================================================
-->

<process:CompositeProcess rdf:ID="CreateAcctSequence">
      <rdfs:comment>
      CreateAcctSequence performs atomic process
      CreateAcct followed by atomic process LoadUserProfile.
      </rdfs:comment>

      <process:hasInput>
```

```
   <process:Input rdf:ID="CreateAcctSequenceCreateAcctInfo">
           <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#AcctInfo</
process:parameterType>
   </process:Input>
   </process:hasInput>

   <process:hasOutput>
   <process:Output rdf:ID="CreateAcctSequenceCreateAcctOutput">
           <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#AcctID</
process:parameterType>
   </process:Output>
   </process:hasOutput>

   <process:composedOf>
   <process:Sequence>
           <process:components>
                   <process:ControlConstructList>
                           <list:first>
                           <process:Perform rdf:ID="CreateAcctPerform">
                                   <process:process rdf:resource="#CreateAcct"/>
                                   <process:hasDataFrom>
                                           <process:InputBinding>
                                                   <process:toParam rdf:resource="#
CreateAcctInfo"/>
                                                   <process:valueSource>
                                                   <process:ValueOf>
                                                           <process:theVar rdf:resource="#
CreateAcctSequenceCreateAcctInfo"/>
                                                           <process:fromProcess rdf:resource=
"&process;#TheParentPerform"/>
                                                   </process:ValueOf>
                                                   </process:valueSource>
                                           </process:InputBinding>
                                   </process:hasDataFrom>
                           </process:Perform>
                           </list:first>
                           <list:rest>
                           <process:ControlConstructList>
                                   <list:first>
                                           <process:Perform rdf:ID="LoadUserProfilePerform1">
                                                   <process:process rdf:resource="#
LoadUserProfile"/>
                                           </process:Perform>
                                   </list:first>
                                   <list:rest rdf:resource="&list;#nil"/>
                           </process:ControlConstructList>
                           </list:rest>
                   </process:ControlConstructList>
           </process:components>
   </process:Sequence>
   </process:composedOf>
   <process:hasResult>
   <process:Result>
           <process:inCondition rdf:resource="&expr;#AlwaysTrue"/>
           <process:withOutput>
                   <process:OutputBinding>
                           <process:toParam rdf:resource="#CreateAcctSequenceCreateAcctOutput
"/>
                           <process:valueSource>
```

```
                                <process:ValueOf>
                                        <process:theVar rdf:resource="#CreateAcctOutput"/>
                                        <process:fromProcess rdf:resource="#CreateAcctPerform"/>
                                </process:ValueOf>
                                </process:valueSource>
                        </process:OutputBinding>
                </process:withOutput>
        </process:Result>
        </process:hasResult>
</process:CompositeProcess>

<!--
=======================================================
        AtomicProcess: LocateItem
=======================================================
-->

<process:AtomicProcess rdf:ID="LocateItem">
        <process:hasInput>
                <process:Input rdf:ID="LocateItemSearch">
                        <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#SearchTerm</
   process:parameterType>
                </process:Input>
        </process:hasInput>

        <process:hasOutput>
                <process:Output rdf:ID="LocateItemResult">
                        <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#Item</
   process:parameterType>
                </process:Output>
        </process:hasOutput>
</process:AtomicProcess>

<!--
=======================================================
        AtomicProcess: PutInCart
=======================================================
-->

<process:AtomicProcess rdf:ID="PutInCart">
        <process:hasInput>
                <process:Input rdf:ID="PutInCartItem">
                        <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#Item</
   process:parameterType>
                </process:Input>
        </process:hasInput>
</process:AtomicProcess>


<!--
=======================================================
        AtomicProcess: SignIn
=======================================================
-->

<process:AtomicProcess rdf:ID="SignIn">
        <rdfs:comment>
        Sign in is a process that requires input of signin info.
        </rdfs:comment>
```

```
      <process:hasInput>
      <process:Input rdf:ID="SignInInfo">
              <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#SignInData</
   process:parameterType>
              </process:Input>
      </process:hasInput>
</process:AtomicProcess>


<!--
=======================================================
      AtomicProcess: CreateAcct
=======================================================
-->

<process:AtomicProcess rdf:ID="CreateAcct">
      <process:hasInput>
      <process:Input rdf:ID="CreateAcctInfo">
              <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#AcctInfo</
   process:parameterType>
      </process:Input>
      </process:hasInput>

      <process:hasOutput>
      <process:Output rdf:ID="CreateAcctOutput">
              <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#AcctID</
   process:parameterType>
      </process:Output>
      </process:hasOutput>
</process:AtomicProcess>


<!--
=======================================================
      AtomicProcess: LoadUserProfile
=======================================================
-->

<process:AtomicProcess rdf:ID="LoadUserProfile">
      <rdfs:comment>
      LoadUserProfile can only be invoked if a user profile already exists.
      </rdfs:comment>
</process:AtomicProcess>

<!--
=======================================================
      AtomicProcess: SpecifyPaymentMethod
=======================================================
-->

<process:AtomicProcess rdf:ID="SpecifyPaymentMethod">
      <process:hasInput>
              <process:Input rdf:ID="SpecifyPaymentMethodCreditCard">
                      <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#CreditCard</
   process:parameterType>
              </process:Input>
      </process:hasInput>
</process:AtomicProcess>
```

```
  <!--
  ======================================================
        AtomicProcess: SpecifyDeliveryDetails
  ======================================================
  -->

  <process:AtomicProcess rdf:ID="SpecifyDeliveryDetails">
        <process:hasInput>
                <process:Input rdf:ID="DeliveryAddress">
                        <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#
     PropertyAddress</process:parameterType>
                </process:Input>
        </process:hasInput>
        <process:hasInput>
                <process:Input rdf:ID="PackagingSelection">
                        <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#
     PackagingType</process:parameterType>
                </process:Input>
        </process:hasInput>
        <process:hasInput>
                <process:Input rdf:ID="DeliveryTypeSelection">
                        <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#DeliveryType
     </process:parameterType>
                </process:Input>
        </process:hasInput>
  </process:AtomicProcess>

  <!--
  ======================================================
        AtomicProcess: FinaliseOrder
  ======================================================
  -->

  <process:AtomicProcess rdf:ID="FinaliseOrder">
        <process:hasInput>
                <process:Input rdf:ID="FinaliseOrderAcctID">
                        <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#AcctID</
     process:parameterType>
                </process:Input>
        </process:hasInput>
        <process:hasOutput>
        <process:Output rdf:ID="ShipmentDetails">
                <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#Shipment</
     process:parameterType>
        </process:Output>
        </process:hasOutput>
  </process:AtomicProcess>


</rdf:RDF>
```

A.5.6 BuyHouse.owls

```
<?xml version="1.0" encoding="WINDOWS-1252"?>
<!DOCTYPE uridef [
  <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema">
```

```
  <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns">
  <!ENTITY owl "http://www.w3.org/2002/07/owl">
  <!ENTITY xsd "http://www.w3.org/2001/XMLSchema">
  <!ENTITY service "http://www.daml.org/services/owl-s/1.2/Service.owl">
  <!ENTITY profile "http://www.daml.org/services/owl-s/1.2/Profile.owl">
  <!ENTITY process "http://www.daml.org/services/owl-s/1.2/Process.owl">
  <!ENTITY grounding "http://www.daml.org/services/owl-s/1.2/Grounding.owl">
  <!ENTITY list "http://www.daml.org/services/owl-s/1.2/generic/ObjectList.owl">
  <!ENTITY swrl "http://www.w3.org/2003/11/swrl">
  <!ENTITY swrlb "http://www.w3.org/2003/11/swrlb">
  <!ENTITY expr "http://www.daml.org/services/owl-s/1.2/generic/Expression.owl">
  <!ENTITY concepts "http://testcases.freya.local/ontology/concepts.owl">
  <!ENTITY THIS "http://testcases.freya.local/purchasing/buyHouse.owls">
]>

<rdf:RDF
        xmlns:rdf="&rdf;#"
        xmlns:xsd="&xsd;#"
        xmlns:rdfs="&rdfs;#"
        xmlns:owl="&owl;#"
        xmlns:expr="&expr;#"
        xmlns:service="&service;#"
        xmlns:grounding="&grounding;#"
    xmlns:process="&process;#"
        xmlns:profile="&profile;#"
    xmlns:list="&list;#"
    xmlns:swrl="&swrl;#"
    xmlns:swrlb="&swrlb;#"
        xmlns:concepts="&concepts;#"
    xmlns="&THIS;#"
        xml:base="&THIS;#">

  <owl:Ontology rdf:about="">
        <owl:versionInfo>
                $ID: buyHouse.owls, v 1.4 2011/02/28 akd07r $
        </owl:versionInfo>
        <rdfs:comment>
                TODO
        </rdfs:comment>
        <owl:imports rdf:resource="&service;" />
        <owl:imports rdf:resource="&process;" />
        <owl:imports rdf:resource="&profile;" />
        <owl:imports rdf:resource="&grounding;" />
        <owl:imports rdf:resource="&concepts;" />
        <owl:imports rdf:resource="http://127.0.0.1/ontology/portal.owl" />
        <owl:imports rdf:resource="http://127.0.0.1/ontology/support.owl" />
  </owl:Ontology>

  <!--
  ================================================
  ================================================
        Service Model
  ================================================
  ================================================
  -->
  <service:Service rdf:ID="BuyHouseService">
        <service:presents rdf:resource="#BuyHouseProfile"/>
        <service:describedBy rdf:resource="#BuyHouse"/>
        <!--<service:supports rdf:resource="#BuyHouseServiceGrounding"/>-->
```

```
</service:Service>

<!--
=======================================================
=======================================================
      Profile Model
=======================================================
=======================================================
-->
<profile:Profile rdf:ID="BuyHouseProfile">
      <profile:serviceName>
              BuyHouse_Service
      </profile:serviceName>
      <profile:textDescription>
      TODO
      </profile:textDescription>

      <service:presentedBy rdf:resource="#BuyHouseService"/>
      <profile:has_process rdf:resource="#BuyHouse"/>

      <profile:hasInput rdf:resource="#BuyHouseSearchTerm"/>
      <profile:hasInput rdf:resource="#BuyHousePropertyOffer"/>
      <profile:hasInput rdf:resource="#BuyHouseFundingInfo"/>

      <profile:hasOutput rdf:resource="#BuyHousePropertyListing"/>
      <profile:hasOutput rdf:resource="#BuyHouseReceipt"/>
      <profile:hasOutput rdf:resource="#BuyHouseDeeds"/>
</profile:Profile>

<!--
=======================================================
=======================================================
      Process Model
=======================================================
=======================================================
-->

<!--
=======================================================
      CompositeProcess: BuyHouse
=======================================================
-->

<process:CompositeProcess rdf:ID="BuyHouse">
      <process:hasInput>
      <process:Input rdf:ID="BuyHouseSearchTerm">
              <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#SearchTerm</
  process:parameterType>
      </process:Input>
      </process:hasInput>

      <process:hasInput>
      <process:Input rdf:ID="BuyHousePropertyOffer">
              <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#PropertyOffer</
  process:parameterType>
      </process:Input>
      </process:hasInput>

      <process:hasInput>
```

```
   <process:Input rdf:ID="BuyHouseFundingInfo">
           <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#FundingInfo</
process:parameterType>
   </process:Input>
   </process:hasInput>

   <process:hasOutput>
   <process:Output rdf:ID="BuyHousePropertyListing">
           <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#PropertyListing</
process:parameterType>
   </process:Output>
   </process:hasOutput>

   <process:hasOutput>
   <process:Output rdf:ID="BuyHouseReceipt">
           <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#Receipt</
process:parameterType>
   </process:Output>
   </process:hasOutput>

   <process:hasOutput>
   <process:Output rdf:ID="BuyHouseDeeds">
           <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#Deeds</
process:parameterType>
   </process:Output>
   </process:hasOutput>

   <process:composedOf>
   <process:Sequence>
           <process:components>
                   <process:ControlConstructList>
                           <list:first>
                           <process:Perform rdf:ID="FindHousePerform">
                                   <process:process rdf:resource="#FindHouse"/>
                                   <process:hasDataFrom>
                                           <process:InputBinding>
                                                   <process:toParam rdf:resource="#
FindHouseSearchTerm"/>
                                                   <process:valueSource>
                                                   <process:ValueOf>
                                                           <process:theVar rdf:resource="#
BuyHouseSearchTerm"/>
                                                           <process:fromProcess rdf:resource=
"&process;#TheParentPerform"/>
                                                   </process:ValueOf>
                                                   </process:valueSource>
                                           </process:InputBinding>
                                   </process:hasDataFrom>
                           </process:Perform>
                           </list:first>
                           <list:rest>
                           <process:ControlConstructList>
                                           <list:first>
                                           <process:Perform rdf:ID="OfferSequencePerform">
                                                   <process:process rdf:resource="#
OfferSequence"/>
                                                   <process:hasDataFrom>
                                                           <process:InputBinding>
```

```
                                                                <process:toParam
rdf:resource="#OfferSequencePropertyOffer"/>
                                                                <process:valueSource>
                                                                <process:ValueOf>
                                                                        <process:theVar
rdf:resource="#BuyHousePropertyOffer"/>
                                                                        <
process:fromProcess rdf:resource="&process;#TheParentPerform"/>
                                                                </process:ValueOf>
                                                                </process:valueSource>
                                                        </process:InputBinding>
                                                </process:hasDataFrom>
                                        </process:Perform>
                                        </list:first>
                                        <list:rest>
                                        <process:ControlConstructList>
                                                        <list:first>
                                                        <process:Perform rdf:ID="
LegalRequirementsPerform">
                                                                <process:process
rdf:resource="#LegalRequirements"/>
                                                                <process:hasDataFrom>
                                                                        <
process:InputBinding>
                                                                                <
process:toParam rdf:resource="#LegalRequirementsPropertyOffer"/>
                                                                                <
process:valueSource>
                                                                                <
process:ValueOf>

        <process:theVar rdf:resource="#OfferSequenceFinalPropertyOffer"/>
                                                                                <
process:fromProcess rdf:resource="&THIS;#OfferSequencePerform"/>
                                                                                </
process:ValueOf>
                                                                                </
process:valueSource>
                                                                        </
process:InputBinding>
                                                                </process:hasDataFrom>
                                                                <
process:hasDataFrom>
                                                                <
process:InputBinding>
                                                                <
process:toParam rdf:resource="#LegalRequirementsFundingInfo"/>
                                                                <
process:valueSource>
                                                                <
process:ValueOf>
                                                                <
process:theVar rdf:resource="#BuyHouseFundingInfo"/>

        <process:fromProcess rdf:resource="&process;#TheParentPerform"/>
                                                                </
process:ValueOf>
                                                                </
process:valueSource>
```

```
                                                                                </
process:InputBinding>
                                                                        </process:hasDataFrom>
                                                                </process:Perform>
                                                                </list:first>
                                                                <list:rest>
                                                                <process:ControlConstructList>
                                                                                <
list:first>
                                                                                <
process:Perform rdf:ID="TransferFundsPerform">
                                                                                <
process:process rdf:resource="#TransferFunds"/>
                                                                                <
process:hasDataFrom>
                                                                                                <
process:InputBinding>
                <process:toParam rdf:resource="#TransferFundsAgreement"/>
                <process:valueSource>
                <process:ValueOf>
                <process:theVar rdf:resource="#LegalRequirementsAgreement"/>
                <process:fromProcess rdf:resource="&THIS;#LegalRequirementsPerform"/>
        </process:ValueOf>
        </process:valueSource>
                                                                                </
process:InputBinding>
                                                                                </
process:hasDataFrom>
                                                                                </
process:Perform>
                                                                        </list:first>
                                                                                <list:rest
>
                                                                                <
process:ControlConstructList>
        <list:first>
                                                                                <
process:Perform rdf:ID="TransferDeedsPerform">
        <process:process rdf:resource="#TransferDeeds"/>
        </process:Perform>
                                                                                </
list:first>
                                                                                <
list:rest rdf:resource="&list;#nil"/>
                                                                                </
process:ControlConstructList>
                                                                        </list:rest>
                                                                </
process:ControlConstructList>
```

```
                                                        </list:rest>
                                                    </process:ControlConstructList>
                                                </list:rest>
                                            </process:ControlConstructList>
                                        </list:rest>
                                    </process:ControlConstructList>
                            </process:components>
                    </process:Sequence>
                    </process:composedOf>

                    <process:hasResult>
                    <process:Result>
                            <process:inCondition rdf:resource="&expr;#AlwaysTrue"/>
                            <process:withOutput>
                                    <process:OutputBinding>
                                            <process:toParam rdf:resource="#BuyHousePropertyListing"/>
                                            <process:valueSource>
                                            <process:ValueOf>
                                                    <process:theVar rdf:resource="#FindHouseProperty"/>
                                                    <process:fromProcess rdf:resource="#FindHousePerform"/>
                                            </process:ValueOf>
                                            </process:valueSource>
                                    </process:OutputBinding>
                            </process:withOutput>
                    </process:Result>
                    </process:hasResult>

                    <process:hasResult>
                    <process:Result>
                            <process:inCondition rdf:resource="&expr;#AlwaysTrue"/>
                            <process:withOutput>
                                    <process:OutputBinding>
                                            <process:toParam rdf:resource="#BuyHouseReceipt"/>
                                            <process:valueSource>
                                            <process:ValueOf>
                                                    <process:theVar rdf:resource="#TransferFundsReceipt"/>
                                                    <process:fromProcess rdf:resource="TransferFundsPerform"/>
                                            </process:ValueOf>
                                            </process:valueSource>
                                    </process:OutputBinding>
                            </process:withOutput>
                    </process:Result>
                    </process:hasResult>

                    <process:hasResult>
                    <process:Result>
                            <process:inCondition rdf:resource="&expr;#AlwaysTrue"/>
                            <process:withOutput>
                                    <process:OutputBinding>
                                            <process:toParam rdf:resource="#BuyHouseDeeds"/>
                                            <process:valueSource>
                                            <process:ValueOf>
                                                    <process:theVar rdf:resource="#TransferDeedsDeeds"/>
                                                    <process:fromProcess rdf:resource="TransferDeeds"/>
                                            </process:ValueOf>
                                            </process:valueSource>
                                    </process:OutputBinding>
                            </process:withOutput>
                    </process:Result>
```

```
      </process:hasResult>

</process:CompositeProcess>


<!--
=====================================================
      CompositeProcess: OfferSequence
=====================================================
-->

<process:CompositeProcess rdf:ID="OfferSequence">
      <process:hasInput>
      <process:Input rdf:ID="OfferSequencePropertyOffer">
              <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#PropertyOffer</
   process:parameterType>
      </process:Input>
      </process:hasInput>

      <process:hasOutput>
      <process:Output rdf:ID="OfferSequenceFinalPropertyOffer">
              <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#PropertyOffer</
   process:parameterType>
      </process:Output>
      </process:hasOutput>

      <process:composedOf>
      <process:Sequence>
              <process:components>
                      <process:ControlConstructList>
                              <list:first>
                              <process:Perform rdf:ID="PlaceOfferPerform">
                                      <process:process rdf:resource="#PlaceOffer"/>
                                      <process:hasDataFrom>
                                              <process:InputBinding>
                                                      <process:toParam rdf:resource="#
   PlaceOfferOffer"/>
                                                      <process:valueSource>
                                                      <process:ValueOf>
                                                              <process:theVar rdf:resource="#
   OfferSequencePropertyOffer"/>
                                                              <process:fromProcess rdf:resource=
   "&process;#TheParentPerform"/>
                                                      </process:ValueOf>
                                                      </process:valueSource>
                                              </process:InputBinding>
                                      </process:hasDataFrom>
                              </process:Perform>
                              </list:first>
                              <list:rest>
                              <process:Perform rdf:ID="FinaliseOfferPerform">
                                      <process:process rdf:resource="#FinaliseOffer"/>
                                      <process:hasDataFrom>
                                              <process:InputBinding>
                                                      <process:toParam rdf:resource="#
   FinaliseOfferOffer"/>
                                                      <process:valueSource>
                                                      <process:ValueOf>
```

```
                                                                <process:theVar rdf:resource="#
   PlaceOfferReply"/>
                                                                <process:fromProcess rdf:resource=
   "&THIS;#PlaceOfferPerform"/>
                                                        </process:ValueOf>
                                                        </process:valueSource>
                                                </process:InputBinding>
                                        </process:hasDataFrom>
                                </process:Perform>
                                </list:rest>
                        </process:ControlConstructList>
                </process:components>
      </process:Sequence>
      </process:composedOf>

