UNIVERSITY OF SOUTHAMPTON

A Design Framework for identifying Optimum Services using Choreography and Model Transformations

By

Saad Ali Alahmari

A thesis submitted for the degree of Doctor of Philosophy

In the

Faculty of Physical and Applied Science

Electronics and Computer Science

August, 2012

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF PHYSICAL & APPLIED SCIENCES

ELECTRONICS AND COMPUTER SCIENCE

Doctor of Philosophy

A Design Framework for identifying Optimum Services using Choreography and Model Transformations

by Saad Ali Alahmari

Service Oriented Architecture (SOA) has become an effective approach for implementing loosely-coupled and flexible systems based on a set of services. However, despite the increasing popularity of the SOA approach, no comprehensive methodology is currently available to identify "optimum" services. Difficulties include the abstraction gap between the business process model and service interface design as well as service quality trade-offs that affect the identification of the "optimum" services. The selection of these "optimum" services implies that SOA implementation should be driven by the business model and should also consider the appropriate level of granularity. The objective of this thesis is to identify the optimum service interface designs by bridging the abstraction gap and balancing the trade-offs between service quality attributes.

This thesis proposes a framework using the choreography concept to bridge the abstraction gap between the business process model and service interface design together with service quality metrics to evaluate service quality attributes. The framework generates the service interface design automatically based on a chain of model transformations from a business process model through the use of the choreography concept (service choreography model). The framework also develops a service quality model to measure service granularity and service quality attributes of complexity, cohesion and coupling. These measurements aim to evaluate service interface designs and then select the optimum service interface design. Throughout this thesis, a pragmatic approach is used to validate the transformation models applying three application scenarios and evaluating consistency. The service quality model will be evaluated empirically using the generated service interface designs.

Despite several remaining challenges for service-oriented systems to identify "optimum" services, this thesis demonstrates that optimum services can be effectively identified using the new framework, as explained herein.

Contents

List of Figures

List of Tables

Declaration of Authorship

I, Saad Ali Alahmari, declare that the thesis entitled "A Design Framework for Identifying Optimum Services using Choreography and Model Transformations" 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 in a number of conference and journal papers (see Section 1.4 for a detailed list).

Signed:.....

Date:.....

Acknowledgements

Throughout the past four years, I was given a privileged opportunity to work with Ed Zaluska, David de Roure and Dave Millard. All of them provide invaluable and unlimited support and guidance.

I cannot thank Ed enough for the outstanding support you showed throughout the years. Your constant detailed comments, encouragements and personal guidance were very essential to present this thesis. David De Roure, thanks for the confidence you gave me from the first day I met you. Your remarkable discussions and suggestions have shaped my research and helped me to be a researcher. I appreciate your continues support that you offered me after joining the Oxford University. Dave Millard, although you joined the supervisory team on my fourth year, your criticisms and excellent advices influence my research and direct me to solve hard research problems.

I would also like to thank Prof. Peter Henderson for his comments and guidance at early stage of my research. I also highly acknowledge effective meetings and helpful discussions with Rob Phippen and Kim Clark (IBM Hursley Park, UK).

I would also like to thank my friends and colleagues in the WAIS group, for their great support I received during my PhD at University of Southampton.

Finally, I would like to thank all my family members for their support, patient enthusiastic encouragement not only throughout my PhD, but also throughout my life. I would like to thank my father and mother and pray for them, who had provided me with endless care and love. I would also thank my oldest brother, Abdulrhman, who would always encourage and support me.

Definitions and Abbreviations Used

ARIS	Architecture of Integrated Information Systems
ASOC	Average Service Operation Cohesion
ASOG	Average Service Operation Granularity
ASOM	Average Service Operation Complexity
ASOU	Average Service Operation Coupling
ATL	ATLAS Transformation Language
BPEL	Business Process Execution Language
BPEL4Chor	Business Process Execution Language for Choreography
BPM	Business Process Management
BPMN	Business Process Modeling Notation
BPMS	Business Process Management System
BPSS	Business Process Specification Schema
CBD	Component Based Design
CIM	Computation-Independent Model
CORBA	Common Object Request Broker Architecture
CRM	Customer Relationship Management
CRUD	Create-Retrieve-Update-Delete
CXML	Commerce eXtensible Markup Language
DCOM	Distributed Component Object Model
ebXML	Electronic Business Extensible Markup Language
EDOC	Enterprise Distributed Object Computing
EMF	Eclipse Modelling Framework
ERP	Enterprise Resource Planning
IDE	Integrated Development Environment
IEC	International Electrotechnical Commission
ISO	International Organization for Standardization
JNI	Java Native Interface
M2M	Model-to-Model
MDA	Model-driven Architecture
MDE	Model-driven Engineering
MEP	Message Exchange Pattern
MOF	Meta-Object Facility
OCL	Object Constraint Language
ODG	Operation Data Granularity
OFG	Operation Function Granularity
OMG	Object Management Group
OO	Object Oriented
OOD	Object-Oriented Design
OWL-S	Ontology Web Language for Services
PCD	Process Chain Diagram

PIM	Platform Independent Model
PSM	Platform Specific Model
QoS	Quality of Service
QVT	Query/Views/Transformation
REST	Representational State Transfer
RUP	Rational Unified Process
SDE	Software Development Environment
SDLC	System Development Life Cycle
SLA	Service Level Agreement
SOA	Service-Oriented Computing
SOC	Service Operation Cohesion
SOG	Service Operation Granularity
UDDI	Universal Description, Discovery and Integration
UML	Unified Modelling Language
URI	Universal Resource Identifier
USM	Unified Service Model
W3C	World Wide Web Consortium
WS	Web service
WSBPEL	Web Services Business Process Execution Language
WS-CDL	Web Services Choreography Description Language
WSDL	Web Services Description Language
XMI	XML Metadata Interchange
XPDL	XML Process Definition Language
XSLT	XML Stylesheet Language Transformation

Chapter 1 Introduction

Service-Oriented Computing (SOC) is a cross-disciplinary paradigm for principles, technologies, and methods, and is based on software services that use its core architectural style, Service-Oriented Architecture (SOA). Organisations have increasingly shifted software development to SOA-based systems in order to improve interoperability, flexibility, and reusability. A software service presents a coherent set of functionality that is exposed via a standardised interface. The implementation of a software service is separated into the service implementation aspects and its interface. However, a key part of developing service-based systems is to break required functionalities down into a set of services, and a key challenge is to find an appropriate breakdown method to identify the "optimum" services. Because the business modelling and service interface designing are disconnected, developed services do not always meet the user requirements and specifications that satisfy software quality attributes. Moreover, its design and implementation suffer from not taking appropriate service granularity into account which results in low aspects of service quality.