      <process:hasResult>
      <process:Result>
              <process:inCondition rdf:resource="&expr;#AlwaysTrue"/>
              <process:withOutput>
                      <process:OutputBinding>
                              <process:toParam rdf:resource="#OfferSequenceFinalPropertyOffer"/>
                              <process:valueSource>
                              <process:ValueOf>
                                      <process:theVar rdf:resource="#FinaliseOfferFinal"/>
                                      <process:fromProcess rdf:resource="#FinaliseOfferPerform"/
   >
                              </process:ValueOf>
                              </process:valueSource>
                      </process:OutputBinding>
              </process:withOutput>
      </process:Result>
      </process:hasResult>
</process:CompositeProcess>

<!--
=======================================================
      AtomicProcess: FindHouse
=======================================================
-->

<process:AtomicProcess rdf:ID="FindHouse">
      <process:hasInput>
              <process:Input rdf:ID="FindHouseSearchTerm">
                      <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#SearchTerm</
   process:parameterType>
              </process:Input>
      </process:hasInput>
      <process:hasOutput>
      <process:Output rdf:ID="FindHouseProperty">
              <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#PropertyListing</
   process:parameterType>
      </process:Output>
      </process:hasOutput>
</process:AtomicProcess>

<!--
=======================================================
      AtomicProcess: PlaceOffer
=======================================================
```

```
-->

<process:AtomicProcess rdf:ID="PlaceOffer">
      <process:hasInput>
              <process:Input rdf:ID="PlaceOfferOffer">
                      <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#
   PropertyOffer</process:parameterType>
              </process:Input>
      </process:hasInput>
      <process:hasOutput>
      <process:Output rdf:ID="PlaceOfferReply">
              <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#PropertyOffer</
   process:parameterType>
      </process:Output>
      </process:hasOutput>
</process:AtomicProcess>

<!--
=======================================================
      AtomicProcess: FinaliseOffer
=======================================================
-->

<process:AtomicProcess rdf:ID="FinaliseOffer">
      <process:hasInput>
              <process:Input rdf:ID="FinaliseOfferOffer">
                      <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#
   PropertyOffer</process:parameterType>
              </process:Input>
      </process:hasInput>
      <process:hasOutput>
      <process:Output rdf:ID="FinaliseOfferFinal">
              <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#PropertyOffer</
   process:parameterType>
      </process:Output>
      </process:hasOutput>
</process:AtomicProcess>

<!--
=======================================================
      AtomicProcess: LegalRequirements
=======================================================
-->

<process:AtomicProcess rdf:ID="LegalRequirements">
      <process:hasInput>
              <process:Input rdf:ID="LegalRequirementsPropertyOffer">
                      <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#
   PropertyOffer</process:parameterType>
              </process:Input>
      </process:hasInput>
      <process:hasInput>
              <process:Input rdf:ID="LegalRequirementsFundingInfo">
                      <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#FundingInfo<
   /process:parameterType>
              </process:Input>
      </process:hasInput>
      <process:hasOutput>
      <process:Output rdf:ID="LegalRequirementsAgreement">
```

```
                <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#Agreement</
    process:parameterType>
        </process:Output>
        </process:hasOutput>
  </process:AtomicProcess>

  <!--
  ========================================================
        AtomicProcess: TransferFunds
  ========================================================
  -->

  <process:AtomicProcess rdf:ID="TransferFunds">
        <process:hasInput>
                <process:Input rdf:ID="TransferFundsAgreement">
                        <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#Agreement</
    process:parameterType>
                </process:Input>
        </process:hasInput>
        <process:hasOutput>
        <process:Output rdf:ID="TransferFundsReceipt">
                <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#Receipt</
    process:parameterType>
        </process:Output>
        </process:hasOutput>
  </process:AtomicProcess>

  <!--
  ========================================================
        AtomicProcess: TransferDeeds
  ========================================================
  -->

  <process:AtomicProcess rdf:ID="TransferDeeds">
        <process:hasOutput>
        <process:Output rdf:ID="TransferDeedsDeeds">
                <process:parameterType rdf:datatype="&xsd;#anyURI">&concepts;#Deeds</
    process:parameterType>
        </process:Output>
        </process:hasOutput>
  </process:AtomicProcess>

</rdf:RDF>
```

A.5.7 Concepts.owls

```
<?xml version="1.0" encoding="WINDOWS-1252"?>
<!DOCTYPE uridef [
  <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema">
  <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns">
  <!ENTITY owl "http://www.w3.org/2002/07/owl">
  <!ENTITY xsd "http://www.w3.org/2001/XMLSchema">
  <!ENTITY list "http://www.daml.org/services/owl-s/1.2/generic/ObjectList.owl">
  <!ENTITY mo "http://127.0.0.1/testcases/ontology/movieontology.owl">
  <!ENTITY foaf "http://xmlns.com/foaf/0.1/">
  <!ENTITY THIS "http://127.0.0.1/testcases/ontology/concepts.owl">
]>
```

```
<rdf:RDF
        xmlns:rdf="&rdf;#"
        xmlns:xsd="&xsd;#"
        xmlns:rdfs="&rdfs;#"
        xmlns:owl="&owl;#"
        xmlns:list="&list;#"
        xmlns:mo="&mo;#"
        xmlns:foaf="&foaf;#"
    xmlns="&THIS;#"
        xml:base="&THIS;#">

  <owl:Ontology rdf:about="">
        <owl:versionInfo>
                $ID: concepts.owl, v 1.8 2011/03/07 akd07r $
        </owl:versionInfo>
        <rdfs:comment>
                TODO
        </rdfs:comment>
  </owl:Ontology>


  <!--
  ///////////////////////////////////////////////////////
        General
  ///////////////////////////////////////////////////////
  -->

  <owl:Class rdf:ID="SearchTerm">
        <rdfs:subClassOf rdf:resource="&xsd;#string"/>
  </owl:Class>

  <owl:Class rdf:ID="BusinessBusiness">
    <rdfs:subClassOf rdf:resource="#Location"/>
  </owl:Class>

  <owl:Class rdf:ID="Location">
        <rdfs:comment>A generic class for locations.  It includes both real and fantastic places</
     rdfs:comment>
    <rdfs:subClassOf rdf:resource="#Tangible-Thing"/>
  </owl:Class>

  <owl:Class rdf:ID="Tangible-Thing">
        <rdfs:comment>Something which is not intangible, something which is physical, made of
     matter. It does not matter whether things are real of imaginary.  Therefore we consider
     Mickey Mouse's car and a hippogriff as  tangible things</rdfs:comment>
        <rdfs:subClassOf rdf:resource="#Temporal-Thing"/>
  </owl:Class>

  <owl:Class rdf:ID="Intangible-Thing">
        <rdfs:comment>Something which is intangible</rdfs:comment>
  </owl:Class>

  <owl:Class rdf:ID="Temporal-Thing">
        <rdfs:comment>Like in Cyc, this is something which has a temporal extent.</rdfs:comment>
  </owl:Class>

  <owl:Class rdf:ID="Time-Position">
```

```
      <rdfs:comment>A time position is either a time interval or a time point. Any time position
    is relative to a time zone</rdfs:comment>
      <rdfs:subClassOf rdf:resource="#Intangible-Thing"/>
      <rdfs:isDefinedBy rdf:resource="&THIS;"/>
</owl:Class>

<owl:DatatypeProperty rdf:ID="in-timezone">
      <rdfs:domain rdf:resource="#Time-Position"/>
      <rdfs:range rdf:resource="&xsd;integer"/>
      <rdfs:isDefinedBy rdf:resource="&THIS;"/>
</owl:DatatypeProperty>

<owl:Class rdf:ID="Time-Point">
      <rdfs:comment>A point in time</rdfs:comment>
      <rdfs:subClassOf rdf:resource="#Time-Position"/>
      <rdfs:subClassOf>
              <owl:Restriction>
                      <owl:onProperty rdf:resource="#second-of"/>
                      <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">1</
   owl:maxCardinality>
              </owl:Restriction>
      </rdfs:subClassOf>
      <rdfs:subClassOf>
              <owl:Restriction>
                      <owl:onProperty rdf:resource="#minute-of"/>
                      <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">1</
   owl:maxCardinality>
              </owl:Restriction>
      </rdfs:subClassOf>
      <rdfs:subClassOf>
              <owl:Restriction>
                      <owl:onProperty rdf:resource="#hour-of"/>
                      <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">1</
   owl:maxCardinality>
              </owl:Restriction>
      </rdfs:subClassOf>
      <rdfs:subClassOf>
              <owl:Restriction>
                      <owl:onProperty rdf:resource="#day-of"/>
                      <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">1</
   owl:maxCardinality>
              </owl:Restriction>
      </rdfs:subClassOf>
      <rdfs:subClassOf>
              <owl:Restriction>
                      <owl:onProperty rdf:resource="#month-of"/>
                      <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">1</
   owl:maxCardinality>
              </owl:Restriction>
      </rdfs:subClassOf>
      <rdfs:subClassOf>
              <owl:Restriction>
                      <owl:onProperty rdf:resource="#year-of"/>
                      <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">1</
   owl:maxCardinality>
              </owl:Restriction>
      </rdfs:subClassOf>
      <rdfs:isDefinedBy rdf:resource="&THIS;"/>
</owl:Class>
```

```
<owl:DatatypeProperty rdf:ID="second-of">
      <rdfs:domain rdf:resource="#Time-Point"/>
      <rdfs:range rdf:resource="&xsd;nonNegativeInteger"/>
      <rdfs:isDefinedBy rdf:resource="&THIS;"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="minute-of">
      <rdfs:domain rdf:resource="#Time-Point"/>
      <rdfs:range rdf:resource="&xsd;nonNegativeInteger"/>
      <rdfs:isDefinedBy rdf:resource="&THIS;"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="hour-of">
      <rdfs:domain rdf:resource="#Time-Point"/>
      <rdfs:range rdf:resource="&xsd;nonNegativeInteger"/>
      <rdfs:isDefinedBy rdf:resource="&THIS;"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="day-of">
      <rdfs:domain rdf:resource="#Time-Point"/>
      <rdfs:range rdf:resource="&xsd;nonNegativeInteger"/>
      <rdfs:isDefinedBy rdf:resource="&THIS;"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="month-of">
      <rdfs:domain rdf:resource="#Time-Point"/>
      <rdfs:range rdf:resource="&xsd;nonNegativeInteger"/>
      <rdfs:isDefinedBy rdf:resource="&THIS;"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="year-of">
      <rdfs:domain rdf:resource="#Time-Point"/>
      <rdfs:range rdf:resource="&xsd;nonNegativeInteger"/>
      <rdfs:isDefinedBy rdf:resource="&THIS;"/>
</owl:DatatypeProperty>

<owl:Class rdf:ID="Calendar-Date">
      <rdfs:comment>A point in time</rdfs:comment>
      <rdfs:subClassOf rdf:resource="#Time-Point"/>
      <rdfs:subClassOf>
              <owl:Restriction>
                      <owl:onProperty rdf:resource="#second-of"/>
                      <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">0</
   owl:maxCardinality>
              </owl:Restriction>
      </rdfs:subClassOf>
      <rdfs:subClassOf>
              <owl:Restriction>
                      <owl:onProperty rdf:resource="#minute-of"/>
                      <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">0</
   owl:maxCardinality>
              </owl:Restriction>
      </rdfs:subClassOf>
      <rdfs:subClassOf>
              <owl:Restriction>
                      <owl:onProperty rdf:resource="#hour-of"/>
                      <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">0</
   owl:maxCardinality>
```

```
                </owl:Restriction>
        </rdfs:subClassOf>
        <rdfs:subClassOf>
                <owl:Restriction>
                        <owl:onProperty rdf:resource="#day-of"/>
                        <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">1</
    owl:maxCardinality>
                </owl:Restriction>
        </rdfs:subClassOf>
        <rdfs:subClassOf>
                <owl:Restriction>
                        <owl:onProperty rdf:resource="#month-of"/>
                        <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">1</
    owl:maxCardinality>
                </owl:Restriction>
        </rdfs:subClassOf>
        <rdfs:subClassOf>
                <owl:Restriction>
                        <owl:onProperty rdf:resource="#year-of"/>
                        <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">1</
    owl:maxCardinality>
                </owl:Restriction>
        </rdfs:subClassOf>
        <rdfs:isDefinedBy rdf:resource="&THIS;"/>
</owl:Class>

<owl:Class rdf:ID="Price">
      <rdfs:subClassOf rdf:resource="#InternationalCurrency" />
</owl:Class>

<!--
=======================================================
      filmLocator
=======================================================
-->
<owl:Class rdf:ID="Rating">
      <rdfs:subClassOf rdf:resource="#Score" />
</owl:Class>

<owl:Class rdf:ID="Score"/>

<owl:DatatypeProperty rdf:ID="value">
      <rdfs:domain rdf:resource="#Score"/>
      <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#integer"/>
</owl:DatatypeProperty>

<owl:Class rdf:ID="Film"/>

<owl:Class rdf:ID="FilmReview">
  <rdfs:subClassOf rdf:resource="#Document"/>
      <rdfs:subClassOf>
          <owl:Restriction>
            <owl:onProperty rdf:resource="#hasScore" />
            <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
          </owl:Restriction>
        </rdfs:subClassOf>
      <rdfs:subClassOf>
          <owl:Restriction>
            <owl:onProperty rdf:resource="#hasFilm" />
```

```
            <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
          </owl:Restriction>
       </rdfs:subClassOf>
</owl:Class>

<owl:DatatypeProperty rdf:ID="hasScore">
      <rdfs:domain rdf:resource="#Filmreview"/>
      <rdfs:range rdf:resource="#Rating"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="hasReview">
      <rdfs:domain rdf:resource="#FilmReview"/>
      <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="hasFilm">
      <rdfs:domain rdf:resource="#FilmReview"/>
      <rdfs:range rdf:resource="&mo;#Film"/>
</owl:DatatypeProperty>

<owl:Class rdf:ID="Cinema">
  <rdfs:subClassOf rdf:resource="#BusinessLocation"/>
</owl:Class>

<owl:Class rdf:ID="FilmInfo">
  <rdfs:subClassOf rdf:resource="#Document"/>
</owl:Class>

<owl:DatatypeProperty rdf:ID="hasCinema">
      <rdfs:domain rdf:resource="#FilmInfo"/>
      <rdfs:range rdf:resource="#Cinema"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="hasTimes">
      <rdfs:domain rdf:resource="#FilmInfo"/>
      <rdfs:range rdf:resource="#Time-Point"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="hasPrice">
      <rdfs:domain rdf:resource="#FilmInfo"/>
      <rdfs:range rdf:resource="#FilmPrice"/>
</owl:DatatypeProperty>

<owl:Class rdf:ID="FilmPrice">
      <rdfs:subClassOf rdf:resource="#Price" />
</owl:Class>

<owl:Class rdf:ID="Map">
      <rdfs:subClassOf>
              <owl:Class>
                      <owl:intersectionOf rdf:parseType="Collection">
                              <rdf:Description rdf:about="#GraphicalInformationRepresentation"/>
                              <owl:Restriction>
                                      <owl:onProperty rdf:resource="#representsInformation"/>
                                      <owl:someValuesFrom rdf:resource="Location"/>
                              </owl:Restriction>
                      </owl:intersectionOf>
              </owl:Class>
      </rdfs:subClassOf>
```

```
</owl:Class>

<owl:DatatypeProperty rdf:ID="imageScale">
      <rdfs:domain rdf:resource="#GraphicalInformationRepresentation"/>
      <rdfs:range rdf:resource="&xsd;double"/>
</owl:DatatypeProperty>

<owl:Class rdf:ID="GraphicalInformationRepresentation">
      <rdfs:subClassOf rdf:resource="#InformationRepresentation"/>
</owl:Class>

<owl:Class rdf:ID="InformationRepresentation">
      <rdfs:subClassOf rdf:resource="#Tangible-Thing"/>
</owl:Class>

<owl:ObjectProperty rdf:about="#isRepresentedBy">
      <rdf:type rdf:resource="&owl;TransitiveProperty"/>
      <rdfs:range rdf:resource="#InformationRepresentation"/>
      <owl:inverseOf rdf:resource="#representsInformation"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="#representsInformation">
      <rdf:type rdf:resource="&owl;TransitiveProperty"/>
      <rdfs:domain rdf:resource="#InformationRepresentation"/>
</owl:ObjectProperty>


<!--
=======================================================
      findExpert
=======================================================
-->

<owl:Class rdf:ID="AcademicPaper"/>

<owl:DatatypeProperty rdf:ID="title">
      <rdfs:domain rdf:resource="#AcademicPaper"/>
      <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="authorList">
      <rdfs:domain rdf:resource="#AcademicPaper"/>
      <rdfs:range rdf:resource="#AuthorList"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="publisher">
      <rdfs:domain rdf:resource="#AcademicPaper"/>
      <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="publicationDate">
      <rdfs:domain rdf:resource="#AcademicPaper"/>
      <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="format">
      <rdfs:domain rdf:resource="#AcademicPaper"/>
      <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
```

```
<owl:Class rdf:ID="AuthorList">
      <owl:intersectionOf rdf:parseType="Collection">
              <owl:Class rdf:about="http://www.w3.org/2002/07/owl#List"/>
              <owl:Restriction>
                      <owl:onProperty rdf:resource="http://www.w3.org/2002/07/owl#item"/>
                      <owl:hasValue rdf:resource="#Author"/>
              </owl:Restriction>
      </owl:intersectionOf>
</owl:Class>

<owl:Class rdf:ID="AcademicPaperList">
      <owl:intersectionOf rdf:parseType="Collection">
              <owl:Class rdf:about="http://www.w3.org/2002/07/owl#List"/>
              <owl:Restriction>
                      <owl:onProperty rdf:resource="http://www.w3.org/2002/07/owl#item"/>
                      <owl:hasValue rdf:resource="#Paper"/>
              </owl:Restriction>
      </owl:intersectionOf>
</owl:Class>

<owl:Class rdf:ID="Author"/>

<owl:DatatypeProperty rdf:ID="authorTitle">
      <rdfs:domain rdf:resource="#Author"/>
      <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="firstName">
      <rdfs:domain rdf:resource="#Author"/>
      <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="lastName">
      <rdfs:domain rdf:resource="#Author"/>
      <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="institution">
      <rdfs:domain rdf:resource="#Author"/>
      <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>


<owl:Class rdf:ID="PaperScore" />

<owl:DatatypeProperty rdf:ID="score">
      <rdfs:domain rdf:resource="#PaperScore"/>
      <rdfs:range rdf:resource="&xsd;#integer"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="scoredpaper">
      <rdfs:domain rdf:resource="#PaperScore"/>
      <rdfs:range rdf:resource="#AcademicPaper"/>
</owl:DatatypeProperty>


<owl:Class rdf:ID="SignInData"/>
```

```
<owl:DatatypeProperty rdf:ID="acctName">
      <rdfs:domain rdf:resource="#SignInData"/>
      <rdfs:range rdf:resource="&xsd;#string"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="password">
      <rdfs:domain rdf:resource="#SignInData"/>
      <rdfs:range rdf:resource="&xsd;#string"/>
</owl:DatatypeProperty>

<owl:Class rdf:ID="UserProfileInfo"/>

<owl:Class rdf:ID="AcctInfo">
      <owl:unionOf rdf:parseType="Collection">
      <owl:Class rdf:about="#SignInData"/>
      <owl:Class rdf:about="#UserProfileInfo"/>
      </owl:unionOf>
</owl:Class>

<owl:Class rdf:ID="AcctID"/>

<owl:ObjectProperty rdf:ID="hasAcctID">
      <rdfs:comment>
      An account exists if there is an account ID associated
      with it.
      </rdfs:comment>
      <rdf:type rdf:resource="&owl;#FunctionalProperty"/>
      <rdfs:domain rdf:resource="#AcctInfo"/>
      <rdfs:range rdf:resource="#AcctID"/>
</owl:ObjectProperty>

<owl:Class rdf:ID="NonExistingAcct">
      <rdfs:comment>
      If there is no account ID for an account then that
      account simply does not exist
      </rdfs:comment>
      <owl:intersectionOf rdf:parseType="Collection">
      <owl:Class rdf:about="#AcctInfo"/>
      <owl:Restriction>
              <owl:onProperty rdf:resource="#hasAcctID"/>
              <owl:maxCardinality rdf:datatype="&xsd;#integer">0</owl:maxCardinality>
      </owl:Restriction>
      </owl:intersectionOf>
</owl:Class>

<owl:Class rdf:ID="Message" />

<owl:DatatypeProperty rdf:ID="subject">
      <rdfs:domain rdf:resource="#Message"/>
      <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="sender">
      <rdfs:domain rdf:resource="#Message"/>
      <rdfs:range rdf:resource="#AcctID"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="recipient">
      <rdfs:domain rdf:resource="#Message"/>
```

```
      <rdfs:range rdf:resource="#AcctID"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="sendDate">
      <rdfs:domain rdf:resource="#Message"/>
      <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>

<owl:Class rdf:ID="MsgSendConfirmation">
      <rdfs:subClassOf rdf:resource="&xsd;#boolean"/>
</owl:Class>


<!--
=======================================================
      bookMedicalAppointment
=======================================================
-->
<owl:Class rdf:ID="Hospital">
      <rdfs:subClassOf rdf:resource ="#CareOrganization"/>
      <rdfs:subClassOf rdf:resource="#TemporaryResidence"/>
      <rdfs:comment>
              A CareOrganization where patients reside for a short period of time while they
   undergo treatment for a disease or disorder.
      </rdfs:comment>
</owl:Class>

<owl:Class rdf:ID="CareOrganization">
      <rdfs:subClassOf rdf:resource="&foaf;#Organization"/>
      <rdfs:comment>
              Any Organization whose purpose is to provide medical care for for Humans who
   reside there, either permanently or temporarily.
      </rdfs:comment>
</owl:Class>

<owl:ObjectProperty rdf:ID="hasPCT">
      <rdf:type rdf:resource="&owl;#FunctionalProperty"/>
      <rdfs:domain rdf:resource="#CareOrganization"/>
      <rdfs:range rdf:resource="#PCT"/>
</owl:ObjectProperty>

<owl:Class rdf:ID= "TemporaryResidence">
      <rdfs:subClassOf rdf:resource ="#Residence"/>
      <rdfs:comment>
              A Residence which is strictly temporary,i.e. where no one makes his/her home.
      </rdfs:comment>
</owl:Class>

<owl:Class rdf:ID= "Residence">
      <rdfs:subClassOf rdf:resource ="#StationaryArtifact"/>
      <rdfs:comment>
              A Building or part of a Building which provides some accomodation for sleeping.
      </rdfs:comment>
</owl:Class>

<owl:Class rdf:ID= "StationaryArtifact">
      <rdfs:subClassOf rdf:resource ="#Location"/>
      <rdfs:comment>
```

```
            A StationaryArtifact is an Artifact that has a fixed spatial location.  Most
   instances of this Class are architectural works, e.g. the Eiffel Tower, the Great Pyramids,
   office towers, single-family houses, etc.
      </rdfs:comment>
</owl:Class>

<owl:Class rdf:ID="AppointmentTime" >
      <rdfs:subClassOf rdf:resource="#Time-Point"/>
</owl:Class>

<owl:Class rdf:ID="MedicalAppointment" />

<owl:DatatypeProperty rdf:ID="department">
      <rdfs:domain rdf:resource="#MedicalAppointment"/>
      <rdfs:range rdf:resource="#HospitalDepartment"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="time">
      <rdfs:domain rdf:resource="#MedicalAppointment"/>
      <rdfs:range rdf:resource="#AppointmentTime"/>
</owl:DatatypeProperty>

<owl:Class rdf:about="#AppointmentTimeList">
      <rdfs:subClassOf>
      <owl:Restriction>
              <owl:onProperty rdf:resource="#hasAppointments" />
              <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:minCardinality>
      </owl:Restriction>
      </rdfs:subClassOf>
</owl:Class>

<owl:ObjectProperty rdf:ID="hasAppointments">
      <rdfs:domain rdf:resource="#AppointmentTimeList"/>
      <rdfs:range rdf:resource="#AppointmentTime"/>
</owl:ObjectProperty>

<owl:Class rdf:ID="DepartmentCode" />

<owl:Class rdf:ID="HospitalDepartment">
      <rdfs:subClassOf rdf:resource ="#CareOrganization"/>
</owl:Class>

<owl:ObjectProperty rdf:ID="hasDepartment">
      <rdf:type rdf:resource="&owl;#FunctionalProperty"/>
      <rdfs:domain rdf:resource="#Hospital"/>
      <rdfs:range rdf:resource="#HospitalDepartment"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="departmentOf">
      <owl:inverseOf rdf:resource="#hasDepartment" />
</owl:ObjectProperty>

<owl:Class rdf:ID="AppointmentGuide" >
      <rdfs:subClassOf rdf:resource="&foaf;#Document"/>
</owl:Class>

<owl:Class rdf:ID="MedicalCondition" />

<owl:Class rdf:ID="PCT" />
```

```
<owl:DatatypeProperty rdf:ID="name">
      <rdfs:domain rdf:resource="#PCT"/>
      <rdfs:range rdf:resource="&xsd;#string"/>
</owl:DatatypeProperty>

<!--
///////////////////////////////////////////////////////
      generateReportForItinerary
///////////////////////////////////////////////////////
-->

<owl:Class rdf:ID="Itinerary" />

<owl:Class rdf:ID="TravelItinerary" >
      <rdfs:subClassOf rdf:resource="#Itinerary"/>
</owl:Class>

<owl:DatatypeProperty rdf:ID="itineraryID">
      <rdfs:domain rdf:resource="#Itinerary"/>
      <rdfs:range rdf:resource="&xsd;#string"/>
</owl:DatatypeProperty>

<owl:ObjectProperty rdf:ID="itineraryOwner">
      <rdf:type rdf:resource="&owl;#FunctionalProperty"/>
      <rdfs:domain rdf:resource="#Itinerary"/>
      <rdfs:range rdf:resource="#User"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="hasFlight">
      <rdf:type rdf:resource="&owl;#FunctionalProperty"/>
      <rdfs:domain rdf:resource="#Itinerary"/>
      <rdfs:range rdf:resource="#Flight"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="hasTransfer">
      <rdf:type rdf:resource="&owl;#FunctionalProperty"/>
      <rdfs:domain rdf:resource="#Itinerary"/>
      <rdfs:range rdf:resource="#Transfer"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="hasHotel">
      <rdf:type rdf:resource="&owl;#FunctionalProperty"/>
      <rdfs:domain rdf:resource="#Itinerary"/>
      <rdfs:range rdf:resource="#Hotel"/>
</owl:ObjectProperty>

<owl:Class rdf:ID="VerificationReport" >
      <rdfs:subClassOf rdf:resource="#ItineraryReport"/>
</owl:Class>

<owl:Class rdf:ID="ItineraryReport" >
      <rdfs:subClassOf rdf:resource="#Document"/>
</owl:Class>

<owl:ObjectProperty rdf:ID="hasItinerary">
      <rdf:type rdf:resource="&owl;#FunctionalProperty"/>
      <rdfs:domain rdf:resource="#ItineraryReport"/>
      <rdfs:range rdf:resource="#Itinerary"/>
```

```
</owl:ObjectProperty>

<owl:DatatypeProperty rdf:ID="verificationSuccess">
      <rdfs:domain rdf:resource="#VerificationReport"/>
      <rdfs:range rdf:resource="&xsd;#boolean"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="report">
      <rdfs:domain rdf:resource="#ItineraryReport"/>
      <rdfs:range rdf:resource="&xsd;#string"/>
</owl:DatatypeProperty>

<owl:Class rdf:ID="Accommodation">
      <rdfs:subClassOf rdf:resource="#Location"/>
</owl:Class>

<owl:Class rdf:ID="Hotel">
      <owl:disjointWith>
              <owl:Class rdf:about="#BedAndBreakfast"/>
      </owl:disjointWith>
      <rdfs:subClassOf rdf:resource="#Accommodation"/>
</owl:Class>

<owl:Class rdf:ID="LuxuryHotel">
      <rdfs:subClassOf>
              <owl:Class rdf:about="#Hotel"/>
      </rdfs:subClassOf>
      <rdfs:subClassOf>
              <owl:Restriction>
                      <owl:hasValue rdf:resource="#ThreeStarRating"/>
                      <owl:onProperty>
                              <owl:ObjectProperty rdf:about="#hasRating"/>
                      </owl:onProperty>
              </owl:Restriction>
      </rdfs:subClassOf>
</owl:Class>

<owl:Class rdf:ID="BedAndBreakfast">
      <owl:disjointWith rdf:resource="#Hotel"/>
      <rdfs:subClassOf rdf:resource="#Accommodation"/>
</owl:Class>

<owl:Class rdf:ID="AccommodationRating">
      <owl:equivalentClass>
              <owl:Class>
                      <owl:oneOf rdf:parseType="Collection">
                              <AccommodationRating rdf:about="#OneStarRating"/>
                              <AccommodationRating rdf:about="#TwoStarRating"/>
                              <AccommodationRating rdf:about="#ThreeStarRating"/>
                      </owl:oneOf>
              </owl:Class>
      </owl:equivalentClass>
</owl:Class>

<owl:Class rdf:ID="BudgetAccommodation">
      <owl:equivalentClass>
              <owl:Class>
                      <owl:intersectionOf rdf:parseType="Collection">
                              <owl:Class rdf:about="#Accommodation"/>
```

```
                        <owl:Restriction>
                                <owl:someValuesFrom>
                                        <owl:Class>
                                                <owl:oneOf rdf:parseType="Collection">
                                                        <AccommodationRating rdf:ID="
    OneStarRating">
                                                                <owl:differentFrom>
                                                                        <
    AccommodationRating rdf:ID="ThreeStarRating">
                                                                                <
    owl:differentFrom>
                                                                                        <
    AccommodationRating rdf:ID="TwoStarRating">

            <owl:differentFrom rdf:resource="#OneStarRating"/>

            <owl:differentFrom rdf:resource="#ThreeStarRating"/>
                                                                                        </
    AccommodationRating>
                                                                                </
    owl:differentFrom>
                                                                                <
    owl:differentFrom rdf:resource="#OneStarRating"/>
                                                                        </
    AccommodationRating>
                                                                </owl:differentFrom>
                                                                <owl:differentFrom
    rdf:resource="#TwoStarRating"/>
                                                        </AccommodationRating>
                                                        <AccommodationRating rdf:about="#
    TwoStarRating"/>
                                                </owl:oneOf>
                                        </owl:Class>
                                </owl:someValuesFrom>
                                <owl:onProperty>
                                        <owl:ObjectProperty rdf:about="#hasRating"/>
                                </owl:onProperty>
                        </owl:Restriction>
                </owl:intersectionOf>
            </owl:Class>
      </owl:equivalentClass>
</owl:Class>

<owl:ObjectProperty rdf:ID="hasRating">
      <rdfs:range rdf:resource="#AccommodationRating"/>
      <rdfs:domain rdf:resource="#Accommodation"/>
</owl:ObjectProperty>

<owl:Class rdf:ID="Airport">
      <rdfs:subClassOf rdf:resource="#Location"/>
</owl:Class>

<owl:ObjectProperty rdf:ID="hasCode">
      <rdf:type rdf:resource="&owl;#FunctionalProperty"/>
      <rdfs:domain rdf:resource="#Airport"/>
      <rdfs:range rdf:resource="http://www.daml.ri.cmu.edu/ont/AirportCodes.daml#AirportCode"/>
</owl:ObjectProperty>

<owl:Class rdf:ID="Transport" />
```

```
<owl:Class rdf:ID="Transfer">
      <owl:disjointWith rdf:resource="#Flight"/>
      <rdfs:subClassOf rdf:resource="#Transport"/>
</owl:Class>

<owl:Class rdf:ID="Flight">
      <owl:disjointWith rdf:resource="#Transfer"/>
      <rdfs:subClassOf rdf:resource="#Transport"/>
</owl:Class>

<owl:DatatypeProperty rdf:ID="flightOrigin">
      <rdfs:domain rdf:resource="#Flight"/>
      <rdfs:range rdf:resource="#Airport"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="flightDestination">
      <rdfs:domain rdf:resource="#Flight"/>
      <rdfs:range rdf:resource="#Airport"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="flightID">
      <rdfs:domain rdf:resource="#Flight"/>
      <rdfs:range rdf:resource="&xsd;#String"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="FlightDepartDateTime">
      <rdfs:domain rdf:resource="#Flight"/>
      <rdfs:range rdf:resource="&xsd;#DateTime"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="FlightArriveDateTime">
      <rdfs:domain rdf:resource="#Flight"/>
      <rdfs:range rdf:resource="&xsd;#DateTime"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="planeType">
      <rdfs:domain rdf:resource="#Flight"/>
      <rdfs:range rdf:resource="&xsd;#String"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="TransferOrigin">
      <rdfs:domain rdf:resource="#Transfer"/>
      <rdfs:range rdf:resource="#Location"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="TransferDestination">
      <rdfs:domain rdf:resource="#Transfer"/>
      <rdfs:range rdf:resource="#Location"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="transferID">
      <rdfs:domain rdf:resource="#Transfer"/>
      <rdfs:range rdf:resource="&xsd;#String"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="TransferDepartDateTime">
      <rdfs:domain rdf:resource="#Transfer"/>
      <rdfs:range rdf:resource="&xsd;#DateTime"/>
```

```
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="TransferArriveDateTime">
      <rdfs:domain rdf:resource="#Transfer"/>
      <rdfs:range rdf:resource="&xsd;#DateTime"/>
</owl:DatatypeProperty>

<owl:Class rdf:ID="BusTransfer">
      <rdfs:subClassOf rdf:resource="#Transfer"/>
</owl:Class>
<owl:Class rdf:ID="TrainTransfer">
      <rdfs:subClassOf rdf:resource="#Transfer"/>
</owl:Class>
<owl:Class rdf:ID="CarTransfer">
      <rdfs:subClassOf rdf:resource="#Transfer"/>
</owl:Class>


<!--
=======================================================
      buyGenericItem
=======================================================
-->

<owl:Class rdf:ID="Item">
      <rdfs:subClassOf rdf:resource="#Tangible-Thing"/>
</owl:Class>

<owl:DatatypeProperty rdf:ID="price">
      <rdfs:domain rdf:resource="#Item"/>
      <rdfs:range rdf:resource="#InternationalCurrency"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="itemID">
      <rdfs:domain rdf:resource="#Item"/>
      <rdfs:range rdf:resource="&xsd;#string"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="itemDescription">
      <rdfs:domain rdf:resource="#Item"/>
      <rdfs:range rdf:resource="&xsd;#string"/>
</owl:DatatypeProperty>

<owl:Class rdf:ID="Cart">
      <owl:intersectionOf rdf:parseType="Collection">
      <owl:Class rdf:about="&list;#List"/>
      <owl:Restriction>
              <owl:onProperty rdf:resource="&list;#first"/>
              <owl:allValuesFrom rdf:resource="#Item"/>
      </owl:Restriction>
      <owl:Restriction>
              <owl:onProperty rdf:resource="&list;#rest"/>
              <owl:allValuesFrom rdf:resource="#Cart"/>
      </owl:Restriction>
      </owl:intersectionOf>
</owl:Class>

<owl:Class rdf:ID="CreditCard">
      <rdfs:subClassOf>
```

```
      <owl:Restriction>
              <owl:onProperty rdf:resource="#cardNumber"/>
              <owl:cardinality rdf:datatype="&xsd;#integer">1</owl:cardinality>
      </owl:Restriction>
      </rdfs:subClassOf>
      <rdfs:subClassOf>
      <owl:Restriction>
              <owl:onProperty rdf:resource="#cardType"/>
              <owl:cardinality rdf:datatype="&xsd;#integer">1</owl:cardinality>
      </owl:Restriction>
      </rdfs:subClassOf>
      <rdfs:subClassOf>
      <owl:Restriction>
              <owl:onProperty rdf:resource="#cardExpiration"/>
              <owl:cardinality rdf:datatype="&xsd;#integer">1</owl:cardinality>
      </owl:Restriction>
      </rdfs:subClassOf>
      <rdfs:subClassOf>
      <owl:Restriction>
              <owl:onProperty rdf:resource="#validity"/>
              <owl:cardinality rdf:datatype="&xsd;#integer">1</owl:cardinality>
      </owl:Restriction>
      </rdfs:subClassOf>
</owl:Class>

<owl:DatatypeProperty rdf:ID="cardNumber">
      <rdfs:domain rdf:resource="#CreditCard"/>
      <rdfs:range rdf:resource="&xsd;#decimal"/>
</owl:DatatypeProperty>

<owl:ObjectProperty rdf:ID="cardType">
      <rdfs:domain rdf:resource="#CreditCard"/>
      <rdfs:range rdf:resource="#CreditCardType"/>
</owl:ObjectProperty>

<owl:DatatypeProperty rdf:ID="cardExpiration">
      <rdfs:domain rdf:resource="#CreditCard"/>
      <rdfs:range rdf:resource="&xsd;#gYearMonth"/>
</owl:DatatypeProperty>

<owl:ObjectProperty rdf:ID="validity">
      <rdfs:domain rdf:resource="#CreditCard"/>
      <rdfs:range rdf:resource="#ValidityType"/>
</owl:ObjectProperty>

<owl:Class rdf:ID="CreditCardType">
      <owl:oneOf rdf:parseType="Collection">
      <CreditCardType rdf:ID="MasterCard"/>
      <CreditCardType rdf:ID="VISA"/>
      <CreditCardType rdf:ID="AmericanExpress"/>
      <CreditCardType rdf:ID="DiscoverCard"/>
      </owl:oneOf>
</owl:Class>

<owl:Class rdf:ID="ValidityType">
      <owl:oneOf rdf:parseType="Collection">
      <ValidityType rdf:ID="Valid"/>
      <ValidityType rdf:ID="Expired"/>
      <ValidityType rdf:ID="InvalidCCNumber"/>
```

```
      <ValidityType rdf:ID="InvalidCCType"/>
      <ValidityType rdf:ID="AuthorizationRefused"/>
      </owl:oneOf>
</owl:Class>

<owl:Class rdf:ID="Shipment">
      <rdfs:subClassOf>
      <owl:Restriction>
              <owl:onProperty rdf:resource="#shippedTo"/>
              <owl:cardinality rdf:datatype="&xsd;#integer">1</owl:cardinality>
      </owl:Restriction>
      </rdfs:subClassOf>
      <rdfs:subClassOf>
      <owl:Restriction>
              <owl:onProperty rdf:resource="#shippedItem"/>
              <owl:cardinality rdf:datatype="&xsd;#integer">1</owl:cardinality>
      </owl:Restriction>
      </rdfs:subClassOf>
      <rdfs:subClassOf>
      <owl:Restriction>
              <owl:onProperty rdf:resource="#deliveryType"/>
              <owl:cardinality rdf:datatype="&xsd;#integer">1</owl:cardinality>
      </owl:Restriction>
      </rdfs:subClassOf>
      <rdfs:subClassOf>
      <owl:Restriction>
              <owl:onProperty rdf:resource="#packagingType"/>
              <owl:cardinality rdf:datatype="&xsd;#integer">1</owl:cardinality>
      </owl:Restriction>
      </rdfs:subClassOf>
</owl:Class>

<owl:ObjectProperty rdf:ID="shippedTo">
      <rdfs:domain rdf:resource="#Shipment"/>
      <rdfs:range rdf:resource="#AcctID"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="shippedItem">
      <rdfs:domain rdf:resource="#Shipment"/>
      <rdfs:range rdf:resource="#Item"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="deliveryType">
      <rdfs:domain rdf:resource="#Shipment"/>
      <rdfs:range rdf:resource="#DeliveryType"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="packagingType">
      <rdfs:domain rdf:resource="#Shipment"/>
      <rdfs:range rdf:resource="#PackagingType"/>
</owl:ObjectProperty>

<owl:Class rdf:ID="PackagingType">
      <owl:oneOf rdf:parseType="Collection">
      <PackagingType rdf:ID="Giftwrap"/>
      <PackagingType rdf:ID="Ordinary"/>
      </owl:oneOf>
</owl:Class>
```

```
<owl:Class rdf:ID="DeliveryType">
      <owl:oneOf rdf:parseType="Collection">
      <DeliveryType rdf:ID="FedExOneDay"/>
      <DeliveryType rdf:ID="FedEx2-3day"/>
      <DeliveryType rdf:ID="UPS"/>
      <DeliveryType rdf:ID="OrdinaryMail"/>
      </owl:oneOf>
</owl:Class>


<!--
=======================================================
      buyHouse
=======================================================
-->

<owl:Class rdf:ID="Property">
      <rdfs:subClassOf rdf:resource="#Location"/>
</owl:Class>

<owl:ObjectProperty rdf:ID="hasAddress">
      <rdf:type rdf:resource="&owl;#FunctionalProperty"/>
      <rdfs:domain rdf:resource="#Property"/>
      <rdfs:range rdf:resource="#PropertyAddress"/>
</owl:ObjectProperty>