In this thesis, "optimum" services refer to identified services that consider three challenges: the purpose of the service, the level of service granularity and the balance between trade-offs of the service quality attributes.

Firstly, the purpose of the service refers to the functionalities offered by the service in terms of service types, e.g., a service that provides Create, Read, Update and Delete functions (CRUD) is different from that one that provides infrastructure functions. The definitions of these functionalities can be derived from business processes in a process-oriented system. The service identification process is an initial step in service modelling for transforming business processes/requirements to candidate services. With the business processes and services residing on different architectural layers, the abstraction gap is the first challenge. We refer to the abstraction gap as the separation between the definitions of business models and the descriptions of service interface designs.

The existing methodologies of such authors as Kohlborn and Arsanjani have failed to bridge this gap (Arsanjani, Ghosh et al. 2008; Kohlborn, Korthaus et al. 2009). As a result, these contributions can be viewed as conceptual frameworks and general guidance. This thesis uses the choreography concept to bridge the semantic gap between the business process model and service interface design. The choreography concept appears at business process level and the service composition. At the business process level, the choreography concept describes an observable behaviour of a participant (e.g., a company) or participant's role (e.g., a buyer or seller) in an interaction. In service composition, the service choreography refers to a peer-to-peer description of the global observable interactions between aggregated services. As a result, bridging the abstraction gap should enable the automatic generation of service interface designs according to corresponding defined business process models.

Secondly, Service designers do not agree on when services should be coarse-grained or fine-grained. A recent study by industry experts that evaluated different SOA development processes concluded that service granularity is a key issue in the design phase (Haines and Rothenberger 2010), to a certain extent there is some agreement on the importance of the granularity concept for service-based systems (Haesen, Snoeck et al. 2008; Rosen, Lublinsky et al. 2008; Haines and Rothenberger 2010; Sweeney 2010). It is difficult to specify heuristic rules for defining the appropriate level of granularity that can be applied in all circumstances. But, the quantification of service granularity using the proposed service quality model can assist selecting the appropriate level of granularity for a given service interfaces. This quantification allows the service designer to evaluate the service granularity for the service interface design in accordance with the service quality attributes of complexity, cohesion and coupling.

Finally, balancing the trade-offs between the service quality attributes that affect identifying the "optimum" services is essential. The level of the service granularity influences the service quality attributes. For example, implementing a system with a number of fine-grained services can result in a negative effect such as poor performance because of increasing communication trips but offer good reusability (as smaller services are more loosely-coupled). Thus, we define a service quality model that defines the properties required to measure service granularity and the service quality attributes of complexity,

cohesion and coupling; employing the software quality attributes for SOA can assist in the achievement of "optimum" services.

This thesis explores the problem of identifying "optimum" service interface design for process-oriented systems, and answers the following questions:

- Is it possible to generate service interface designs automatically from business process models using the choreography concept?
- What is the impact of a high level of service granularity on the quality attributes of complexity, cohesion and coupling compared to a service interface design with a low level of service granularity?
- What are the relationships between each of the service quality attributes of complexity, cohesion and coupling?

In section 1.1, the research hypotheses and questions are discussed. In section 1.2, an outline of the thesis contributions is given. The thesis structure and publications are explained in sections 1.3 and 1.4 respectively.

1.1 Research Hypothesis

There is a need to develop a complete methodology for identifying optimum services. Given the above challenges: the abstraction gap, the service granularity and balancing service quality trade-offs, the hypotheses of this thesis as follows:

H1*: "It is possible to use service choreographies (WS-CDL) to derive the* ***automatic transformation*** *of a business process choreography model (BPMN 2.0) into a service interface design (WSDL)".*

- **Automatic transformations.** The transformation process should be automated fully from the business process model to the service interface designs. That is, no manual human intervention should be required to determine the semantic elements that should be defined for a service interface. This is because manual intervention decreases the robustness of the service identification process and affects the level of detail, depending on the human's understanding of system requirements. In particular, it increases the abstraction gap between the descriptions of the business process model and the corresponding service interface design. With respect

to current intensive research and practice in service modelling methodologies in various domains, a significant shift from human-based decisions and manual architectural activities to a higher degree of automation is needed.

- **Standardised mapping.** The semantic mapping between different models (e.g., business process models and service choreography models) needs to be based on standard specifications and firm theoretical grounds. This is particularly important for defining the meta-models for source and target models and developing a theory to bridge the abstraction gaps. The framework herein is based on a coherent series of transformed models that achieve ultimately SOA benefits in heterogeneous development environments.
- **Improve flexibility and accuracy.** Implementing the transformation should be flexible enough to generate a variety of service interface designs to enable service designers to evaluate the impact of trade-offs on various designs and selecting the optimum design. These service interfaces are supported with benchmarks for service quality attributes to provide accurate measurements. The time needed to generate the various service interface designs automatically is more efficient compared to the manual human process.

H2: "*A set of services with a high value of service granularity would correspond with a positive effect on the quality attributes of complexity and cohesion and a negative effect on the quality attribute of coupling compared to services with a low value of service granularity*".

- **The relationships between quality attributes**. The relationships between service granularity and service quality attributes of complexity, cohesion and coupling need to be analysed. The statistical method of linear/nonlinear regression can be used to analyse the effect of service granularity as an independent variable on service quality attributes as dependent variables. Prior to the analysis, a quality model that quantifies service granularity and the service quality attributes of complexity, cohesion and coupling need to be developed.
- **Valid quality metrics**. The quality model should provide theoretically and empirically valid metrics. The theoretical validations can be based on standard property definitions; empirical validations can use the dataset

generated from the service interface designs. The quality model should provide key features of defined metrics and show how these metrics are driven. This is important when the cause-effect relationship between these attributes is investigated. The service quality model should assist with the achievement of optimum service interface designs by providing numerical results.

H3: "*The following architectural quality attributes are dependent on one another, cohesion correlated with coupling, coupling correlated with complexity and complexity correlated with cohesion*".

- **The correlated relationships**. The results of correlation investigation will be useful to understand the significant effects of these quality attributes on each other which might provide an insight to the selection of optimum service interface designs. The correlation relationships between service quality attributes can be investigated statistically using the correlation test. The correlation coefficient value can be interpreted into different scale values. All data computations and extractions can be completed using the proposed service quality model.

1.2 Research Contributions

As a summary, the main conceptual contributions of our research work are:

- A method to generate a service interface design (WSDL) automatically from the business process model (BPMN 2.0) using service choreography (WS-CDL) thus enabling the choreography concept to bridge the abstraction gap between a business model and service interface designs. This method also supports seamless integration between SOA and MDA and offers an application for such integration. (Explained in chapter Chapter 4).
- A service quality model was developed to provide metrics for measuring the service granularity and SOA quality design attributes of complexity, cohesion and coupling. The service quality model was also used to select the optimum service interface design for a set of services. We developed theories of these metrics based on our understanding and knowledge together with existing literature on the topic of software quality

measurement. We provided a measurement for service granularity that can be enhanced to include additional factors. (Explained in chapter Chapter 5).

- We offered an extension of the semantics of BPMN 2.0 specifications to generate service choreographies (WS-CDL) and to facilitate measurements of service quality attributes. (Explained in chapter 4.3).

The practical contributions of this thesis are as follows:

- Implementation of a chain of transformation programs in ATL from the business process model (BPMN 2.0) to service choreography (WS-CDL) and then from service choreography (WS-CDL) to service interface design (WSDL). (Explained in chapter 6.4).
- Implementation of a Java-based application for the analysis and computation of a set of metrics for service granularity and the service quality attributes of complexity, cohesion and coupling. (Explained in chapter 6.5).

A further contribution of this thesis is as follows:

- The service granularity metrics (OFG, ODG, SOA and ASOG are described in section 5.2) that are proposed in this thesis are recognized and adapted by Prof. Cássio Prazeres at Department of Computer Science (DCC) at Federal University of Bahia, Brazil. The metrics will be implemented in a project to develop a test platform for evaluating service compositions. The implementation will be published at the 14th International Conference on Information Integration and Web-based Applications & Services (iiWAS2012).

1.3 Thesis structure

The remainder of this thesis is structured as follows:

- Chapter 2 presents the field disciplines of SOA and MDA that are relevant to this thesis. The service development cycle for SOA is described, focussing on service modelling. We provide an overview of the service definitions and elements used in this thesis and phases of service identification showing the currently used strategies for identifying

services. We explain the concept of service granularity and how quality attributes fit in SOA. We then introduce the basic concept of the modelling language BPMN 2.0 and the choreography language WS-CDL. After that, a general review of the model-driven approaches is provided along with supported technology standards, model transformations and languages (in particular, ATL is reviewed).

- Chapter 3 discusses current research in the area of service identification by classifying current research efforts into three views: business-driven, ontology-driven and legacy system-driven. We also investigate current approaches concerning service quality attributes and metrics. A comparative analysis is conducted based on a number of criteria for current research efforts, as explained at the beginning of this Chapter. Important challenges of service identification are bridging the abstraction gap between business models and service implementation, and measuring quality attributes. These two challenges are discussed in Chapters 4 and 5 which form the initial framework design.
- Chapter 4 presents choreography concepts, which are important for bridging the abstraction gap and transforming models. We discuss why and how the choreography concepts are applicable for use in the research hypotheses. An analogy is developed between business process choreography and service choreographies. To realise the model-driven approach, we defined the meta-models that are required for the model transformations using BPMN 2.0, WS-CDL and WSDL. We attempt to adapt available meta-models in literature and standard specifications, rather than define meta-models from scratch. We also evaluate the suitability of the choreography specifications in BPMN 2.0 and WS-CDL against a number of choreography requirements.
- Chapter 5 introduces a service quality model that is developed based on the service granularity definitions. We present the basic definitions for service granularity metrics. After investigating the impact of service operation granularity on architectural quality attributes, metrics for the service quality attributes of complexity, cohesion and coupling are defined. These metrics are validated theoretically using a number of mathematical property definitions.
- Based on the choreography concept (chapter 4) and the service quality models (chapter 5), the framework architecture and detailed

implementations are described in chapter 6. The framework architecture is presented in two architectural parts: model transformations using choreography and a service quality model. The implementation of each part is individually explained in detail. First, the technical implementations and transformation rules are explained for each model transformation. Second, the implementation of the service quality model is described.

- To evaluate and analyse our framework, we decided to conduct two types of evaluation using different scales: pragmatic and empirical. In Chapter 7, we validate the consistency of the modelling behaviour between inputs and outputs during the transformations. Three application scenarios are used to demonstrate the pragmatic approach.
- In chapter 8, after the computation of quality metrics using the service quality model, we empirically evaluate the generated service interface designs. The analysis and findings of the research results are discussed.
- Chapter 9 concludes the thesis with a review of its contributions to the field of "service modelling" and a presentation of extensions for future work.

1.4 Publications

During the research, the following peer-reviewed papers have been published:

- S. Alahmari, Ed. Zaluska, D. De Roure (2011). A Metrics Framework for Evaluating SOA Service Granularity. In, The 8th IEEE 2011 International Conference on Services Computing (SCC 2011), Washington, D.C, USA, 04 - 09 Jul 2011. IEEE Computer Society Press.
- S. Alahmari, D. De Roure, Ed. Zaluska (2010). A Model-Driven Architecture Approach to the Efficient Identification of Services on Service-oriented Enterprise Architecture. At The Second Workshop on Service oriented Enterprise Architecture for Enterprise Engineering in conjunction with the 14th IEEE International Enterprise Distributed Object Computing Conference, Vitória, Brazil. IEEE Computer Society Press.