<owl:Class rdf:ID="PropertyAddress" />

<owl:DatatypeProperty rdf:ID="addressLine1">
      <rdfs:domain rdf:resource="#PropertyAddress"/>
      <rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="addressLine2">
      <rdfs:domain rdf:resource="#PropertyAddress"/>
      <rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="addressLine3">
      <rdfs:domain rdf:resource="#PropertyAddress"/>
      <rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="addressLine4">
      <rdfs:domain rdf:resource="#PropertyAddress"/>
      <rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="postcode">
      <rdfs:domain rdf:resource="#PropertyAddress"/>
      <rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="country">
      <rdfs:domain rdf:resource="#PropertyAddress"/>
      <rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>

<owl:Class rdf:ID="PropertyListing" />
```

```
<owl:ObjectProperty rdf:ID="hasProperty">
      <rdf:type rdf:resource="&owl;#FunctionalProperty"/>
      <rdfs:domain rdf:resource="#PropertyListing"/>
      <rdfs:range rdf:resource="#Property"/>
</owl:ObjectProperty>

<owl:DatatypeProperty rdf:ID="listingID">
      <rdfs:domain rdf:resource="#PropertyListing"/>
      <rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>

<owl:Class rdf:ID="Currency">
      <rdfs:subClassOf rdf:resource="&xsd;#float"/>
</owl:Class>

<owl:Class rdf:ID="InternationalCurrency">
      <rdfs:subClassOf rdf:resource="#Currency"/>
</owl:Class>

<owl:DatatypeProperty rdf:ID="countryCode">
      <rdfs:domain rdf:resource="#InternationalCurrency"/>
      <rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>

<owl:Class rdf:ID="PropertyOffer">
      <rdfs:subClassOf rdf:resource="#Document"/>
</owl:Class>

<owl:DatatypeProperty rdf:ID="offerID">
      <rdfs:domain rdf:resource="#PropertyOffer"/>
      <rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="thisProperty">
      <rdfs:domain rdf:resource="#PropertyOffer"/>
      <rdfs:range rdf:resource="#PropertyListing"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="currentOffer">
      <rdfs:domain rdf:resource="#PropertyOffer"/>
      <rdfs:range rdf:resource="#InternationalCurrency"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="currentCounterOffer">
      <rdfs:domain rdf:resource="#PropertyOffer"/>
      <rdfs:range rdf:resource="#InternationalCurrency"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="accepted">
      <rdfs:domain rdf:resource="#PropertyOffer"/>
      <rdfs:range rdf:resource="&xsd;boolean"/>
</owl:DatatypeProperty>

<owl:Class rdf:ID="FundingInfo">
      <rdfs:subClassOf rdf:resource="#Document"/>
</owl:Class>

<owl:Class rdf:ID="Bank" />
```

```
  <owl:ObjectProperty rdf:ID="hasBank">
        <rdf:type rdf:resource="&owl;#FunctionalProperty"/>
        <rdfs:domain rdf:resource="#FundingInfo"/>
        <rdfs:range rdf:resource="#Bank"/>
  </owl:ObjectProperty>

  <owl:DatatypeProperty rdf:ID="fundingAmount">
        <rdfs:domain rdf:resource="#FundingInfo"/>
        <rdfs:range rdf:resource="#InternationalCurrency"/>
  </owl:DatatypeProperty>

  <owl:Class rdf:ID="Agreement">
        <rdfs:subClassOf rdf:resource="#Document"/>
  </owl:Class>

  <owl:ObjectProperty rdf:ID="agreedFunding">
        <rdf:type rdf:resource="&owl;#FunctionalProperty"/>
        <rdfs:domain rdf:resource="#Agreement"/>
        <rdfs:range rdf:resource="#FundingInfo"/>
  </owl:ObjectProperty>

  <owl:ObjectProperty rdf:ID="agreedProperty">
        <rdf:type rdf:resource="&owl;#FunctionalProperty"/>
        <rdfs:domain rdf:resource="#Agreement"/>
        <rdfs:range rdf:resource="#PropertyOffer"/>
  </owl:ObjectProperty>

  <owl:Class rdf:ID="Receipt">
        <rdfs:subClassOf rdf:resource="#Document"/>
  </owl:Class>

  <owl:Class rdf:ID="Deeds">
        <rdfs:subClassOf rdf:resource="#Document"/>
  </owl:Class>

  <owl:ObjectProperty rdf:ID="propertyDeeds">
        <rdf:type rdf:resource="&owl;#FunctionalProperty"/>
        <rdfs:domain rdf:resource="#Deeds"/>
        <rdfs:range rdf:resource="#Property"/>
  </owl:ObjectProperty>