- S. Alahmari, Ed. Zaluska, D. De Roure (2010). Migrating Legacy Systems to a Service-Oriented Architecture with Optimal Granularity. ICEIS 2010 - Proceedings of the 12th International Conference on Enterprise Information Systems, Volume 1, DISI, Funchal, Madeira, Portugal, June 8 - 12, 2010.
- S. Alahmari, Ed. Zaluska, D. De Roure (2010). A Service Identification Framework for Legacy System Migration into SOA. In, IEEE SCC 2010 -7th International Conference on Services Computing, Miami, Florida, USA, 05 - 10 Jul 2010. IEEE Computer Society Press.

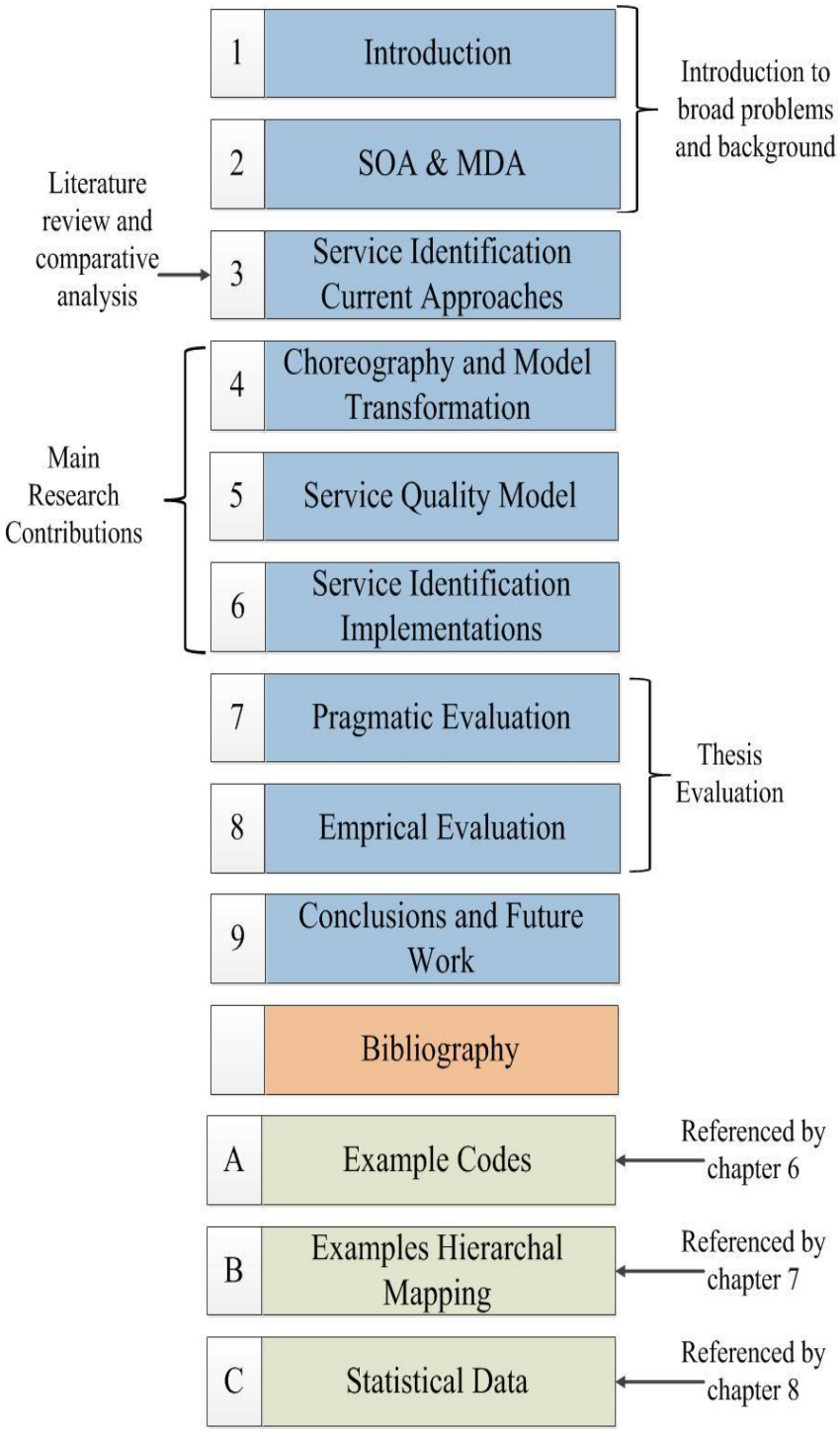


Figure 1-1 Thesis structure

Chapter 2 SOA and MDA

The need for a complete methodology to identify optimum services in the context of business process has been intensively discussed (Zdun and Dustdar 2007). In Chapter 1, we described the challenges that face the service identification process. In this chapter we provide an overview of the field, and the technologies that are particularly important and relevant for understanding the context of the thesis. The overview is important to define the two major fields of SOA and MDA. These technologies are used in current approaches for the service identification process which will be explained in Chapter 3.

This chapter is structured as follows: Section 2.1 introduces the field of Service modelling in the service development cycle. In particular, an overview is given which explores the service identification process within the service modelling and the definitions of a term "service". Section 2.2 discuses service oriented decomposition as one of the modelling strategies used to identify services in enterprise architecture. Service designers do not know the size of functionalities a service should offer nor when a service can be called "optimum". The size of a service is presented through the discussion of the service design issues related to granularity with considerations of the importance of having an appropriate level of granularity, where employing the software quality attributes for SOA can assist to achieve the "optimum" services. During the service identification process, they may be composable and described by the choreography languages from a global viewpoint.

Section 2.3 describes business process modelling with a focus on Business Process Model Notation (BPMN) representations. Section 2.4 explores the field of Model Driven Architecture (MDA) and the concept of model transformation,

using relevant technologies and methods. These show techniques and technologies of MDA can be used to identify potentially "optimum" services.

2.1 Service Modelling

Service-Oriented Architecture (SOA) is a modern approach to implementing (re-implementing) a system as a set of interoperable services. Service-oriented analysis, design, and architectural disciplines all contribute to the service modelling approach (Bell 2008). Within the development life cycle, the term "modelling" denotes what was previously referred to as "analysis and design" in previous design methodologies (Bieberstein, Bose et al. 2005). These methodologies of modelling service-oriented systems are built on theoretical foundations, adopting a variety of effective approaches such as Model Driven Architecture (MDA). Service modelling considers the process of service delivery within an interoperable environment, beginning with a model representing real business requirements and includes the construction of a code skeleton to assist the implementation of these requirements. The software is required to conform to key design characteristics such as flexibility and reusability because these characteristics are important to decide whether the service design is appropriate. These might be fundamental non-functional requirements for the system.

The notion of modelling has received significant attention within SOA. A reference model has been proposed to formalize the underlying aspects of SOA (Haesen, Snoeck et al. 2008). This proposal is intended to cover the significant entities and properties of SOA, as well as their relationships, although the proposed model is limited in its description of advanced service interaction scenarios and therefore not comprehensive. In industry, development activities that relate to the design phase are almost invariably different from one organization to another because of the absence of development standards (Haines and Rothenberger 2010). With more general views comparing to the "reference model", Dijkman and Dumas propose a core model for service-oriented design, based on multi-viewpoints of choreography, orchestration and provider behaviour, as well as interface behaviour with specific characteristics such as high autonomy, and low level of granularity (Dijkman and Dumas 2004). In fact, currently there are neither clear characteristics nor a formal approach that might guide modelling services.

2.1.1 Service Oriented Architecture (SOA)

Accommodating technological evolution and rapid business changes is a significant problem with current software systems. Current software systems were typically developed with embedded business rules and logic, scattered and duplicated code, unstructured modules and tightly-coupled functions. Furthermore, external changes in business and application requirements are introduced (e.g., the recent emphasis on governance), emphasising the requirement for a modern architectural style such as SOA. A design methodology based on SOA provides a standardized way to improve both efficiency and flexibility because SOA enables transformation of the logic and views of business applications to a number of reusable services (Sweeney 2010). It provides a mechanism to incorporate the business strategies, implementation methodologies and operational aspects of the service-oriented system. SOA is not a new concept, having evolved from previous module-based development methodologies such as modular programming, software component and O-O design (Endrei, Ang et al. 2004). In fact, the term "SOA" has traditionally been defined from a number of different perspectives, for example; its functional aspects as being layered-enterprise based (Rosen, Lublinsky et al. 2008), usefulness in achieving business and solution strategies (Rosen, Lublinsky et al. 2008), and from a technical or business aspect (Bieberstein, Bose et al. 2005). This breadth illustrates that SOA can be presented and discussed from various different viewpoints. With this in mind, the level of abstraction provides an effective technique to study software architecture(Bieberstein, Bose et al. 2005). Figure 2-1 shows SOA layers of abstraction, as typically presented in the literature (Erradi, Anand et al. 2006; Rosen, Lublinsky et al. 2008) which partitions the architecture into six specific layers as follows:

- *Presentation layer:* this layer provides users with specific applications or alternatively a mechanism for interaction with business processes.
- *Business process layer:* this layer represents workflows (business processes) which are uniquely defined as sequences of activities responding to a business function or functions. A business process is often implemented as a service or a composite of services, and executed as part of a Business Process Management System (BPMS).
- *Business services layer:* this layer provides a number of services that respond to the business process layer, presenting coherent business functionalities.

Typical services are coarse-grained, though every service can be implemented with a number of fine-grained services. A Service Level Agreement (SLA) can be specified to govern and manage the quality of service provided to the service consumer.

- *Infrastructure service layer:* this layer provides a number of services supporting shared functions (e.g., to implement authorization or perform performance tuning) and also can support other enterprise services such as data services and integration services.
- *Service Component layer:* this layer typically comprises a block of services designed specifically to meet a potential future requirement (e.g., future re-use or an anticipated new requirement).
- *Operational and resources layer*: this layer usually represents existing applications (i.e., legacy systems and custom applications). These applications provide operational functionalities for underlying service components (e.g., existing systems Enterprise Resource Planning (ERP) and Customer Relationship Management (CRM) or custom application J2EE).

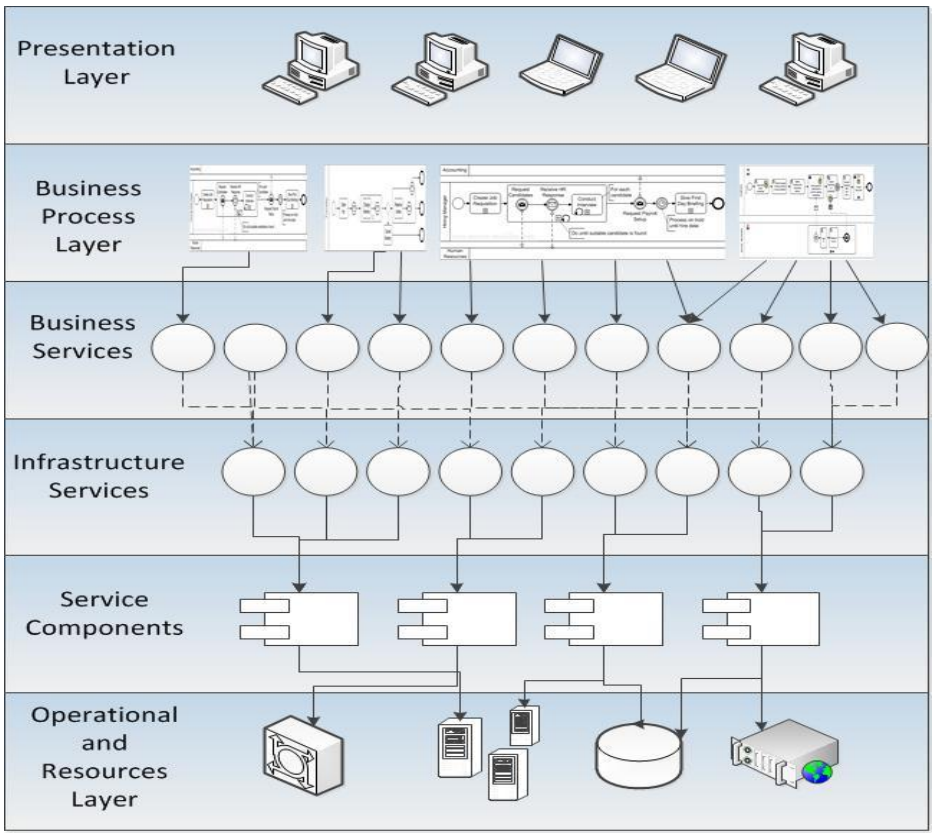


Figure 2-1 Typical SOA Layers of Abstraction

Each level of abstraction can however be further divided into finer detail. In addition, business and infrastructure service layers in particular will often be consolidated which narrows the abstraction gaps between the SOA layers and facilitates the service identification process.