</rdf:RDF>
```

Appendix: Results

Test	Complexity	Runs	Time to finish (sec)
1	One simple NPE-L catalog, one simple OWL-S document.	8	22.04
2	One large NPE-L catalog, five complex OWL-S documents.	8	35.52
3	One simple NPE-L catalog, one simple OWL-S document.	8	12.79
4	One large NPE-L catalog, five complex OWL-S documents.	8x5	22.03
5	One large NPE-L catalog, five complex OWL-S documents.	8x5	20.75
6	One large NPE-L catalog, five complex OWL-S documents.	8	40.24

Table 1: Prototyping Results

		Time to finish					Score				
		Agent					Agent				
Iteration	Run	0	1	2	3	4	0	1	2	3	4
1	1	14.79	22.88	23.17	19.99	25.33	4	0	0	1	0
1	2	11.94	24.03	26.18	19.00	21.73	0	0	1	0	1
1	3	14.20	22.85	22.01	18.18	19.60	0	0	1	3	1
1	4	15.16	27.12	21.01	21.27	18.64	3	3	4	1	0
1	5	12.01	31.35	19.03	22.20	16.24	3	3	0	1	3
2	1	15.44	30.32	26.05	23.29	16.59	4	0	0	3	0
2	2	17.74	29.89	29.14	22.24	24.24	3	4	0	0	5
2	3	14.85	27.15	25.01	17.56	20.03	0	0	0	1	1
2	4	13.44	33.22	24.00	27.44	21.70	4	0	3	0	0
2	5	15.94	33.05	22.20	25.13	16.20	2	0	1	0	0
3	1	15.78	35.18	29.17	23.34	22.11	3	1	2	3	5
3	2	17.42	35.10	32.06	25.16	27.86	0	1	3	0	5
3	3	20.72	31.04	28.11	25.39	24.32	4	0	0	3	5
3	4	15.78	37.04	27.05	28.53	24.69	1	2	2	0	5
3	5	14.03	41.01	25.03	28.19	15.44	0	2	0	4	5
4	1	13.92	26.82	23.04	20.21	22.62	2	0	0	1	0
4	2	12.05	24.35	26.12	19.16	22.01	2	0	1	0	1
4	3	13.26	22.47	22.03	16.03	18.58	0	0	0	1	3
4	4	13.54	29.23	21.23	20.31	19.05	4	0	3	0	0
4	5	13.62	31.14	19.25	22.12	22.65	4	1	0	1	0
5	1	14.15	28.64	26.06	22.91	28.23	3	0	0	3	0
5	2	14.95	29.56	29.07	22.02	24.92	0	0	1	2	3
5	3	13.46	22.25	25.08	18.05	22.57	0	3	0	1	0
5	4	14.89	31.04	24.10	28.86	21.47	0	0	3	1	0
5	5	13.10	33.76	22.23	25.11	14.12	3	2	3	0	0
6	1	17.98	33.96	29.02	26.86	18.90	2	1	0	3	0
6	2	16.30	32.82	32.00	25.10	27.96	1	2	0	2	1
6	3	17.95	29.41	28.20	22.27	26.79	0	0	1	1	1
6	4	17.13	33.42	27.24	16.48	24.82	1	0	2	0	1
6	5	18.39	39.51	25.11	28.03	25.92	0	1	3	0	0
	Average	15.13	30.32	25.30	22.68	21.84	1.77	0.87	1.13	1.20	1.53
	Std dev	2.11	4.92	3.42	3.73	3.92	1.61	1.20	1.31	1.24	1.98
						Retrieved	30	30	30	30	24
						Relevant	11	18	14	15	17

Table 2: All Results for Simulation 0

		Time to finish					Score				
		Agent					Agent				
Iteration	Run	0	1	2	3	4	0	1	2	3	4
1	1	12.29	23.23	20.14	16.96	16.54	3	0	2	3	3
1	2	10.21	22.77	22.63	17.16	18.52	0	0	3	2	1
1	3	12.00	19.29	19.63	19.14	16.99	4	0	1	3	2
1	4	12.39	23.44	18.13	22.23	16.99	0	3	4	3	0
1	5	9.67	24.40	17.02	19.70	14.57	4	3	0	0	0
2	1	13.33	27.02	22.59	17.13	18.53	2	0	0	3	0
2	2	14.57	26.13	25.05	19.61	21.08	3	4	0	0	5
2	3	11.21	23.57	22.09	15.64	16.28	2	0	0	1	1
2	4	11.07	29.71	20.65	15.19	19.37	4	0	4	0	0
2	5	11.45	32.98	19.75	22.06	16.34	4	0	1	0	0
3	1	14.40	31.64	25.00	21.53	19.41	1	1	2	3	5
3	2	14.07	31.08	27.71	22.21	23.26	3	2	3	0	5
3	3	16.03	28.25	24.58	18.50	15.24	0	0	0	3	5
3	4	14.00	33.01	23.02	25.23	21.31	1	2	2	0	5
3	5	11.53	36.61	22.19	24.65	18.19	3	1	0	4	5
4	1	11.19	22.48	20.11	14.53	19.52	4	0	0	0	1
4	2	9.53	22.90	22.69	17.03	19.66	2	0	1	0	1
4	3	11.09	20.76	19.69	14.82	16.60	4	0	1	1	2
4	4	9.86	26.08	18.08	20.05	16.98	4	0	3	0	0
4	5	12.04	26.48	17.10	19.61	14.34	4	1	0	1	0
5	1	11.59	26.30	22.74	18.27	20.15	3	0	0	0	0
5	2	11.79	22.07	25.23	19.66	21.96	0	0	1	0	3
5	3	12.10	21.24	22.22	14.98	17.03	2	0	3	1	3
5	4	12.54	28.70	20.72	19.51	18.89	3	0	1	0	0
5	5	10.37	29.69	19.72	22.15	15.69	2	2	3	0	0
6	1	15.52	29.87	25.15	20.37	15.40	1	1	0	0	0
6	2	15.73	29.62	27.71	22.00	24.14	1	0	0	2	1
6	3	14.17	26.42	24.50	15.43	19.11	1	0	0	1	0
6	4	14.25	31.10	23.01	19.45	21.58	1	0	1	0	1
6	5	14.30	35.01	22.13	24.69	15.80	3	1	2	1	0
	Average	12.48	27.06	22.03	19.32	18.32	2.30	0.70	1.27	1.07	1.63
	Std dev	1.85	4.46	2.83	3.04	2.57	1.42	1.12	1.34	1.31	1.96
						Retrieved	30	30	30	30	24
						Relevant	6	21	14	19	16

Table 3: All Results for Simulation 1

		Time to finish					Score				
		Agent					Agent				
Iteration	Run	0	1	2	3	4	0	1	2	3	4
1	1	9.65	21.00	18.20	20.95	16.91	4	0	0	0	2
1	2	11.59	26.93	17.02	19.64	15.41	0	0	0	0	2
1	3	12.62	23.97	20.03	16.55	16.93	4	0	0	1	3
1	4	12.09	21.77	22.56	17.12	19.49	0	4	1	0	1
1	5	10.73	20.78	19.70	18.20	17.50	4	3	1	0	0
2	1	12.88	29.02	20.53	13.61	18.70	3	0	0	0	0
2	2	14.16	30.74	19.69	22.08	22.25	2	0	1	0	5
2	3	11.14	26.89	22.64	14.65	15.02	4	0	0	4	0
2	4	11.46	27.73	25.13	19.75	21.72	3	4	1	1	0
2	5	12.86	22.61	22.09	15.91	19.84	3	0	0	0	0
3	1	14.09	33.07	23.22	23.32	21.17	1	2	4	0	5
3	2	14.78	36.64	22.03	24.58	18.75	3	2	0	0	5
3	3	15.78	31.55	25.05	22.94	21.40	0	1	0	1	5
3	4	17.20	31.15	27.61	22.23	23.54	4	4	1	0	5
3	5	16.53	28.04	24.60	15.07	16.08	0	1	1	0	5
4	1	10.74	24.05	18.15	18.99	16.61	3	0	0	1	0
4	2	11.92	29.22	17.03	19.51	15.11	2	0	0	0	5
4	3	11.59	23.87	20.04	16.02	16.71	0	0	0	2	0
4	4	12.47	24.12	22.55	17.20	19.36	4	3	1	0	1
4	5	12.86	19.36	19.65	17.81	16.96	2	2	0	0	1
5	1	12.34	26.68	20.59	14.56	19.41	2	0	1	1	0
5	2	14.87	28.77	19.70	22.18	15.28	0	0	3	0	5
5	3	11.83	27.45	22.73	20.41	20.45	2	0	0	1	0
5	4	12.61	24.43	25.18	19.70	21.69	3	3	0	2	0
5	5	10.96	23.82	22.14	14.97	17.87	2	3	0	0	4
6	1	16.37	29.85	23.13	17.97	21.39	1	0	4	0	0
6	2	16.42	33.54	22.22	24.68	23.09	3	0	0	0	5
6	3	14.81	29.65	25.12	19.77	15.00	2	1	0	4	0
6	4	15.70	30.92	27.70	22.19	24.08	1	3	1	0	1
6	5	15.22	27.97	24.68	24.38	20.62	2	3	1	0	4
	Average	13.28	27.19	22.02	19.23	18.94	2.13	1.30	0.70	0.60	2.13
	Std dev	2.06	4.16	2.84	3.25	2.75	1.41	1.51	1.12	1.10	2.21
						Retrieved	30	30	30	30	21
						Relevant	8	15	21	23	12

Table 4: All Results for Simulation 2

		Time to finish					Score				
		Agent					Agent				
Iteration	Run	0	1	2	3	4	0	1	2	3	4
1	1	12.00	23.24	22.63	17.14	19.35	3	3	1	0	0
1	2	11.38	17.96	19.55	16.32	17.55	0	0	0	3	4
1	3	13.58	24.11	18.02	20.93	17.14	4	3	3	1	2
1	4	13.16	26.39	17.10	19.72	16.05	4	0	2	0	0
1	5	12.27	23.20	20.15	16.69	19.58	0	3	0	2	0
2	1	15.21	24.80	25.02	19.57	20.86	2	3	0	1	0
2	2	14.90	22.53	22.14	14.66	18.83	3	0	0	0	4
2	3	14.39	26.50	20.61	17.01	19.06	4	0	0	3	1
2	4	12.72	33.08	19.52	22.07	19.51	4	0	0	0	0
2	5	13.02	26.38	22.68	14.31	15.46	3	3	0	1	0
3	1	16.51	31.11	27.64	22.20	23.44	1	3	1	0	5
3	2	16.73	28.22	24.52	20.20	19.18	3	0	0	0	5
3	3	15.56	33.20	23.20	27.18	21.22	2	0	4	2	5
3	4	16.48	36.71	22.17	24.58	20.47	3	0	1	1	5
3	5	14.54	31.51	25.22	20.59	14.53	3	0	2	4	5
4	1	12.24	21.65	22.57	17.14	19.48	4	0	0	0	0
4	2	10.87	20.57	19.51	19.40	14.73	3	0	0	0	3
4	3	11.35	23.80	18.01	17.61	16.44	0	0	0	1	0
4	4	11.64	29.66	17.21	19.50	15.57	0	0	1	1	0
4	5	12.89	23.78	20.20	15.77	16.83	4	2	0	1	0
5	1	10.93	27.54	25.22	19.57	21.16	3	2	0	3	3
5	2	12.94	22.06	22.20	15.72	19.76	2	2	3	0	2
5	3	11.77	28.01	20.53	15.36	19.43	0	0	0	1	3
5	4	12.51	31.09	19.60	22.13	14.39	0	0	3	0	0
5	5	11.57	25.09	22.55	17.74	17.30	0	3	0	4	0
6	1	17.30	27.03	27.74	22.13	23.85	2	0	0	1	0
6	2	14.35	27.36	24.53	24.28	19.77	0	1	0	0	4
6	3	15.65	31.61	23.07	20.72	21.83	2	0	2	1	1
6	4	14.91	34.00	22.20	24.51	14.95	3	0	2	0	0
6	5	15.68	29.69	25.06	19.24	25.02	0	1	0	0	0
	Average	13.63	27.06	22.01	19.47	18.76	2.07	0.97	0.83	1.03	1.73
	Std dev	1.91	4.45	2.84	3.24	2.83	1.55	1.30	1.21	1.25	2.03
						Retrieved	30	30	30	30	25
						Relevant	10	18	19	17	15

Table 5: All Results for Simulation 3

		Time to finish					Score				
		Agent					Agent				
Iteration	Run	0	1	2	3	4	0	1	2	3	4
1	1	11.73	28.21	17.06	19.69	19.30	4	0	2	1	0
1	2	11.09	21.53	20.14	15.55	17.84	3	0	0	4	1
1	3	11.22	21.46	22.55	17.18	19.11	4	3	3	2	1
1	4	11.49	20.79	19.55	15.19	17.47	0	0	1	0	2
1	5	9.56	26.19	18.10	19.84	17.14	4	0	0	0	0
2	1	12.12	29.34	19.51	22.19	14.69	4	0	1	0	0
2	2	12.63	27.77	22.69	16.90	14.58	2	0	0	0	5
2	3	12.22	24.77	25.08	19.73	21.57	4	4	0	1	0
2	4	10.02	24.65	22.15	16.76	18.94	0	0	3	0	0
2	5	10.80	27.14	20.57	22.36	19.27	2	1	3	0	0
3	1	13.13	36.61	22.21	24.53	14.73	2	3	0	0	5
3	2	11.63	31.52	25.01	21.69	16.01	2	0	0	2	5
3	3	14.10	31.08	27.52	22.07	24.32	4	4	0	1	5
3	4	13.16	28.07	24.66	23.60	14.59	3	0	1	2	5
3	5	11.56	33.23	23.12	22.45	22.20	2	2	2	2	5
4	1	11.19	27.40	17.16	19.55	15.20	3	0	0	1	0
4	2	7.80	24.11	20.21	16.25	16.67	2	0	0	3	1
4	3	10.12	21.65	22.56	17.23	19.53	4	0	0	1	0
4	4	9.62	21.23	19.64	17.11	15.49	0	0	1	0	2
4	5	10.31	25.63	18.22	14.82	17.04	2	1	1	0	0
5	1	10.51	32.29	19.63	22.24	14.67	2	0	0	0	0
5	2	11.55	26.21	22.50	17.68	16.97	0	0	0	0	1
5	3	12.20	25.35	25.17	19.63	21.32	0	3	0	3	0
5	4	12.39	22.62	22.04	18.77	17.50	3	2	3	0	4
5	5	10.27	25.87	20.56	19.91	19.01	3	0	3	0	3
6	1	14.52	33.29	22.15	24.60	22.50	1	0	2	1	0
6	2	14.08	28.50	25.05	19.33	17.56	0	0	0	3	1
6	3	14.14	28.63	27.52	22.18	24.50	0	3	0	3	1
6	4	13.63	26.33	24.72	20.78	20.63	1	0	1	0	0
6	5	14.52	31.71	23.04	14.98	22.11	2	1	1	0	3
	Average	11.78	27.11	22.00	19.49	18.42	2.10	0.90	0.93	1.00	1.67
	Std dev	1.66	4.06	2.81	2.89	2.94	1.47	1.37	1.14	1.23	1.99
						Retrieved	30	30	30	30	24
						Relevant	8	20	19	18	17

Table 6: All Results for Simulation 4

		Time to finish					Score				
		Agent					Agent				
Iteration	Run	0	1	2	3	4	0	1	2	3	4
1	1	10.06	19.00	19.69	15.08	16.29	4	0	1	3	4
1	2	10.03	22.45	18.21	20.21	16.30	3	0	1	3	2
1	3	10.96	27.36	17.18	19.59	16.61	0	3	2	1	0
1	4	10.44	23.04	20.09	16.84	19.57	0	0	2	0	0
1	5	9.89	20.19	22.70	17.03	19.14	0	3	3	0	1
2	1	13.00	23.15	22.24	20.83	14.48	3	0	0	0	0
2	2	13.94	27.00	20.68	14.90	19.30	0	0	1	0	1
2	3	10.16	30.62	19.60	22.14	15.01	2	0	0	3	0
2	4	10.94	28.08	22.53	20.48	15.98	3	0	1	3	0
2	5	10.26	26.88	25.00	19.71	20.97	2	3	0	1	0
3	1	13.01	28.24	24.52	17.50	14.67	1	0	0	4	5
3	2	14.80	33.20	23.10	25.74	21.35	1	2	4	3	5
3	3	14.56	36.59	22.13	24.60	22.08	2	1	0	1	5
3	4	14.15	31.70	25.20	23.04	17.33	1	0	1	0	5
3	5	13.72	31.05	27.57	22.21	24.02	0	1	3	0	5
4	1	11.73	19.35	19.65	17.27	17.98	2	0	1	0	0
4	2	9.61	25.82	18.04	20.74	16.92	0	0	3	0	5
4	3	11.28	26.49	17.05	19.53	14.16	4	0	0	1	0
4	4	11.02	24.67	20.08	15.43	15.82	4	0	1	1	0
4	5	9.51	23.48	22.68	17.15	19.28	3	2	0	0	0
5	1	10.31	23.43	22.02	15.21	17.49	2	2	0	0	0
5	2	10.53	29.21	20.68	17.00	19.28	2	0	4	0	2
5	3	10.92	30.62	19.63	22.03	15.84	2	0	0	3	0
5	4	11.99	27.10	22.72	18.79	17.75	0	3	1	1	0
5	5	11.30	24.76	25.14	19.68	22.02	0	3	1	0	0
6	1	13.70	27.15	24.57	23.68	20.82	3	3	0	0	0
6	2	14.77	30.92	23.12	23.55	21.85	0	0	3	0	1
6	3	14.16	34.17	22.07	24.68	23.89	2	0	2	1	0
6	4	15.66	28.31	25.11	20.44	14.09	1	0	1	2	0
6	5	14.83	28.65	25.24	22.08	23.56	0	3	0	0	0
	Average	12.04	27.09	21.94	19.91	18.46	1.57	0.97	1.20	1.03	1.37
	Std dev	1.93	4.28	2.68	3.09	3.03	1.38	1.30	1.27	1.30	2.04
						Retrieved	30	30	30	30	24
						Relevant	11	19	16	19	20

Table 7: All Results for Simulation 5

Appendix: SPSS Analysis

A.6 Oneway ANOVA (Time)

A.6.1 Descriptives

Time

	N	Mean	Std. Deviation	Std. Error	95% Confidence Interval for Mean		Minimum	Maximum
					Lower Bound	Upper Bound		
0	150	12.641049	1.9921514	.1626585	12.319634	12.962465	7.8045	17.2953
1	150	27.100997	4.2262941	.3450755	26.419124	27.782871	17.9561	36.7088
2	150	22.002678	2.7614839	.2254742	21.557138	22.448218	17.0213	27.7415
3	150	19.482632	3.0714773	.2507851	18.987077	19.978187	13.6117	27.1838
4	150	18.579297	2.7990501	.2285415	18.127696	19.030898	14.0900	25.0188
Total	750	19.961331	5.6123220	.2049330	19.559019	20.363642	7.8045	36.7088

Figure 1: Oneway ANOVA Descriptives (Time)

A.6.2 Test of Homogeneity of Variances

Time

Levene Statistic	df1	df2	Sig.
20.492	4	745	.000

Figure 2: Oneway ANOVA Test of Homogeneity of Variances (Time)

A.6.3 ANOVA

Time

	Sum of Squares	df	Mean Square	F	Sig.
Between Groups	16630.144	4	4157.536	444.897	.000
Within Groups	6961.977	745	9.345		
Total	23592.121	749			

Figure 3: Oneway ANOVA Test (Time)

A.6.4 Robust Tests of Equality of Means

Time

	Statistic[a]	df1	df2	Sig.
Welch	531.093	4	367.217	.000

a. Asymptotically F distributed.

Figure 4: Oneway ANOVA Robust Tests of Equality of Means (Time)

A.6.5 Multiple Comparisons

Dependent Variable: Time

Games-Howell

(I) Agent	(J) Agent	Mean Difference (I-J)	Std. Error	Sig.	95% Confidence Interval	
					Lower Bound	Upper Bound
0	1	-14.45995*	.3814903	.000	-15.509576	-13.410320
	2	-9.361629*	.2780223	.000	-10.125142	-8.598115
	3	-6.841583*	.2989163	.000	-7.662812	-6.020353
	4	-5.938248*	.2805156	.000	-6.708645	-5.167851
1	0	14.459948*	.3814903	.000	13.410320	15.509576
	2	5.0983193*	.4122083	.000	3.965871	6.230768
	3	7.6183653*	.4265797	.000	6.446909	8.789822
	4	8.5217000*	.4138941	.000	7.384683	9.658717
2	0	9.3616287*	.2780223	.000	8.598115	10.125142
	1	-5.098319*	.4122083	.000	-6.230768	-3.965871
	3	2.5200460*	.3372414	.000	1.594405	3.445687
	4	3.4233807*	.3210449	.000	2.542255	4.304507
3	0	6.8415827*	.2989163	.000	6.020353	7.662812
	1	-7.618365*	.4265797	.000	-8.789822	-6.446909
	2	-2.520046*	.3372414	.000	-3.445687	-1.594405
	4	.9033347	.3392998	.062	-.027941	1.834610
4	0	5.9382480*	.2805156	.000	5.167851	6.708645
	1	-8.521700*	.4138941	.000	-9.658717	-7.384683
	2	-3.423381*	.3210449	.000	-4.304507	-2.542255
	3	-.9033347	.3392998	.062	-1.834610	.027941

*. The mean difference is significant at the 0.05 level.

Figure 5: Oneway ANOVA Multiple Comparisons (Time)

A.6.6 Homogeneous Subsets

Tukey HSD[a]

Agent	N	Subset for alpha = 0.05			
		1	2	3	4
0	150	12.641049			
4	150		18.579297		
3	150		19.482632		
2	150			22.002678	
1	150				27.100997
Sig.		1.000	.079	1.000	1.000

Means for groups in homogeneous subsets are displayed.

a. Uses Harmonic Mean Sample Size = 150.000.

Figure 6: Oneway ANOVA Homogeneous Subsets (Time)

A.6.7 Means Plots

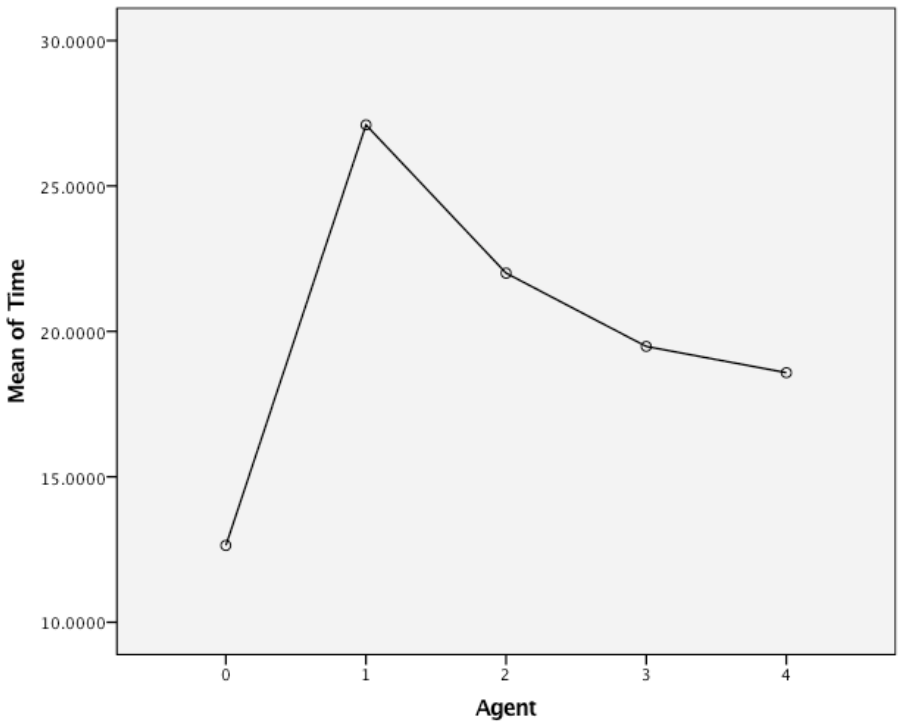

Figure 7: Oneway ANOVA Means Plots (Time)

A.7 Oneway ANOVA (Score)

A.7.1 Descriptives

Score

		N	Mean	Std. Deviation	Std. Error	95% Confidence Interval for Mean		Minimum	Maximum	Between-Component Variance
						Lower Bound	Upper Bound			
0		150	2.03	1.449	.118	1.80	2.27	0	4	
1		150	.97	1.323	.108	.75	1.18	0	4	
2		150	.99	1.221	.100	.79	1.18	0	4	
3		150	.95	1.236	.101	.75	1.15	0	4	
4		150	1.71	2.035	.166	1.38	2.04	0	5	
Total		750	1.33	1.548	.057	1.22	1.44	0	5	
Model	Fixed Effects			1.484	.054	1.22	1.43			
	Random Effects				.227	.70	1.96			.244

Figure 8: Oneway ANOVA Descriptives (Score)

A.7.2 Test of Homogeneity of Variances

Score

Levene Statistic	df1	df2	Sig.
30.126	4	745	.000

Figure 9: Oneway ANOVA Test of Homogeneity of Variances (Score)

A.7.3 ANOVA

Score

	Sum of Squares	df	Mean Square	F	Sig.
Between Groups	155.005	4	38.751	17.600	.000
Within Groups	1640.307	745	2.202		
Total	1795.312	749			

Figure 10: Oneway ANOVA Test (Score)

A.7.4 Robust Tests of Equality of Means

Score

	Statistic[a]	df1	df2	Sig.
Welch	18.431	4	370.416	.000
Brown-Forsythe	17.600	4	618.898	.000

a. Asymptotically F distributed.

Figure 11: Oneway ANOVA Robust Tests of Equality of Means (Score)

A.7.5 Multiple Comparisons

Dependent Variable: Score
Games-Howell

(I) Agent	(J) Agent	Mean Difference (I-J)	Std. Error	Sig.	95% Confidence Interval	
					Lower Bound	Upper Bound
0	1	1.067*	.160	.000	.63	1.51
	2	1.047*	.155	.000	.62	1.47
	3	1.087*	.155	.000	.66	1.51
	4	.327	.204	.498	-.23	.89
1	0	-1.067*	.160	.000	-1.51	-.63
	2	-.020	.147	1.000	-.42	.38
	3	.020	.148	1.000	-.39	.43
	4	-.740*	.198	.002	-1.28	-.20
2	0	-1.047*	.155	.000	-1.47	-.62
	1	.020	.147	1.000	-.38	.42
	3	.040	.142	.999	-.35	.43
	4	-.720*	.194	.002	-1.25	-.19
3	0	-1.087*	.155	.000	-1.51	-.66
	1	-.020	.148	1.000	-.43	.39
	2	-.040	.142	.999	-.43	.35
	4	-.760*	.194	.001	-1.29	-.23
4	0	-.327	.204	.498	-.89	.23
	1	.740*	.198	.002	.20	1.28
	2	.720*	.194	.002	.19	1.25
	3	.760*	.194	.001	.23	1.29

*. The mean difference is significant at the 0.05 level.

Figure 12: Oneway ANOVA Multiple Comparisons (Score)

A.7.6 Homogeneous Subsets

Tukey HSD[a]

Agent	N	Subset for alpha = 0.05	
		1	2
3	150	.95	
1	150	.97	
2	150	.99	
4	150		1.71
0	150		2.03
Sig.		.999	.315

Means for groups in homogeneous subsets are displayed.

a. Uses Harmonic Mean Sample Size = 150.000.

Figure 13: Oneway ANOVA Homogeneous Subsets (Score)

A.7.7 Means Plots

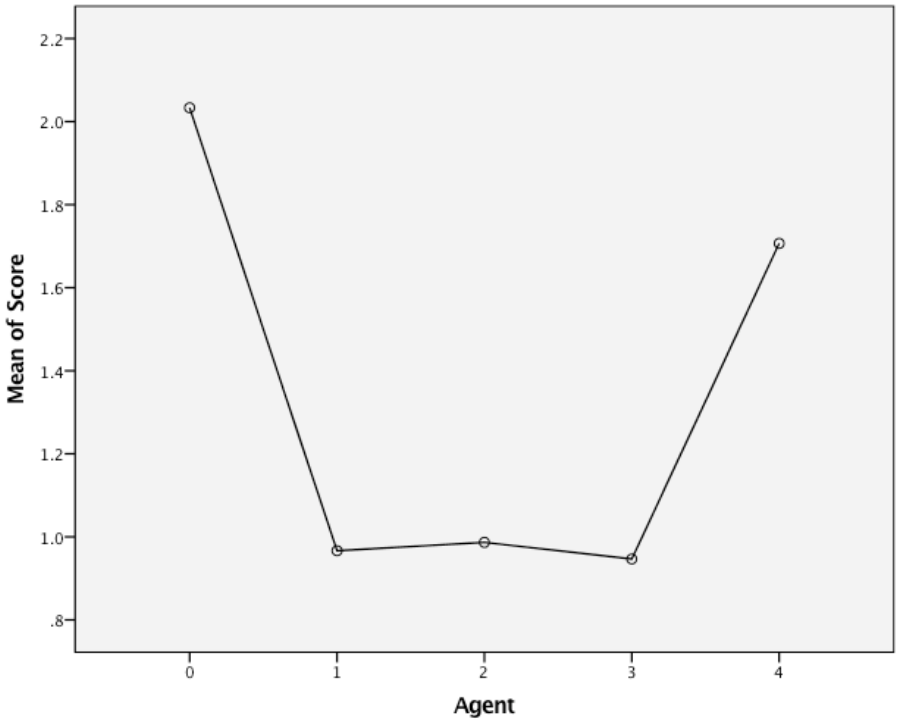

Figure 14: Oneway ANOVA Means Plots (Score)

A.8 Independent-Samples T-Test Control vs Mean NPM (Time)

A.8.1 Group Statistics

	HasNPE	N	Mean	Std. Deviation	Std. Error Mean
Time	Control	150	12.641049	1.9921514	.1626585
	NPE Based	600	21.791401	4.6502584	.1898460

Figure 15: Independent-Samples T-Test Group Statistics (Time)

A.8.2 Independent-Samples Test

		Levene's Test for Equality of Variances		t-test for Equality of Means						
									95% Confidence Interval of the Difference	
		F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Time	Equal variances assumed	70.859	.000	-23.556	748	.000	-9.1503518	.3884565	-9.9129466	-8.3877571
	Equal variances not assumed			-36.602	568.857	.000	-9.1503518	.2499986	-9.6413848	-8.6593189

Figure 16: Independent-Samples T-Test (Time)

A.9 Independent-Samples T-Test Control vs Mean NPM (Score)

A.9.1 Group Statistics

	HasNPE	N	Mean	Std. Deviation	Std. Error Mean
Score	Control	150	2.03	1.449	.118
	NPE Based	600	1.15	1.523	.062

Figure 17: Independent-Samples T-Test Group Statistics (Score)

A.9.2 Independent-Samples Test

		Levene's Test for Equality of Variances		t-test for Equality of Means						
									95% Confidence Interval of the Difference	
		F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Score	Equal variances assumed	.225	.635	6.403	748	.000	.882	.138	.611	1.152
	Equal variances not assumed			6.597	238.139	.000	.882	.134	.618	1.145

Figure 18: Independent-Samples T-Test (Score)

A.10 Independent-Samples T-Test Agent 3 vs Agent 4 (Time)

A.10.1 Group Statistics

	Agent	N	Mean	Std. Deviation	Std. Error Mean
Time	3	150	19.482632	3.0714773	.2507851
	4	150	18.579297	2.7990501	.2285415

Figure 19: Independent-Samples T-Test Group Statistics (Time)

A.10.2 Independent-Samples Test

		Levene's Test for Equality of Variances		t-test for Equality of Means						
									95% Confidence Interval of the Difference	
		F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Time	Equal variances assumed	.655	.419	2.662	298	.008	.9033347	.3392998	.2356074	1.5710619
	Equal variances not assumed			2.662	295.466	.008	.9033347	.3392998	.2355840	1.5710853

Figure 20: Independent-Samples T-Test (Time)

A.11 Independent-Samples T-Test Agent 3 vs Agent 4 (Score)

A.11.1 Group Statistics

	Agent	N	Mean	Std. Deviation	Std. Error Mean
Score	3	150	.95	1.236	.101
	4	150	1.71	2.035	.166

Figure 21: Independent-Samples T-Test Group Statistics (Score)

A.11.2 Independent-Samples Test

		Levene's Test for Equality of Variances		t-test for Equality of Means						
									95% Confidence Interval of the Difference	
		F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Score	Equal variances assumed	74.577	.000	-3.909	298	.000	-.760	.194	-1.143	-.377
	Equal variances not assumed			-3.909	245.740	.000	-.760	.194	-1.143	-.377

Figure 22: Independent-Samples T-Test (Score)

A.12 Independent-Samples T-Test Agent 0 vs Agent 3 (Time)

A.12.1 Group Statistics

	Agent	N	Mean	Std. Deviation	Std. Error Mean
Time	0	150	12.641049	1.9921514	.1626585
	3	150	19.482632	3.0714773	.2507851

Figure 23: Independent-Samples T-Test Group Statistics (Time)

A.12.2 Independent-Samples Test

		Levene's Test for Equality of Variances		t-test for Equality of Means						
									95% Confidence Interval of the Difference	
		F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Time	Equal variances assumed	25.700	.000	-22.888	298	.000	-6.8415827	.2989163	-7.4298369	-6.2533285
	Equal variances not assumed			-22.888	255.513	.000	-6.8415827	.2989163	-7.4302360	-6.2529293

Figure 24: Independent-Samples T-Test (Time)

A.13 Independent-Samples T-Test Agent 0 vs Agent 3 (Score)

A.13.1 Group Statistics

	Agent	N	Mean	Std. Deviation	Std. Error Mean
Score	0	150	2.03	1.449	.118
	3	150	.95	1.236	.101

Figure 25: Independent-Samples T-Test Group Statistics (Score)

A.13.2 Independent-Samples Test

		Levene's Test for Equality of Variances		t-test for Equality of Means						
									95% Confidence Interval of the Difference	
		F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Score	Equal variances assumed	6.621	.011	6.988	298	.000	1.087	.155	.781	1.393
	Equal variances not assumed			6.988	290.763	.000	1.087	.155	.781	1.393

Figure 26: Independent-Samples T-Test (Score)

References

A. A. De Moor and W. van den Heuvel. Web service selection in virtual communities. In *Proceedings of the 37th Annual Hawaii International Conference on System Sciences.*, 2004.

A. T. Acree, T. A. Budd, R. A. DeMillo, R. J. Lipton, and F. G. Sayward. Mutation analysis. Final report, Georgia Inst of Tech Atlanta School of Information and Computer Science, 1979.