The service development life cycle is an incremental process with multiple phases (Papazoglou and Van 2006). Different research methodologies propose different service development cycles. A "traditional" service development cycle consists of six phases: planning, service analysis, service design, service development, service testing, service deployment and service administration (Erl 2005), figure 2-2. The planning phase creates business and IT strategies that assist in achieving the benefits of an SOA implementation, studying the feasibility of the proposed system. It can also ease the transformation from traditional architectural and development practices towards a robust, flexible development environment within a service-oriented approach. The service analysis phase gathers business and software requirements, defines constraints, and identifies candidate business services using a specific modelling strategy. For example, a policy to re-use valuable existing components using a re-engineering method (e.g., the use of web-service wrappers).

The service design phase defines the specifications and features of services within the service boundaries in order to allow tracing of service specifications between requirements. The service development phase transforms service elements into executable software which operates using appropriate technologies. The service-testing phase comprises verification and validation of service code using rigorous testing techniques and is intended to ensure that the service implementation satisfies the functions and proprieties defined at the design stage. The service deployment phase carries out the configuration and advertising of services in a repository enterprise, i.e., installing and integrating middleware software. The service administration phase manages service issues such as monitoring, versioning, and maintenance, through, for example, defining ways to enhance and monitor performance.

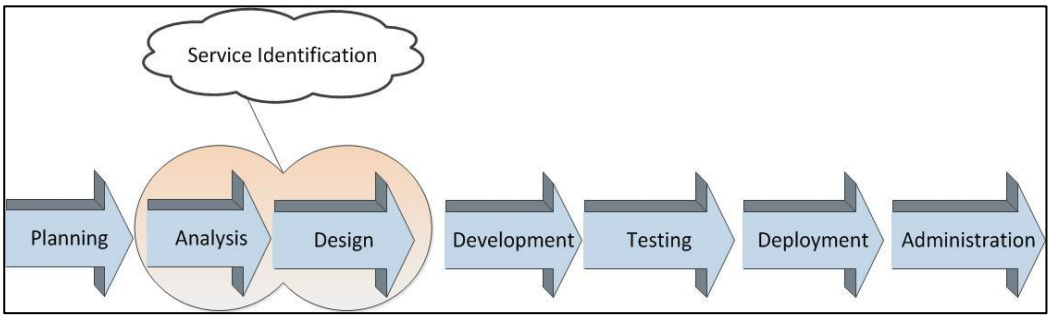


Figure 2-2 Service Development Life-cycle

The disciplines of analysis and design are embodied in the service-oriented modelling paradigm (Bell 2008). The service identification process aims to transform a description of a service (in either text or model form), and will move typically from business application requirements at the planning phase to more detailed formal specifications with a mapping technique (such as top-down mapping). The final result of the process is a skeleton containing a full specification of the service elements in the design phase. This transformation is an iterative process, based on the state of the service during the life cycle, and correspondingly it leads to a service-modelling discipline (Bell 2008). Essential challenges addressed are the ways in which services are identified, described, and realized to deliver maximum flexibility, agility and reusability (Arsanjani 2004; Erradi, Anand et al. 2006; Dwivedi and Kulkarni 2008; Bell 2010). Service identification is one of the most important tasks in defining the optimum set of services, as any ill-advised modelling decision can result in compromises that will affect the entire service-oriented enterprise. We would argue that the "optimum" services are those that correspond to the requirements of business applications and consider trade-offs between service quality attributes according to the system/user requirements.

2.1.2 The Definition of Service

The service is the core element of any SOA implementation. The term service is used generally across a wide spectrum of different computer science areas, with many different specific meanings. Study of the literature in service design and modelling will reveal a number of different service definitions, based on (for example) the analysis techniques used in modelling, the potential benefits of adopting SOA, and an understanding of the guiding principles of SOA. From a business perspective, a service can be defined as a discrete unit of business functionality (Rosen, Lublinsky et al. 2008). Technically a service can be

defined as a software resource exposed and discovered via an interface, with policies to facilitate different configurations (Arsanjani, Ghosh et al. 2008). A number of other service definitions can be found in (Wiersma 2010).

For the purposes of this thesis, it is essential to have a clear understanding of the term "service" (Bell 2010). We have adopted the W3C definition for a service, "*an abstract resource that represents a capability of performing tasks that represents a coherent functionality from the point of view of provider entities and requester entities*" (W3C 2004). The capability offered depends on the level of abstraction and the type of service. For example, data services residing in the data layer will typically support data access and manipulation. Unlike other service definitions, the W3C definition emphasises that (functionally speaking) the service always offers benefits as a resource in a self-contained representation between a service provider and a service recipient. It is worth noting that with this definition, the W3C attempts also to link service definition with a web service (WS) definition (service implementation) by means of the term "*resource*". According to W3C (W3C 2004), the service also embodies the properties of the definition of the term "*resource*" such as *an identifier* in service definition. Although the W3C definition of a service (Funk, Kuhmunch et al. 2005) is general, it also addresses the key characteristics necessary to call a software unit a service.

The design of a service can be defined conceptually according to three elements: the contract, the interface and the implementation (as shown in Figure 2-3). The service contract provides informal specifications of the purpose, message types, functionality, constraints, and usage of services which are published as documents. The service interface exposes the service functionalities to the representation layer through a set of operations. The design of an interface is isolated from the design of the software system in most modern software approaches (Berners-Lee 2003), with the service implementation encapsulating both business logic and related data.

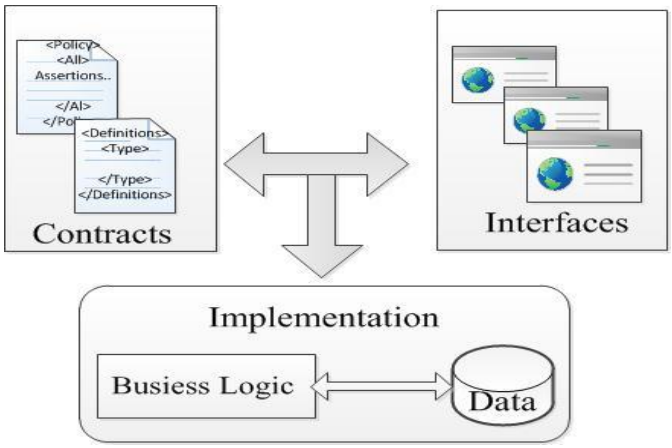


Figure 2-3 Service Elements

2.2 Service-Oriented Architecture Decomposition

Service-oriented decomposition is one of the modelling strategies used to identify services in enterprise architecture, describing the way in which a business-domain model is partitioned into services. In the literature, the term "composition" is often used in conjunction with the word "service" to refer to a combination of services to provide new functionality (Rosen, Lublinsky et al. 2008). As software complexity has increased, the technique of decomposition has become more important and is intended to separate entire applications into a number of separate programs (Rosen, Lublinsky et al. 2008) .

In the context of SOA, decomposition is the breaking down of hierarchical business domains into business processes or functions using a top-down analysis technique. A considerable number of existing methodologies are available to define services based on decomposition of business processes models (Zhang and Yang 2004; Zhang, Liu et al. 2005; Jamshidi, Sharifi et al. 2008) (these methodologies are explained in section 3.1). Each business process is decomposed into activities (a set of tasks) which can be realized as either a candidate service (or a set of services), and consideration of the appropriate level of service granularity by the service identification process is the main task of the service-oriented decomposition process (Erradi, Kulkarni et al. 2009). The underlying technique of service identification affects both the service features and also the level of granularity. The key issue here is that it is important to find a methodology to identify the optimum services. The methodology should consider service quality attributes and the design issue of service granularity.

2.2.1 Service Identification

As explained above, service identification is the key issue when identifying business services in service-oriented systems (Endrei, Ang et al. 2004 ; Rosen, Lublinsky et al. 2008). The service identification process is part of both the analysis and design phases of the SOA development cycle and denotes the process of generating definitions of an appropriate set of services in a service-oriented project. Indeed, the service identification process is based on analysis techniques that depend on the available resources and project constraints, e.g., migrating legacy code by simply wrapping the code as one or more web services because budget constrains prevent a more comprehensive review.

Although there are a number of approaches for service identification in SOA, identifying the optimum services for a service-oriented system remains a significant challenge. A number of possible approaches have been delivered from a variety of different perspectives, including business process driven, tool-based MDA, wrap legacy, developing legacy code as components, data-driven, and message-driven approaches (Arsanjani 2005). Further classification of SOA developmental approaches is possible, based on the delivery strategy, lifecycle coverage, degree of prescription, target availability, process agility, and planned retention of existing processes, techniques and notation (Ramollari, Dranidis et al. 2007). However, the SOA paradigm has the potential to address distinctive features and requirements, which requires a comprehensive methodology in order to provide sufficient guidance for every phase of the service development cycle. (A full review of the literature will be provided in chapter 3).

The service identification phase is crucial because mistakes made at this stage can lead to overall failure of the resulting SOA-based systems. The set of services defined at this stage needs to be of an appropriate size for the required system and we believe that service granularity is one of the key architectural issues affecting service identification process to achieve the optimum service interface design. In fact, SOA has inherited important architectural considerations (such as software size (Costagliola, Ferrucci et al. 2005)) from former architectural approaches (e.g., O-O (Booch, Maksimchuk et al. 2007), CORBA (Mowbray and Malveau 1997)). Success is critically dependent on the correct identification, presentation and definition of key services at the “right” level of granularity since the exposed functionalities in a service define its granularity. It is important to appreciate that achieving an appropriate level of

service granularity inevitably requires a compromise between many elements, both technical and non-technical. In particular, the optimal granularity of key services can be expected to vary in different layers with different service types (Kohlmann and Alt 2007) and layers (Reldin and Sundling 2007; Kulkarni and Dwivedi 2008).

Despite these requirements, there is an increasing acceptance of the SOA based design approach for developing large-scale systems, despite there being no standardised methodology. The typical strategies for SOA development are referred to as "top-down", "bottom-up" and "meet-in-the-middle" (Perepletchikov, Ryan et al. 2005). In this thesis, we will focus on these strategies, because most of the available published work has used these terms:

Top-down strategy: This strategy identifies business services from a business perspective, by (for example) mapping products, business processes or use cases onto a set of business services (Galster and Bucherer 2008), and decomposing business domains into functional areas and components (Perepletchikov, Ryan et al. 2005). SOA can be specifically differentiated from other software methodologies because it is explicitly intended to be strategically aligned with the underlying business vision (Arsanjani and Allam 2006). It is particularly relevant in business models which must respond to business transactions using a set of sequenced activities or tasks. This strategy makes use of domain analysis, which itself requires use of specific analysis methods. Chen et al. suggest a feature analysis method that can be used to identify, model, locate, and then aggregate system features, and also assist in the conceptual classification of legacy system granularity (Chen, Li et al. 2005). Zhang and Yang (Zhang and Yang 2004) apply clustering analysis methods together with human supervision to specify acceptable levels of granularity and service loose coupling for the migrated code (Fraley and Raftery 1998). Although the top-down strategy defines service with improved quality attributes, in practice some migration of existing infrastructures is always required.

Bottom-up strategy: This strategy deliberately works 'backward' from the technical basis to the system requirements based on existing technologies, i.e., legacy-system components are grouped into services on the basis of existing system functionalities. This strategy particularly advocates the migration of legacy systems into services (Krafzig, Banke et al. 2005). It requires an analysis of the business requirements in order to define service functionalities, and

integrates appropriate functions of the legacy systems into independent components based on the validity of the business logic. Adaptors can then be created which shield the legacy systems from the web service interface; this strategy is sometimes referred to as the "black-box" approach (Sneed 2001). It might also develop web services to implement the key business logic of the existing code (Zou and Kontogiannis 2001). Jianzhi, Zhuopeng et al use a reverse engineering technique on a component-based approach using a Java Native Interface (JNI) wrapper to encapsulate code, and the Commerce eXtensible Markup Language (CXML) to describe specifications for communication within a workflow (Jianzhi, Zhuopeng et al. 2005).