T. Agotnes, W. V. D. Hoek, J. Rodriguez-Aguilar, C. Sierra, and M. Wooldridge. On the logic of normative systems. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pages 1175–1180, 2007.

R. Akkiraju, J. Farrell, J.Miller, M. Nagarajan, M. Schmidt, A. Sheth, and K. Verma. Web service semantics - wsdl-s. Technical Note 1, International Business Machines Corporation and University of Georgia, April 2005.

A. Al-Ajlan and H. Zedan. E-learning (moodle) based on service oriented architecture. White Paper, De Montfort University, April 2008.

N. Alechina, M. Dastani, and B. Logan. Programming norm-aware agents. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems - Volume 2*, AAMAS '12, pages 1057–1064, Richland, SC, 2012. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 0-9817381-2-5, 978-0-9817381-2-3.

V. Almeida. Capacity planning for web services. In *Performance Evaluation of Complex Systems: Techniques and Tools, Performance 2002, Tutorial Lectures*, pages 142–157, London, UK, UK, 2002. Springer-Verlag. ISBN 3-540-44252-9.

A. Alves, A. Arkin, S. Askary, C. Barreto, B. Bloch, F. Curbera, M. Ford, Y. Goland, N. Kartha, C. K. Liu, R. Khalaf, D. König, M. Marin, V. Mehta, S. Thatte, D. van der Rijn, P. Yendluri, and A. Yiu. Web services business process execution language version 2.0. Oasis standard, Oasis, April 2007.

Amazon.com. Amazon associates web services. Online (http://aws.amazon.com/associates/), March 2009a.

Amazon.com. What is aws? Online (http://aws.amazon.com/what-is-aws/), 2009b.

E. Ambrosi, M. Bianchi, C.Gaibisso, G.Gambosi, and F.Lombardi. Extending the uddi api for service instance ranking. In *Proceedings of the International Symposium on Web Services (ISWS05)*, College of Economics and Business Administration, Chongqing University, China, June 2005.

E. Andonoff, L. Bouzguenda, and C. Hanachi. Specifying web workflow services for finding partners in the context of loose inter-organizational workflow. In *International Conference on Business Process Management*, pages 120–136. Springer, 2005.

T. Andrews, F. Curbera, H. Dholakia, and Y. Goland. Business process execution language for web services. Specification 1.1, IBM, May 2003.

G. Andrighetto, M. Campenni, R. Conte, and M. Paolucci. On the immergence of norms: a normative agent architecture. In *Proceedings of AAAI Sympoisa Fall 2007*. AAAI, AAAI Press, 2007a.

G. Andrighetto, M. Campennì, F. Cecconi, and R. Conte. Normal = normative? the role of intelligent agents in norm innovation. In G. Boella, P. Noriega, G. Pigozzi, and H. Verhagen, editors, *Normative Multi-Agent Systems*, number 09121 in Dagstuhl Seminar Proceedings, Dagstuhl, Germany, 2009. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany.

G. Andrighetto, R. Conte, P. Turrini, and M. Paolucci. Emergence in the loop: Simulating the two way dynamics of norm innovation. In G. Boella, L. van der Torre, and H. Verhagen, editors, *Normative Multi-agent Systems*, number 07122 in Dagstuhl Seminar Proceedings, Dagstuhl, Germany, 2007b. Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany.

G. Andrighetto, D. Villatoro, and R. Conte. Norm internalization in artificial societies. *AI Commun.*, 23(4):325–339, December 2010. ISSN 0921-7126.

A. Ankolekar, D. Martin, D. McGuinness, S. McIlraith, M. P., and B. Parsia. Owl-s' relationship to selected other technologies. Member submission, W3C, November 2004.

M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, and I. Stoica. A view of cloud computing. *Communications of the ACM*, 43(4):50–58, 2010.

R. Ashri, M. Luck, and M. d'Inverno. From smart to agent systems development. *Eng. Appl. Artif. Intell.*, 18(2):129–140, March 2005. ISSN 0952-1976.

S. Awad, G. Decker, and M. Weske. Efficient compliance checking using bpmn-q and temporal logic. In *International Conference on Business Process Management*, pages 326–341. Springer, 2008.

W. Balke and M. Wagner. Towards personalized selection of web services. In *Proc. of the Int. World Wide Web Conf. (WWW2003)*, 2003.

S. Balzer, T. Liebig, and M. Wagner. Pitfalls of owl-s: a practical semantic web use case. In *Proceedings of the 2nd international conference on Service oriented computing*, pages 289–298. ACM, 2004.

L. Barakat, S. Miles, and M. Luck. Efficient correlation-aware service selection. In *Web Services (ICWS), 2012 IEEE 19th International Conference on*, pages 1–8. IEEE, IEEE Computer Society, 2012a.

L. Barakat, S. Miles, and M. Luck. Reactive service selection in dynamic service environments. In F. Paoli, E. Pimentel, and G. Zavattaro, editors, *Service-Oriented and Cloud Computing*, volume 7592 of *Lecture Notes in Computer Science*, pages 17–31. Springer Berlin Heidelberg, 2012b. ISBN 978-3-642-33426-9.

M. Barnett, M. Boesenhofer, D. Chavez, C. Eckenschwiller, S. Herrera, M. Kuhndt, M. Lehmann, V. Lindefjeld, C. Lugt, L. Plugge, G. Rynhart, and L. Tejlgård. Operational guide for medium-scale enterprises. Operational guide, United Nations, 2009.

A. Basu and R. Blanning. Workflow analysis using attributed metagraphs. In *System Sciences, 2001. Proceedings of the 34th Annual Hawaii International Conference on*, page 9. IEEE, 2001.

B. Beckert, U. Keller, and P. H. Schmitt. Translating the object constraint language into first-order predicate logic. In *In Proceedings, VERIFY, Workshop at Federated Logic Conferences (FLoC)*, pages 113 – 123, 2001.

F. Bellifemine, A. Poggi, and G. Rimassa. Jade: a fipa2000 compliant agent development environment. In *AGENTS '01: Proceedings of the fifth international conference on Autonomous agents*, pages 216 – 217, New York, USA, 2001. ACM.

T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, 284 (5):28–37, 2001.

C. Biagioli, P. Mariani, and D. Tiscornia. Esplex: A rule and conceptual based model for representing statutes. In *In The First International Conference on Artificial Intelligence and Law*, pages 240–251. ACM, May 1987.

J. Bloomberg. Case study zapnote: The hartford uddi. White Paper, ZapThink, March 2004.

B. Boehm. A spiral model of software development and enhancement. *Computer*, 21(5): 61–72, May 1988.

G. Boella and L. Lesmo. Deliberate normative agents. In *Social order in MAS*. Kluwer, 2001.

G. Boella and L. van der Torre. Regulative and constitutive norms in normative multiagent systems. In *In Procs. of KR'04*, pages 255–265. AAAI Press, 2004.

G. Boella and L. van der Torre. An architecture of a normative system: counts-as conditionals, obligations and permissions. In P. Stone and G. Weiss, editors, *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pages 229–231. ACM, 2006a.

G. Boella. Obligations and cooperation: Two sides of social rationality. In H. Hexmoor, C. Castelfranchi, and R. Falcone, editors, *Agent Autonomy*, volume 7 of *Multiagent Systems, Artificial Societies, and Simulated Organizations*, pages 75–102. Springer US, 2003. ISBN 978-1-4613-4833-7.

G. Boella, G. Pigozzi, and L. van der Torre. Normative framework for normative system change. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems - Volume 1*, AAMAS '09, pages 169–176, Richland, SC, 2009. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 978-0-9817381-6-1.

G. Boella and L. van der Torre. Introduction to normative multiagent systems. *Computational and Mathematical Organization Theory*, 12:71–79, 2006b.

G. Boella and L. van der Torre. Substantive and procedural norms in normative multiagent systems. *Journal of Applied Logic*, 6(2):152 – 171, 2008. ISSN 1570-8683.

C. Boettiger. An introduction to docker for reproducible research. *SIGOPS Oper. Syst. Rev.*, 49(1):71–79, January 2015.

O. Boissier, M. Colombetti, M. Luck, J.-J. Meyer, and A. Polleres. Norms, organizations, and semantics. *The Knowledge Engineering Review*, 28:107–116, 2 2013. ISSN 1469-8005.

W. J. Bolosky, J. R. Douceur, D. Ely, and M. Theimer. Feasibility of a serverless distributed file system deployed on an existing set of desktop pcs. In *Proceedings of the 2000 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pages 34–43. ACM, 2000.

P. A. Bonatti and P. Festa. On optimal service selection. In *Proceedings of the 14th international conference on World Wide Web*, pages 530 – 538. ACM, 2005.

A. Boot, T. Milbourn, and A. Schmeits. Credit ratings as coordination mechanisms. *Review of Financial Studies*, 19(1):81–118, 2006.

D. Booth, H. Haas, F. McCabe, E. Newcomer, M. Champion, C. Ferris, and D. Orchard. Web services architecture. Working group note, W3C, Febuary 2004.

R. H. Bordini, M. Wooldridge, and J. F. Hübner. *Programming Multi-Agent Systems in AgentSpeak Using Jason*. John Wiley and Sons, Ltd, 2007.

E. Bou, M. López-Sánchez, and J. A. Rodríguez-Aguilar. *Coordination, Organizations, Institutions, and Norms in Agent Systems III*, volume 4870 of *Lecture Notes in Computer Science*, chapter Using Case-Based Reasoning in Autonomic Electronic Institutions, pages 125–138. Springer Berlin / Heidelberg, 2008.

N. Boudriga and M. Obaidat. Intelligent agents on the web: a review. *Computing in Science and Engineering*, 6(4):35–42, July/August 2004.

G. Box. *Statistics for experimenters : an introduction to design, data analysis, and model building*. Wiley, New York ,USA, 1978.

T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler, and F. Yergeau. Extensible markup language (xml) 1.0 (fourth edition). Recommendation 4, W3C, September 2006.

M. L. Brodie. Illuminating the dark side of web services. In *Proceedings of the 29th international conference on VLDB*, volume 29, pages 1046–1049. VLDB Endowment, 2003.

J. Broekstra, A. Kampman, and F. V. Harmelen. Sesame: A generic architecture for storing and querying rdf and rdf schema. In *Proceedings of the First International Semantic Web Conference, ISWC2002*, pages 54–68. Springer Berlin / Heidelberg, 2002.

J. Broersen, M. Dastani, J. Hulstijn, Z. Huang, and L. van der Torre. The boid architecture: conflicts between beliefs, obligations, intentions and desires. In *Proceedings of the fifth international conference on Autonomous agents*, pages 9–16. SIGART, ACM, 2001.

J. Broersen, D. Gabbay, A. Herzig, E. Lorini, J. Meyer, X. Parent, and L. van der Torre. *Agreement Technologies*, chapter Chapter 10, Deontic Logic, pages 171–180. Number 8 in Law, Governance, and Technology Series 8. Springer Berlin Heidelberg, 2012.

J. Broersen and L. van der Torre. Ten problems of deontic logic and normative reasoning in computer science. In N. Bezhanishvili and V. Goranko, editors, *Lectures on Logic and Computation*, volume 7388 of *Lecture Notes in Computer Science*, pages 55–88. Springer Berlin Heidelberg, 2012.

A. Brogi, S. Corfini, and S. Iardella. *Service-Oriented Computing - ICSOC 2007 Workshops*, volume 4907 of *Lecture Notes in Computer Science*, chapter From OWL-S Descriptions to Petri Nets, pages 427–438. Springer Berlin / Heidelberg, 2009.

M. Brown and A. Forsythe. Robust tests for the equality of variances. *Journal of the American Statistical Association*, 69(346):364–367, 1974.

M. Buchheit, F. M. Donini, and A. Schaerf. Decidable reasoning in terminological knowledge representation systems. *Journal of Artificial Intelligence Research*, 1:109–138, 1993.

N. Bulling and M. Dastani. Normative programs and normative mechanism design. In *The 10th International Conference on Autonomous Agents and Multiagent Systems - Volume 3*, AAMAS '11, pages 1187–1188, Richland, SC, 2011. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 0-9826571-7-X, 978-0-9826571-7-1.

L. Cabral, J. Domingue, E. Motta, T. Payne, and F. Hakimpour. Approaches to semantic web services: an overview and comparisons. In *European Semantic Web Symposium*, pages 225–239. Springer, 2004.

G. Cabri, L. Ferrari, and L. Leonardi. Supporting the development of multi-agent interactions via roles. In J. P. Müller and F. Zambonelli, editors, *Agent-Oriented Software Engineering VI*, volume 3950 of *Lecture Notes in Computer Science*, pages 154–166. Springer Berlin Heidelberg, 2006. ISBN 978-3-540-34097-3.

M. Cambronero, G. Díaz, E. Martínez, and V. Valero. A comparative study between wsci, ws-cdl, and owl-s. In *2009 IEEE International Conference on e-Business Engineering*, pages 377–382. IEEE Press, October 2009.

J. Cardoso and A. Sheth. Semantic e-workflow composition. *Journal of Intelligent Information Systems*, 21(3):191–225, 2003.

C. Castelfranchi, F. Dignum, C. Jonker, and J. Treur. Deliberate normative agents:principles and architecture. In N. Jennings and Y. Lesperance, editors, *Intelligent Agents VI: Proceedings of The Sixth Inernational Workshop on Agent Theories, Architectures, and Languages (ATAL'99)*, pages 364–378, 1999.

C. Castelfranchi. Modeling social action for ai agents. In *Proceedings of the Fifteenth international joint conference on Artifical intelligence - Volume 2*, IJCAI'97, pages 1567–1576, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc. ISBN 1-555860-480-4.

A. Charfi and M. Mezini. Hybrid web service composition: Business processes meet business rules. In *Proceedings of the 2nd international conference on Service oriented computing, ICSOC '04*, pages 30–38, New York,USA, 2004. ACM.

S. Chatterjee, J. Webber, and D. Bunnell. *Developing enterprise Web services*. Prentice Hall, 2004.

G. Chen and W. Starosta. Chinese conflict management and resolution: Overview and implications. *Journal of Intercultural Communication Studies*, 7(1):1–16, 1998.

E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana. Web services description language (wsdl) 1.1. W3C Note 1.1, W3C, March 2001.

P. Chung, L. Cheung, J. Stader, P. Jarvis, J. Moore, and A. Macintosh. Knowledge-based process management—an approach to handling adaptive workflow. *Knowledge-Based Systems*, 16(2):149 – 160, April 2003.

E. Cimpian and A. Mocan. Wsmx process mediation based on choreographies. In *International Conference on Business Process Management*, pages 130–143. Springer, 2005.

E. Cimpian, A. Mocan, and M. Stollberg. Mediation enabled semantic web services usage. In *Asian Semantic Web Conference*, pages 459–473. Springer, 2006.

A. H. F. Civello, J. Howse, S. Kent, and R. Mitchell. *Selected papers from the First International Workshop on The Unified Modeling Language UML'98: Beyond the Notation*, volume 1618 of *Lecture Notes in Computer Science*, chapter Reflections on the Object Constraint Language, pages 162 – 172. Springer Berlin / Heidelberg, 1998.

R. Conte, N. Gilbert, G. Bonelli, C. Cioffi-Revilla, G. Deffuant, J. Kertesz, V. Loreto, S. Moat, J.-P. Nadal, A. Sanchez, A. Nowak, A. Flache, M. San Miguel, and D. Helbing. Manifesto of computational social science. *The European Physical Journal Special Topics*, 214:325–346, 2012. ISSN 1951-6355.

R. Conte and C. Castelfranchi. The mental path of norms. *Ratio Juris*, 19(4):501–517, 2006.

R. Conte and F. Dignum. From social monitoring to normative influence. *J. Artificial Societies and Social Simulation*, 4(2), 2001.

S. Cranefield. *Coordination, Organizations, Institutions, and Norms in Multi-Agent Systems*, volume 3913 of *Lecture Notes in Computer Science*, chapter A Rule Language for Modelling and Monitoring Social Expectations in Multi-agent Systems, pages 246–258. Springer Berlin / Heidelberg, May 2006.

S. Cranefield, S. Haustein, and M. Purvis. Uml-based ontology modelling for software agents. In *Proceedings of the Workshop on Ontologies in Agent Systems, 5th International Conference on Autonomous Agents*, 2001.

N. Criado, E. Argente, and V. Botti. Normative deliberation in graded bdi agents. In *Proceedings of the 8th German Conference on Multiagent System Technologies*, volume 6251 of *MATES'10*, pages 52–63, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

N. Criado, E. Argente, V. Botti, and P. Noriega. Reasoning about norm complience. In *In Proceedings of the tenth international conference on autonomouse agents and multiagent systems*, pages 1191–1192, 2011.

A. Cuzzocrea, J. Coi, M. Fisichella, and D. Skoutas. Graph-based matching of composite owl-s services. In J. Xu, G. Yu, S. Zhou, and R. Unland, editors, *Database Systems for Adanced Applications*, volume 6637 of *Lecture Notes in Computer Science*, pages 28–39. Springer Berlin Heidelberg, 2011. ISBN 978-3-642-20243-8.

V. T. da Silva. From the specification to the implementation of norms: an automatic approach to generate rules from norms to govern the behavior of agents. *Autonomous Agents and Multi-Agent Systems*, 17(1):113–155, August 2008.

P. Dabholkar, W. Johnston, and A. Cathey. The dynamics of long-term business-to-business exchange relationships. *Journal of the Academy of Marketing Science*, 22(2): 130–145, March 2008.

R. L. Dahl. Node.js (version 0.5.10). Online, October 2011.

M. Dastani. 2apl: a practical agent programming language. *Autonomous Agents and Multi-Agent Systems*, 16(3):214–248, June 2008. ISSN 1387-2532.

R. Davis and R. G. Smith. Negotiation distributed as a metaphor for problem solving. *Distributed Artificial Intelligence*, 20(1):63–109, January 1983.

J. de Bruijn, D. Fensel, M. Kifer, J. Kopecký, R. Lara, H. Lausen, A. Polleres, D. Roman, and J. Scicluna. Relationship of wsmo to other relevant technologies. Member submission, W3C, June 2005.

I. de Jong. Web services/soap and corba. Online, White Paper, April 2002.

A. de Moor. Language/action meets organisational semiotics: Situating conversations with norms. *Information Systems Frontiers*, 4(3):257–272, September 2002.

A. de Moor and M. A. Jeusfeld. Making workflow change acceptable. *Requirements Engineering*, 6(2):75 – 96, 2001.

K. Decker, K. Sycara, and M. Williamson. Middle-agents for the internet. In *Proceedings of the 15th International Joint Conference on Artificial Intelligence*, 1997, January 1997.

M. E. Delamaro, J. C. Maidonado, and A. P. Mathur. Interface mutation: An approach for integration testing. *Software Engineering, IEEE Transactions on*, 27(3):228 –247, mar 2001. ISSN 0098-5589.

R. A. DeMillo, R. J. Lipton, and F. G. Sayward. Hints on test data selection: Help for the practicing programmer. *Computer*, 11(4):34 –41, april 1978. ISSN 0018-9162.

L. deSilva and L. Padgham. Planning on demand in bdi systems. In *Proceeding of he International Conference on Automated Planning and Scheduling (ICAPS)*, June 2005.

M. Deutsch and H. B. Gerard. A study of normative and informational social influences upon individual judgment. *The Journal of Abnormal and Social Psychology*, 51(3): 629–636, 1955.

I. Dickinson and M. Wooldridge. Agents are not (just) web services: considering bdi agents and web services. Technical report, HP Digital Media Systems Laboratory, July 2005.

F. Dignum. Autonomous agents with norms. *Artificial Intelligence and Law*, 7(1):69–79, March 1999.

F. Dignum. Abstract norms and electronic institutions. In *Proceedings of International Workshop on Regulated Agent-Based Social Systems: Theories and Applications*, pages 93 – 104, 2002.

F. Dignum, D. Morley, E. Sonnenberg, and L. Cavedon. Towards socially sophisticated bdi agents. In *Proceedings of the Fourth International Conference on MultiAgent Systems*, page 111. IEEE Computer Society, 2000.

M. d'Inverno and M. Luck. *Understanding agent systems*. Springer-Verlag New York, Inc., New York, USA, 2001.

J. Domingue, L. Cabral, S. Galizia, V. Tanasescu, A. Gugliotta, B. Norton, and C. Pedrinaci. Irs-iii: A broker-based approach to semantic web services. *Journal of Web Semantics: Science, Services and Agents on the World Wide Web*, 6(2):109–132, April 2008.

J. Domingue, D. Roman, and M. Stollberg. Web service modeling ontology (wsmo)-an ontology for semantic web services. In *Position paper at the W3C Workshop on Frameworks for Semantics in Web Services*, pages 9–10, 2005.

A. Douglas, R. Walters, and G. WIlls. Npe - a conceptual model and language for the representation of norms. In *Quality of Service-Based Systems (in conjunction with 10th International IEEE Conference on Quality Software)*. IEEE, 2010.

N. Dragoni, S. Giallorenzo, A. Lluch-Lafuente, M. Mazzara, F. Montesi, R. Mustafin, and L. Safina. Microservices: yesterday, today, and tomorrow. *CoRR*, abs/1606.04036, 2016.

J. N. Drobak. *Norms and the Law*. Cambridge University Press, 2006.

S. Dustdar and M. Papazoglou. Services and service composition–an introduction (services und service komposition–eine einführung). *it-Information Technology*, 50(2/2008): 86–92, 2008.

Ebay.com. Shopping web services - ebay developer program. Online (http://developer.ebay.com/products/shopping/), March 2009.

M. Ehrig, A. Koschmider, and A. Oberweis. Measuring similarity between semantic business process models. In *Proceedings of the fourth Asia-Pacific conference on Comceptual modelling-Volume 67*, pages 71–80. Australian Computer Society, Inc., 2007.

R. C. Ellickson. Law and economics discovers social norms. *Journal of Legal Studies*, 27 (2):537–552, June 1998.

Z. Fayçal and T. Mohamed. A semantic web services for medical analysis using the owl-s language. *International Journal of Computer Applications (0975–8887,)*, 30(5), 2011.

C. H. Felicissimo, C. Lucena, G. Carvalho, and R. Paes. Normative ontologies to define regulations over roles in open multi-agent systems. Technical Report FS-05-08, AAAI Fall Symposium "Roles, an Interdisciplinary Perspective: Ontologies, Programming Languages, and Multiagent Systems", November 2005.

D. Fensel and C. Bussler. The web service modeling framework. *Electronic Commerce Research and Applications*, 1(2):113–137, August 2002.

J. Ferber, O. Gutknecht, and F. Michel. From agents to organizations: An organizational view of multi-agent systems. In P. Giorgini, J. P. Müller, and J. Odell, editors, *Agent-Oriented Software Engineering IV*, volume 2935 of *Lecture Notes in Computer Science*, pages 214–230. Springer Berlin Heidelberg, 2004. ISBN 978-3-540-20826-6.

A. Field. *Discovering statistics using IBM SPSS statistics*. Sage, 2013.

J. Forrester. Gentle murder, or the adverbial samaritan. *Journal of Philosophy*, 81: 193–197, 1984.

I. Foster. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Publishers Inc., 2 edition, 2004.

T. Gagnes, T. Plagemann, and E. Munthe-Kaas. Discovering semantic web services in dynamic discovering semantic web services in dynamic environments. In W. Löwe and J. Martin-Flatin, editors, *Proceedings of the 3rd IEEE European Conference on Web Services (IEEE ECOWS 2005)*. IEEE, November 2005.

Y. Ganjisaffar, H. Abolhassani, M. Neshati, and M. Jamali. A similarity measure for owl-s annotated web services. In *Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence*, WI '06, pages 621–624, Washington, DC, USA, 2006. IEEE Computer Society. ISBN 0-7695-2747-7.

A. García-Camino, J. A. Rodríguez-Aguilar, and W. W. Vasconcelos. A distributed architecture for norm management in multi-agent systems. In J. S. Sichman, J. Padget, S. Ossowski, and P. Noriega, editors, *Coordination, Organizations, Institutions, and Norms in Agent Systems III*, volume 4870 of *Lecture Notes in Computer Science*, pages 275–286. Springer Berlin Heidelberg, 2008. ISBN 978-3-540-79002-0.

M. Garriga, A. Flores, C. Mateos, A. Zunino, and A. Cechich. Service selection based on a practical interface assessment scheme. *International Journal of Web and Grid Services*, 9(4):369–393, 2013.

L. Gasparini, T. Norman, M. Kollingbaum, L. Chen, and J. Meyer. Co'ir : Verifying normative specifications of complex systems. In *Proceedings of the 19th International Workshop on Coordination, Organisations, Institutions and Norms.*, 2015.

R. Gasser and M. N. Huhns. *Distributed Artificial Intelligence*, volume 2 of *Research Notes in Artificial Intelligence*. Morgan Kaufmann Publishers Inc., May 2014.

J. Gerbrandy. *Logic, Language and Computation*, volume 2, chapter Dynamic epistemic logic, pages 67–84. Center for the Study of Language and Information, Stanford, CA, USA, 1999.

E. Gerding, K. Larson, and N. Jennings. Eliciting expert advice in service-oriented computing. In *Proceedings of the 11th International Workshop on Agent-Mediated Electronic Commerce (AMEC 2009)*, May 2009.

M. Ghallab, C. Knoblock, K. Golden, S. Penberthy, D. Smith, Y. Sun, D. Weld, and D. Mcdermott. The planning domain definition language. Technical report, AIPS-98 planning competition committee, 1998.

Y. Gil, E. Deelman, M. Ellisman, T. Fahringer, G. Fox, D. Gannon, C. Goble, M. Livny, L. Moreau, and J. Myers. Examining the challenges of scientific workflows. *IEEE Computer*, 40(12):26 – 34, December 2007.

M. Ginsberg. Knowledge interchange format: The kif of death. *AI Magazine*, 12(3): 57–63, 1991.

D. Gisolfi. Is web services the reincarnation of corba? Online article (http://www.ibm.com/developerworks/webservices/library/ws-arc3/), July 2001.

A. Gokhale, B. Kumar, and A. Sahuguet. Reinventing the wheel? corba vs. web services. In *Proceedings of The Eleventh International World Wide Web Conference*. World Wide Web Conference Comittee, May 2002.

F. Gonidis, I. Paraskakis, and A. Simons. A development framework enabling the design of service-based cloud applications. In G. Ortiz and C. Tran, editors, *Advances in Service-Oriented and Cloud Computing*, volume 508 of *Communications in Computer and Information Science*, pages 139–152. Springer International, 2015.

S. Gonzalo. A business outlook on electronic agents. *International Journal of Law and Information Technology*, 9(3):189–212, 2001.

Google. Google custom search api. Online (http://code.google.com/apis/customsearch/), March 2009.

Google.com. Google help center: I'm feeling lucky. Online (http://www.google.co.uk/help/features.html#lucky), March 2009.

G. Governatori and Z. Milosevic. Dealing with contract violations: formalism and domain specific language. In *Ninth IEEE International EDOC Enterprise Computing Conference (EDOC'05)*, pages 46–57. IEEE Computer Society, 2005.

T. R. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2):199–220, June 1993.

M. Gudgin, M. Hadley, N. Mendelsohn, J. Moreau, H. F. Nielsen, A. Karmarkar, and Y. Lafon. Soap version 1.2 part 1: Messaging framework. Recommendation 2, W3C, April 2007.

C. Gülcü. Simple logging facade for java (version 1.6.4). Online, October 2011.

R. Guo, D. Dong, and J. Le. Discovery for web services based on relationship model. In *Proceedings of the Sixth IEEE International Conference on Computer and Information Technoligy, CIT06*, pages 253–253. IEEE Computer Society, 2006.

A. Haller, E. Cimpian, A. Mocan, E. Oren, and C. Bussler. Wsmx-a semantic service-oriented architecture. In *IEEE International Conference on Web Services (ICWS'05)*, pages 321–328. IEEE, 2005.

A. Hamie, J. Howse, and S. Kent. Interpreting the object constraint language. In *Software Engineering Conference, 1998. Proceedings.*, pages 288 – 295. IEEE Computer Society, 1998.

S. Hansson. Deontic diversity. In C. F, D. Grossi, J. Meheus, and X. Parent, editors, *Deontic Logic and Normative Systems*, volume 8554 of *Lecture Notes in Computer Science*, pages 5–18. Springer International Publishing, 2014.

S. Hashemian and F. Mavaddat. A graph-based approach to web services composition. In *Applications and the Internet, 2005. Proceedings. The 2005 Symposium on*, pages 183–189, 2005.

M. Hechter and K. Opp. *Social Norms*. Russell Sage Foundation, illustrated edition, 2005.

I. Herman. Web ontology language (owl). Online (http://www.w3.org/2004/OWL/), October 2007.

B. Hindess. *Choice, rationality, and social theory*. Routledge, 1988.

G. Holmstrom-Hintikka, R. Sliwinski, and S. Lindstrom. *Collected Papers of Stig Kanger with Essays on his Life and Work*. Springer US, 2009.

C. W. Holsapple and K. D. Joshi. A collaborative approach to ontology design. *Communications of the ACM*, 45(2):42–47, Febuary 2002.

I. Horrocks, P. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, and M. Dean. Swrl: A semantic web rule language combining owl and ruleml. W3c member submission, W3C, May 2004.

I. Horrocks and U. Sattler. Ontology reasoning in the shoq(d) description logic. In *In Procceedings of the 17th Int. Joint Conf. on Artificial Intelligence (IJCAI 2001)*, 2001.

J. Hoskins. Real world soa stories. White Paper, IBM Global Technology Services, October 2008.

M. N. Huhns and M. P. Singh. Service-oriented computing: key concepts and principles. *IEEE Internet Computing*, 9(1):75–81, January 2005.

IBM. Ibm websphere® application server. Online (http://www.ibm.com/software/webservers/appserv/was/index.html), 2009.

D. Jackson. Alloy: A lightweight object modelling notation. *ACM Transactions on Software Engineering And Methodology*, 11(2):256 – 290, 2002.

I. Jacobs. About the world wide web consortium (w3c). Online article (http://www.w3.org/Consortium/), April 2008.

N. R. Jennings. On agent-based software engineering. *Artificial Intelligence*, 117(2): 277–296, March 2000.

S. Joosten. Trigger modelling for workflow analysis. In *Proceedings of CON '94:Workflow Management*, pages 236–247, Vienna, Munchen, 1994. R. Oldenbourg.

M. Kandori. Social norms and community enforcement. *The Review of Economic Studies*, 59(1):63–80, January 1992.

A. Karande and D. Kalbande. Web service selection based on qos using tmodel working on feed forward network. In *Issues and Challenges in Intelligent Computing Techniques (ICICT), 2014 International Conference on*, pages 29–33. IEEE, IEEE Press, Febuary 2014.

N. Kavantzas, D. Burdett, G. Ritzinger, T. Fletcher, Y. Lafon, and C. Barreto. Web services choreography description language version 1.0. Candidate recommendation, W3C, November 2005.

A. Keller and H. Ludwig. The wsla framework: Specifying and monitoring service level agreements for web services. *Journal of Network and Systems Management*, 11(1): 57–81, 2003. ISSN 1573-7705.

J. O. Kephart and W. E. Walsh. An artificial intelligence perspective on autonomic computing policies. In *Proceedings of 5th IEEE International Workshop on Policies for Distributed Systems and Networks*, pages 3–12. IEEE, IEEE, June 2004.

M. E. Kharbili, A. de Medeiros, S. Stein, and W. van der Aalst. Business process compliance checking: Current state and future challenges. *MobIS*, 8:107–113, 2008.

J. Kim, J. Lee, and B. Lee. Runtime service discovery and reconfiguration using owl-s based semantic web service. In *Proceedings of the 7th IEEE International Conference on Computer and Information Technology*, pages 891–896. IEEE Computer Society, 2007.

A. King. The united nations human rights norms for business and the un global compact. Online www.kingzollinger.ch/pdf/un

M. Klusch. Overview of the s3 contest: Performance evaluation of semantic service matchmakers. In *Semantic web services*, pages 17–34. Springer, 2012.

M. Klusch, B. Fries, M. Khalid, and K. Sycara. Owls-mx: Hybrid owl-s service matchmaking. In *Proceedings of 1st Intl. AAAI Fal l Symposium on Agents and the Semantic Web*, 2005.

M. Klusch and A. Gerber. Fast composition planning of owl-s services and application. In *Proceedings of the European Conference on Web Services*, pages 181–190. IEEE Computer Society, 2006.

M. Klusch, P. Kapahnke, and B. Fries. Hybrid semantic web service retrieval: A case study with owls-mx. In *Proceedings of the Second IEEE International Conference on Semantic Computing*, pages 323–330. IEEE Computer Society, August 2008.

R. Kohavi. Mining e-commerce data: the good, the bad, and the ugly. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 8–13, San Francisco, CA, USA, 2001. SIGMOD, ACM.

A. Kolber. Defining business rules what are they really? Technical report, The Business Rules Group, July 2000.

M. Kollingbaum and T. Norman. Noa - a normative agent architecture. In G. Gottlob and T. Walsh, editors, *Eighteenth International Joint Conference on Artificial Intelligence IJCAI'03*, pages 1465–1466. Morgan Kaufmann Publishers Inc., 2003.

V. Kolovski, B. Parsia, and E. Sirin. Extending the shoiq(d) tableaux with dl-safe rules: First results. In *Description Logics*, volume 189 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2006.

J. Kopeckỳ and T. Vitvar. Wsmo-lite: Lowering the semantic web services barrier with modular and light-weight annotations. In *Semantic Computing, 2008 IEEE International Conference on*, pages 238–244. IEEE, 2008.

J. Kopecký, T. Vitvar, C. Bournez, and J. Farrell. Sawsdl: Semantic annotations for wsdl and xml schema. *Internet Computing, IEEE*, 11(6):60–67, November 2007.

J. Kopeckỳ, T. Vitvar, D. Fensel, M. Maleshkova, and B. B. UIBK. Microwsmo and hrests. *SOA4All Project Deliverable D*, 2009.

R. Kramer. *Organizational trust*. Oxford University Press, illustrated edition, 2006.

H. Kreger. Fulfilling the web services promise. *Communications of the ACM, SPECIAL ISSUE: E-services*, 46(6):29–ff, June 2003.

R. Krummenacher, B. Norton, and A. Marte. Towards linked open services and processes. In *Future internet symposium*, pages 68–77. Springer, 2010.

U. Küster and B. König-Ries. Measures for benchmarking semantic web service matchmaking correctness. In *Extended Semantic Web Conference*, pages 45–59. Springer, 2010.

T. Kvaløy, E. Rongen, A. Tirado-Ramos, and P. Sloot. Automatic composition and selection of semantic web services. In *European Grid Conference*, pages 184–192. Springer, 2005.

M. Kwan and P. Balasubramanian. Adding workflow analysis techniques to the is development toolkit. In *System Sciences, 1998., Proceedings of the Thirty-First Hawaii International Conference on*, volume 4, pages 312–321. IEEE, 1998.

M. Lanthaler and C. Gütl. Hydra: A vocabulary for hypermedia-driven web apis. *Linked Data on the Web Workshop*, 996, 2013.

R. Lara, D. Roman, A. Polleres, and D. Fensel. *Web Services*, volume 3250 of *Lecture Notes in Computer Science*, chapter A Conceptual Comparison of WSMO and OWL-S, pages 254–269. Springer Berlin / Heidelberg, September 2004.

M. Laukkanen and H. Helin. *Extending Web Services Technologies*, volume 13 of *Multiagent Systems, Artificial Societies, and Simulated Organizations*, chapter Composing Workflows of Semantic Web Services, pages 1568–2617. Springer US, March 2006.

C. Lee, S. Park, D. Lee, J. Lee, O. Jeong, and S. Lee. A comparison of ontology reasoning systems using query sequences. In *Proceedings of the 2nd international conference on Ubiquitous information management and communication*, pages 543 – 546. ACM, 2008a.

S. Lee, X. Bai, and Y. Chen. Automatic mutation testing and simulation on owl-s specified web services. *AnnualSimulation Symposium*, 0:149–156, 2008b.

R. M. Levich, G. Majnoni, and C. M. Reinhart. *Ratings, Rating Agencies and the Global Financial System*. Springer US, 2002.

K. Leyking, R. Angeli, and I. AG. Model-based competency-oriented business process analysis. In *MobIS 2008*, pages 39–57, 2008.

J. Li, Y. Fan, and M. Zhou. Timing constraint workflow nets for workflow analysis. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 33(2): 179–193, 2003.

Y. Liu, S. Muller, and K. Xu. A static compliance-checking framework for business process models. *IBM Systems Journal*, 46(2):335–361, 2007.

A. Lomuscio, H. Qu, M. J. Sergot, and M. Solanki. *Service-Oriented Computing – ICSOC 2007*, chapter Verifying Temporal and Epistemic Properties of Web Service Compositions, pages 456–461. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2007.

A. Lomuscio, H. Qu, and M. Solanki. Towards verifying compliance in agent-based web service compositions. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems*, pages 265–272. ACM, 2008.

F. Lopez y Lopez and M. Luck. Modelling norms for autonomous agents. In *Proceedings of the Fourth Mexican International Conference on Computer Science*, page 236. IEEE Computer Society, 2003.

F. Lopez y Lopez, M. Luck, and M. d'Inverno. A framework for norm-based inter-agent dependence. In *Proceedings of 3rd Mexican International Conference on Computer Science*, pages 15–19, September 2001.

F. Lopez y Lopez, M. Luck, and M. d'Inverno. Constraining autonomy through norms. In *AAMAS '02: Proceedings of the first international joint conference on Autonomous agents and multiagent systems*, pages 674–681, New York, NY, USA, 2002. ACM. ISBN 1-58113-480-0.

F. Lopez y Lopez, M. Luck, and M. D'Inverno. A normative framework for agent-based systems. *Comput. Math. Organ. Theory*, 12(2-3):227–250, October 2006. ISSN 1381-298X.

F. Lopez y Lopez and A. A. Marquez. An architecture for autonomous normative agents. In *Proceedings of the Fifth Mexican International Conference in Computer Science*, volume 20, pages 96–103. IEEE, IEEE Computer Society, 2004.

M. Luck, S. Mahmoud, F. Meneguzzi, M. Kollingbaum, T. Norman, N. Criado, and M. Fagundes. *Agreement Technologies*, chapter Chapter 14, Normative Agents, pages 209–220. Number 8 in Law, Governance, and Technology Series 8. Springer Berlin Heidelberg, 2012.

A. Ludwig. Clonal selection based genetic algorithm for workflow service selection. In *2012 IEEE Congress on Evolutionary Computation*, pages 1–7. IEEE, 2012.

J. Luo, B. Montrose, A. Kim, A. Khashnobish, and M. Kang. Adding owl-s support to the existing uddi infrastructure. In *Proceedings of the IEEE International Conference on Web Services*, pages 153–162. IEEE Computer Society, 2006.

W. Luo and Y. A. Tung. A framework for selecting business process modeling methods. *Industrial Management & Data Systems*, 99(7):312–319, 1999.

S. Macaulay. Non-contractual relations in business: A preliminary study. *American Sociological Review*, 28(1):55–67, Febuary 1963.

D. Maharry, C. Ullman, K. Watson, and D. Foggon. *Programming Microsoft® .NET XML Web Services*. Pro-Developer. Microsoft Press, 2003.

E. Mally. *Grundgesetze des Sollens: Elemente der Logik des Willens*. Universitäts-Buchhandlung, 1926.

J. Márquez. About oasis. Online article (http://www.oasis-open.org/who/), 2008.

D. Martin, M. Burstein, J. Hobbs, O. Lassila, D. McDermott, S. McIlraith, S. Narayanan, M. Paolucci, B. Parsia, T. Payne, E. Sirin, N. Srinivasan, and K. Sycara. Owl-s: Semantic markup for web services. Member submission, W3C, November 2004.

V. Mascardi, M. Martelli, and L. Sterling. Logic-based specification languages for intelligent software agents. *Theory and Practice of Logic Programming*, 4(4):429–494, July 2003.

Mashape. Mashape the free online api marketplace. Online (https://www.mashape.com/), June 2015.

E. M. Maximilien and M. P. Singh. Toward autonomic web services trust and selection. In *Proceedings of the 2nd international conference on Service oriented computing*, pages 212 – 221. ACM, ACM, 2004.

B. McBride. Jena: A semantic web toolkit. *Internet Computing, IEEE*, 6(6):55–59, November 2002.

J. McCarthy. *Semantic Information Processing*, chapter Situations, Actions and Causal Laws. The MIT Press, 1969.

L. McCarty. A language for legal discourse i: Basic features. In *In The Second International Conference on Artificial Intelligence and Law*, pages 180–189. Association for Computing Machinery, 1989.

E. McCrum-Gardner. Which is the correct statistical test to use? *British Journal of Oral and Maxillofacial Surgery*, 46(1):38–41, 2008.

D. L. McGuinness and F. van Harmelen. Deborah l. mcguinness (knowledge systems laboratory, stanford university) d l m at k s l dot stanford dot edu owl web ontology language overview. W3C Recommendation 1.0, W3C, Febuary 2004.

S. McIlraith, T. Son, and Z. Honglei. Semantic web services. *IEEE Intelligent Systems*, 16(2):1541–1672, March 2001.

B. McLaughlin and J. Edelson. *Java and XML*. O'Reilly Media, Inc., 2006.

P. McNamara and H. Prakken. *Norms, Logics and Information systems: new studies in deontic logic and computer science*, volume 49. IOS press, 1999.

R. Mead. *The design of experiments : statistical principles for practical applications*. Cambridge University Press, 1990.

F. Meneguzzi, O. Rodrigues, N. Oren, W. Vasconcelos, and M. Luck. Bdi reasoning with normative considerations. *Engineering Applications of Artificial Intelligence*, 43(1):127 – 146, 2015.

F. Meneguzzi, W. Vasconcelos, N. Oren, and M. Luck. Nu-bdi: Norm-aware bdi agents. In *Proceedings of the 10th European Workshop on Multi-Agent Systems*, 2012.

F. Meneguzzi and M. Luck. Norm-based behaviour modification in bdi agents. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems - Volume 1*, AAMAS '09, pages 177–184, Richland, SC, 2009. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 978-0-9817381-6-1.

F. Meneguzzi, N. Oren, and W. Vasconcelos. Using constraints for norm-aware BDI agents. In *The Fourth Annual Conference of the International Technology Alliance*, London, UK, 2010.

H. Meyer. Beware when you compare. Guardian News Paper, June 2007.

J. J. C. Meyer, F. P. M. Dignum, and R. J. Wieringa. The paradoxes of deontic logic revisited: A computer science perspective (or: Should computer scientists be bothered by the concerns of philosophers?). Technical Report UU-CS-1994-38, Department of Information and Computing Sciences, Utrecht University, 1994.

H. Miao, T. He, and Z. Qian. Modeling and analyzing composite semantic web service using petri nets. In *ICEBE '08. IEEE International Conference on e-Business Engineering*, pages 600–664. IEEE Computer Society, October 2008.

Microsoft. Case study: Thomson financial. White Paper, Microsoft Corperation, November 2006.

Microsoft. Microsoft soa. Online (http://www.microsoft.com/soa/default.aspx), 2007.

Microsoft. Uddi services: Qwest technical case study. White Paper, Microsoft Corperation, December 2008.

Microsoft. Windows server 2003 uddi services. Online (http://uddi.microsoft.com/), 2009.

Microsoft. The microsoft azure marketplace. Online (http://azure.microsoft.com/en-gb/marketplace/), June 2015.

N. Milanovic and M. Malek. Current solutions for web service composition. *Internet Computing, IEEE*, 8(6):51–59, November 2004.

D. Miller. The norm of self-interest. *American Psychologist*, 54(12):1053 – 1060, December 1999.

Mindswap. Owl-s @ mindswap. Online (http://www.mindswap.org/2004/owl-s/services.shtml), 2004.

W. H. Money and S. Cohen. Developing a marketplace for smart cities foundational services with policy and trust. *International Journal of Computer Science: Theory and Application*, 3(1):1–12, Febuary 2015.

MoneySupermarket.com. Annual report 2008. Online, Febuary 2008.

J. Morales, M. Lopez-Sanchez, J. A. Rodriguez-Aguilar, M. Wooldridge, and W. Vasconcelos. Synthesising liberal normative systems. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '15, pages 433–441, Richland, SC, 2015a. International Foundation for Autonomous Agents and Multiagent Systems.

J. Morales, I. Mendizabal, D. Sánchez-Pinsach, J. A. Rodriguez-Aguilar, M. López-Sánchez, M. Wooldridge, and W. Vasconcelos. Extending normlab to spur research on norm synthesis. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '15, pages 1931–1932, Richland, SC, 2015b. International Foundation for Autonomous Agents and Multiagent Systems.

L. Moreau. Agents for the grid: A comparison for web services (part 1: the transport layer). In *Second IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID 2002)*, pages 220 – 228. IEEE, IEEE Computer Society, May 2002.

T. Murnane and K. Reed. On the effectiveness of mutation analysis as a black box testing technique. *Software Engineering Conference, Australian*, 0:0012, 2001.

M. Najafi, K. Sartipi, and N. Archer. Web service competition: A new approach to service selection. In *Proceedings of the 2012 Conference of the Center for Advanced Studies on Collaborative Research*, CASCON '12, pages 161–175, Riverton, NJ, USA, 2012. IBM, IBM Corperation.

S. Narayanan. Owl-s 1.1 release: Examples. Online (http://www.daml.org/services/owl-s/1.1/examples.html), Febuary 2005.

S. Narayanan and S. McIlraith. Analysis and simulation of web services. *Computer Networks*, 42(5):675–693, 2003.

F. Nawaz, M. Pasha, H. F. Ahmad, and H. Suguri. Pushing semantic web service profiles to subscribers for efficient service discovery. In *Proceedings of the Third International Conference on Semantics Knowledge and Grid*, pages 503–506. IEEE Computer Society, October 2007.

NCUB. Universities and business forge stronger partnership – launch of new national centre announced. Online (http://www.ncub.co.uk/press-releases/universities-and-business-forge-stronger-partnership-launch-of-new-national-centre-announced.html), March 2013.

S. Newman. *Building Microservices*. O'Reilly Media, Inc., 2015.

B. Newnam. Are you feeling lucky? google is. Online article (http://marketplace.publicradio.org/display/web/2007/11/19/face_of_google/), 2007.

P. S. O'Donnell. Social norms & law: An introduction. *Theory & Science*, 9(2), 2007.

U. of Basel. Owl-s api introduction. Online (http://on.cs.unibas.ch/owls-api), March 2010.

J. Offutt, P. Ammann, and L. Liu. Mutation testing implements grammar-based testing. In *Second Workshop onMutation Analysis, 2006.*, pages 12–24, 7-10 2006.

Oracle-Consulting. Luton borough council leverages web services to reduce costs and improve customer service. White Paper, Oracle, December 2008.

Oracle-Consulting. Hypovereinsbank ag reduces administrative burden by 50service-oriented content management. White Paper, Oracle, Febuary 2009.

N. Oren, M. Luck, and S. Miles. A model of normative power. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1 - Volume 1*, AAMAS '10, pages 815–822, Richland, SC, 2010. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 978-0-9826571-1-9.

N. Oren, W. Vasconcelos, F. Meneguzzi, and M. Luck. Acting on norm constrained plans. In *Proceedings of the 12th international conference on Computational logic in multi-agent systems*, CLIMA'11, pages 347–363, Berlin, Heidelberg, 2011. Springer-Verlag. ISBN 978-3-642-22358-7.

A. Padovitz, S. Krishnaswamy, S. A. Padovitz, S. Krishnaswamy, and S. W. Loke. Towards efficient selection of web services. In *Proceedings of the Workshop on Web Services and Agent-based Engineering (WSABE 2003)*. ACM, July 2003.

F. Pan and J. R. Hobbs. Time in owl-s. In *Proceedings of AAAI-04 Spring Symposium on Semantic Web Services*, Stanford University, California, 2004. AAAI Press.

S. Panagiotidi and J. Vazquez-Salceda. Towards practical normative agents: A framework and an implementation for norm-aware planning. In S. Cranefield, M. Riemsdijk, J. Vzquez-Salceda, and P. Noriega, editors, *Coordination, Organizations, Institutions, and Norms in Agent System VII*, volume 7254 of *Lecture Notes in Computer Science*, pages 93–109. Springer Berlin Heidelberg, 2012. ISBN 978-3-642-35544-8.

Y. Panagis, K. Papakonstantinou, E. Sakkopoulos, and A. Tsakalidis. Web service workflow selection using system and network qos constraints. *International Journal of Web Engineering and Technology*, 4(1):114–134, 2007.

M. Paolucci, T. Kawamura, T. R. Payne, and K. Sycara. *The Semantic Web — ISWC 2002*, volume 2342 of *Lecture Notes in Computer Science*, chapter Semantic Matching of Web Services Capabilities, pages 333–347. Springer Berlin / Heidelberg, January 2002.

D. Parnas, P. Clements, and D. Weiss. The modular structure of complex systems. In *ICSE '84: Proceedings of the 7th international conference on Software engineering*, pages 408–417, Piscataway, NJ, USA, 1984. IEEE Press.

P. Patel-Schneider. A proposal for a swrl extension towards first-order logic. W3c member submission, W3C, April 2005.

D. Paulraj, S. Swamynathan, and M. Madhaiyan. Process model-based atomic service discovery and composition of composite semantic web services using web ontology language for services (owl-s). *Enterprise Information Systems*, 6(4):445–471, 2012.

M. Payton, M. Greenstone, and N. Schenker. Overlapping confidence intervals or standard error intervals: What do they mean in terms of statistical significance? *Journal of Insect Science*, 3(34), 2004.

P. Pettit. *Rules, Reasons, and Norms.* Oxford University Press, 2002.

F. Pirri and R. Reiter. Some contributions to the metatheory of the situation calculus. *Journal of the ACM*, 46(3):325–361, 1998.

A. Polleres, H. Lausen, D. Roman, J. de Bruijn, and D. Fensel. D4.1v0.1 a conceptual comparison between wsmo and owl-s. Working draft, WSMO, January 2005.

M. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

R. Posner. Social norms and the law: An economic approach. *The American Economic Review*, 87(2):365 – 369, May 1997.

A. Powles and S. Krishnaswamy. Extending uddi with recommendations: an association analysis approach. In *Proceedings of the 19th International Conference On Advanced Information Networking and Applications, 2005*, volume 2, pages 715–720. IEEE Computer Society, March 2005.

V. R. Pratt. Action logic and pure induction. In J. van Eijck, editor, *Proceedings of Logics in AI: European Workshop JELIA '90*, pages 97–120. Springer-Verlag, September 1990.

E. Prud'hommeaux and A. Seaborne. Sparql query language for rdf. Online (http://www.w3.org/TR/rdf-sparql-query/), January 2008.

S. Ran. A model for web services discovery with qos. *ACM SIGecom Exchanges*, 4(1): 1–10, Febuary 2003.

A. S. Rao and M. P. Georgeff. Bdi agents: from theory to practice. In *Proceedings of the First International Conference on Multi-Agent Systems ICMAS 95*, 1995.

H. Reijers, S. Limam, and W. V. D. Aalst. Product-based workflow design. *Journal of Management Information Systems*, 20(1):229–262, 2003.

P. Resnick, K. Kuwabara, R. Zeckhauser, and E. Friedman. Reputation systems. *Communications of the ACM*, 43(12):45–48, December 2000.

C. J. V. Rijsbergen. *Information Retrieval*. Butterworth-Heinemann, Newton, MA, USA, 2nd edition, 1979. ISBN 0408709294.

S. Robinson and D. Rousseau. Violating the psychological contract: Not the exception but the norm. *Journal of Organizational Behavior*, 15(3):245 – 259, May 1994.

J. A. Rodríguez-Aguilar. *On the Design and Construction of Agent-Mediated Electronic Institutions*. PhD thesis, CSIC: Barcelona, 2001.

S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, Upper Saddle River, NJ, 2 edition, 2003.

G. Ruxton and G. Beauchamp. Time for some a priori thinking about post hoc testing. *Behavioral Ecology*, 19(3):690–693, 2008.

M. Sabou, D. Richards, and S. V. Splunter. An experience report on using daml-s. In *Workshop on E-Services and the Semantic Web*, 2003.

P. Sargent. Back to school for a brand new abc. *The Guardian*, page 28, March 1992.

C. Schröpfer, M. Schönherr, P. Offermann, and M. Ahrens. *Emerging Web Services Technology*, chapter A Flexible Approach to Service Management-Related Service Description in SOAs, pages pp 47–64. Whitestein Series in Software Agent Technologies and Autonomic Computing. Birkhäuser Basel, 2007.

S. Sciaraffa. *Wired World*, chapter Review of Norms. Notre Dame Philosophical Reviews, 2005.

K. Segerberg. Getting started: Beginnings in the logic of action. *Studia Logica*, 51(3): 347–378, September 1992.

M. Sensoy, T. J. Norman, W. W. Vasconcelos, and K. Sycara. Owl-polar: Semantic policies for agent reasoning. In P. Patel-Schneider, Y. Pan, P. Hitzler, P. Mika, L. Zhang, J. Pan, I. Horrocks, and B. Glimm, editors, *The Semantic Web ,AI ISWC 2010*, volume 6496 of *Lecture Notes in Computer Science*, pages 679–695. Springer Berlin Heidelberg, 2010. ISBN 978-3-642-17745-3.

M. Sensoy, T. J. Norman, W. W. Vasconcelos, and K. Sycara. Owl-polar: A framework for semantic policy representation and reasoning. *Web Semant.*, 12-13:148–160, April 2012. ISSN 1570-8268.

A. Serenko and B. Detlor. Agent toolkits: A general overview of the market and an assessment of instructor satisfaction with utilizing toolkits in the classroom. Working Paper 455, July 2002.

A. Serenko and B. Detlor. Agent toolkit satisfaction and use in higher education. *Journal of Computing in Higher Education*, 15(1):65–88, September 2003.

M. Sergot and R. Craven. The deontic component of action language nc+. In *International Workshop on Deontic Logic and Artificial Normative Systems*, pages 222–237. Springer, 2006.

O. Shafiq, M. Moran, E. Cimpian, A. Mocan, M. Zaremba, and D. Fensel. Investigating semantic web service execution environments: a comparison between wsmx and owl-s tools. In *Internet and Web Applications and Services, 2007. ICIW'07. Second International Conference on*, pages 31–31. IEEE, 2007.

A. ShaikhAli, O. F. Rana, R. Al-Ali, and D. W. Walker. Uddie: an extended registry for web services. In *Proceeding of the 2003 Symposium on Applications and the Internet Workshops*, pages 25–29. IEEE Computer Society, January 2003.

P. Sheeran and S. Taylor. Predicting intentions to use condoms: A meta-analysis and comparison of the theories of reasoned action and planned behavior. *Journal of Applied Social Psychology*, 29(8):1624 – 1675, July 2006.

O. Shenkar and S. Ronen. The cultural context of negotiations: The implications of chinese interpersonal norms. *The Journal of Applied Behavioral Science*, 23(2), 1987.

C. Sierra. Agent-mediated electronic commerce. *Autonomous Agents and Multi-Agent Systems*, 9(3):285–301, November 2004.

A. Sinha. Client-server computing. *Communications of the ACM*, 35(7):77–98, 1992.

E. Sirin and B. Parsia. The owl-s java api. In *The Semantic Web - ISWC 2004*, Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2004.

E. Sirin, B. Parsiaa, B. C. Graua, A. Kalyanpura, and Y. Katz. Pellet: A practical owl-dl reasoner. *Web Semantics: Science, Services and Agents on the World Wide Web*, 5(2): 51 – 53, June 2007.

R. M. Sreenath and M. P. Singh. Agent-based service selection. *Web Semantics: Science, Services and Agents on the World Wide Web*, 1(3):261–279, April 2004.

N. Srinivasan. Owl-s 1.1 api – version 1.1. Online (http://www.daml.ri.cmu.edu/owlsapi/), December 2009.

S. Stein, N. R. Jennings, and T. Payne. Flexible service provisioning with advance agreements. In *Seventh International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS-08)*, May 2008.

R. Stoline. The status of multiple comparisons: simultaneous estimation of all pairwise comparisons in one-way anova designs. *The American Statistician*, 35(3):134–141, 1981.

K. Sycara, M. Klusch, S. Widoff, and J. Lu. Dynamic service matchmaking among agents in open information environments. *SIGMOD Rec.*, 28(1):47–53, 1999.

K. Sycara, M. Paolucci, J. Soudry, and N. Srinivasan. Dynamic discovery and coordination of agent-based semantic web services. *IEEE Internet Computing*, 8(3):66–73, June 2004.

B. Tabachnick, L. Fidell, and S. Osterlind. *Using multivariate statistics*. Allyn and Bacon Boston, 2001.

E. Tamani and P. Evripidou. *On the Move to Meaningful Internet Systems 2007: OTM 2007 Workshops*, chapter Combining Pragmatics and Intelligence in Semantic Web Service Discovery, pages 824–833. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2007.

R. H. Thomason. Desires and defaults: A framework for planning with inferred goals. In *In Proceedings of KR 2000*, pages 702–713. Morgan Kaufmann Publishers Inc., 2000.

J. Thones. Microservices. *Software, IEEE*, 32(1):116–116, Jan 2015.

T.Miller, P. McBurney, S. Munroe, and M. Luck. Intelligent fleet cargo scheduling - magenta technology and tankers international: A case study. Online (Case Study at http://www.agentlink.org/), N. Yakounina 2005.

E. Tonkin. Uddi and iesr:a. Report, Joint Information Systems Committee-funded IESR project. UKOLN, August 2005.

L. V. D. Torre. Contextual deontic logic: Normative agents, violations and independence. *Annals of Mathematics and Artificial Intelligence*, 37:33–63, 2003.

W. van der Aalst. Don't go with the flow: Web services composition standards exposed. *IEEE Intelligent Systems*, January 2003.

M. Y. Vardi. On epistemic logic and logical omniscience. In *TARK '86: Proceedings of the 1986 conference on Theoretical aspects of reasoning about knowledge*, pages 293–305, San Francisco, CA, USA, 1986. Morgan Kaufmann Publishers Inc.

W. W. Vasconcelos, A. García-Camino, D. Gaertner, J. A. Rodríguez-Aguilar, and P. Noriega. Distributed norm management for multi-agent systems. *Expert Systems with Applications*, 39(5):5990 – 5999, 2012. ISSN 0957-4174.

W. W. Vasconcelos, M. J. Kollingbaum, and T. J. Norman. Normative conflict resolution in multi-agent systems. *Autonomous Agents and Multi-Agent Systems*, 19(2):124–152, October 2009. ISSN 1387-2532.

M. Vaziri and D. Jackson. Some shortcomings of ocl, the object constraint language of uml. In *TOOLS archive Proceedings of the Technology of Object-Oriented Languages and Systems (TOOLS 34'00)*. IEEE Computer Society, 2000.

J. L. Vendetti. The protégé ontology editor and knowledge acqusition system. Online (http://protege.stanford.edu/), January 2009.

R. Verborgh, T. Steiner, J. Gabarro, E. Mannens, and R. van de Walle. A social description revolution-describing web apis' social parameters with restdesc. In *AAAI Spring Symposium: Intelligent Web Services Meet Social Computing*, 2012.

J. Vidal, M. Lama, and A. Bugarn. Toward the use of petri nets for the formalization of owl-s choreographies. *Knowledge and Information Systems*, 32(3):629–665, 2012. ISSN 0219-1377.

W. Viriyasitavat and L. D. Xu. Compliance checking for requirement-oriented service workflow interoperations. *IEEE Transactions on Industrial Informatics*, 10(2):1469–1477, 2014.

T. Vitvar, J. Kopeckỳ, J. Viskova, and D. Fensel. Wsmo-lite annotations for web services. In *European Semantic Web Conference*, pages 674–689. Springer, 2008.

T. Vitvar, J. Kopecky, M. Zaremba, and D. Fensel. Wsmo-lite: Lightweight semantic descriptions for services on the web. In *Web Services, 2007. ECOWS'07. Fifth European Conference on*, pages 77–86. IEEE, 2007.

G. H. von Wright. Deontic logic. *Mind*, 60:1–15, 1951.

X. Wang and C. Chan. Ontology modeling using uml. In *Proceedings of 7th International Conference on Object Oriented Information Systems*, pages 59–70, Calgary, Canada, 2001.

X. Wang, N. Huang, and R. Wang. Mutation test based on owl-s requirement model. *IEEE International Conference on Web Services*, 0:1006–1007, 2009.

H. Weigand and A. D. Moor. Workflow analysis with communication norms. *Data & Knowledge Engineering*, 47(3):349–369, 2003.

J. Wielemaker, M. Hildebrand, and J. van Ossenbruggen. Prolog as the fundament for applications on the semantic web. In *Proceedings of the ICLP'07 Workshop on Applications of Logic Programming to the Web, Semantic Web and Semantic Web Services, ALPSWS 2007*, volume 287. CEUR-WS.org, September 2007.

R. J. Wieringa and J. j. Ch. Meyer. Applications of deontic logic in computer science: A concise overview. In *Deontic Logic in Computer Science: Normative System Specification*, pages 17–40. John Wiley and Sons, Ltd Sons, 1993.

P. Wohed, W. M. P. van der Aalst, M. Dumas, and A. H. M. ter Hofstede. *Conceptual Modeling - ER 2003*, volume 2813 of *Lecture Notes in Computer Science*, chapter Analysis of Web Services Composition Languages: The Case of BPEL4WS, pages 200–215. Springer Berlin / Heidelberg, October 2003.

D. Wong, N. Paciorek, T. Walsh, J. DiCelie, M. Young, and B. Peet. Concordia: An infrastructure for collaborating mobile agents. In W. Brauer, D. Gries, and J. Stoer, editors, *Proceedings of the First International Workshop, MA '97 Berlin*, volume 1219 of *Lecture Notes in Computer Science*, pages 86–97. Springer-Verlag, 1997.

M. Wooldridge. *An Introduction to MultiAgent Systems*. John Wiley and Sons, Ltd, Febuary 2002.

M. Wooldridge and N. R. Jennings. Intelligent agents: Theory and practice. *Knowledge Engineering Review*, 10(2), June 1995.

X. Wuzhi, J. Offutt, and J. Luo. Testing web services by xml perturbation. In *16th IEEE International Symposium on Software Reliability Engineering, 2005. ISSRE 2005.*, pages 266–276, 2005.

J. Yang, M. P. Papazoglou, B. Orriens, and W. van den Heuvel. A rule based approach to the service composition life-cycle. In *Proceedings of the Fourth International Conference on Web Information Systems Engineering, WISE 2003.*, pages 295–298. IEEE Computer Society, December 2003.

H. Yu and S. Reiff-Marganiec. A method for automated web service selection. In *2008 IEEE Congress on Services-Part I*, pages 513–520. IEEE, 2008.

T. Yu, Y. Zhang, and K. Lin. Efficient algorithms for web services selection with end-to-end qos constraints. *ACM Transactions on the Web (TWEB)*, 1(1), May 2007.