Meet-in-the-middle: This strategy combines both the bottom-up and top-down approaches, with an emphasis on migrating valuable components from the legacy system. Software designers start by deciding what existing software assets should be migrated and the best way to migrate them without losing significant system functionalities. It is an iterative process; along with integration of available software assets (by defining web service wrappers for legacy functionality), high-level business activities are decomposed into business services. Defined services from both approaches are validated iteratively against the software requirements. Erradi et al. (Erradi, Anand et al. 2006) advocate a hybrid approach, incorporating a top-down approach for domain decomposition and a bottom-up approach for application portfolio analysis, using a variety of manual techniques (e.g., interviews and questionnaires) together with automation tools. Other design and development approaches are also available (such as Middle-Out, and Front-to-Back (Shirazi, Fareghzadeh et al. 2009; Bell 2010), but these alternatives are less well accepted than the strategies discussed above. Middle-out models services based on defined goals as goal-service modelling, Front-to-Back tracks calls for the user interface and presentation layer logic.

In summary, there is no comprehensive strategy that guides the analysis and design phases of service identification for a complex system. Furthermore, ambiguity in the definition of major enterprise business processes is a common issue with all of these strategies (top-down, bottom-up and meet-in-the-middle) when applied to the development of business scenarios (Papazoglou and Van 2006). Nonetheless, a number of approaches assert that the meet-in-the-middle

approach combines the advantages of the other strategies (Erradi, Anand et al. 2006; Arsanjani, Ghosh et al. 2008; Kohlborn, Korthaus et al. 2009).

2.2.2 Service Granularity

The term granularity is defined as "the scale or level of details in a set of data", according to the Oxford dictionary[1]. Granularity reflects the degree of system complexity in software design, and is thus a key design factor in defining software units for software development methods, irrespective of whether the software unit is a module, object, component, or service. Indeed, this increasing level of modularity and abstraction is designed to solve issues related to granularity (Brereton and Budgen 2000), e.g., objects in object-oriented programming were intended to represent real-world concepts. In the context of SOA, *service granularity* refers to the complexity of the functionality offered by a service . Granularity refers to the functional capabilities offered by a service, or the number of business transactions/processes implemented by a service. Coarser-grained services include large numbers of operations and exchange larger amounts of data.

To a certain extent there is some agreement on the importance of the granularity concept for service-based systems (Kohlborn, Korthaus et al. 2009). A recent study by industry experts which evaluated SOA development processes concluded that service granularity is one of the key issues in the design phase (Haines and Rothenberger 2010). Nonetheless, the definition of this property is still not fully agreed, due to the subjectivity of the relative aspects and a lack of any theoretical grounding (Haesen, Snoeck et al. 2008). Architectural layering of services in the SOA is used to classify services and then define levels of granularity based on different service types (figure 2-1). Dwivedi and Kulkarni define in broad terms eight hierarchical service types: process service, business service, composite service, informational service, data service, utility service, infrastructure service, and partner service (Dwivedi and Kulkarni 2008) (more classifications can be found in (Erl 2005; Papazoglou and Van 2006)). Service granularity is evaluated based on the type and definition of every service. For example, a business service is coarse grained compared to an infrastructure service due to a higher level of abstraction, and vice versa.

[1] (2011) Granularity: Compact Oxford English Dictionary Online http://oxforddictionaries.com/.

The concept of granularity applies to different levels of abstraction, i.e., the functionality offered by an operation in a particular service interface is different from the functionality that is offered by a service implementation. It is important that we differentiate between different types of granularity in order to analyse the relative quality attributes. Erl et al. propose four types of service granularity (Erl, Karmarkar et al. 2008). The first is *service granularity*, which indicates the functional scope of the overall service context. The second is *capability granularity*, which focuses on the functional scope at an individual service level. The third is *constraint granularity*, which aims to quantify the level of validation logic detailed. Finally, *data granularity* refers to the size of the exposed data. In a more structural classification, Haesen et al (Haesen, Snoeck et al. 2008) classify three types of service granularity: *functionality granularity*, which refers to the size of functionalities offered by a service, *data granularity*, which is the size of data exchange within a service, and *business value granularity,* which refers to the business value added by a service. These service types and levels of abstraction are also used together to assist with the definition of the various types of service granularity. A number of resources have discussed granularity from the perspective of development strategies, including top-down and bottom-up, focusing on the impact of development strategies on the correct definition (Perepletchikov, Ryan et al. 2005; Boerner and Goeken 2009; Ma, Zhou et al. 2009). According to these classifications, functionality, data, and level of abstractions are the most important elements in the classification of granularity. Further analysis of these elements would assist in providing better decisions regarding the service design.

The underlying service identification process in SOA specifically depends on defining the right services with a proper level of granularity. A considerable amount of literature has proposed methodologies for identification of the right services with appropriate granularity (Papazoglou and Van 2006; Dwivedi and Kulkarni 2008; Kim, Kim et al. 2008; Kulkarni and Dwivedi 2008; Zhang, Zhou et al. 2008) (these references will be explained later in chapter 3, section 3.1). Although these approaches have used a variety of different techniques, they have not agreed on how to define the correct level of granularity effectively, agreeing instead on the difficulty of delivering a set of services with appropriate granularity. Furthermore, when designing the services, the impact of granularity on quality of service (QoS) aspects must also be considered. Identification of services with an appropriate level of granularity has the

potential to provide other potential benefits of SOA such as flexibility, reusability, and functionality.

2.2.3 Service Quality Models

Evaluation and enhancement of software quality is a key objective of software engineering. The definition of the important software qualities are always different from one stakeholder to another, e.g., do we require a flexible set of services with high reusability standards or alternatively low complexity service components with high agility? In literature, a number of quality models have been suggested to evaluate various quality attributes within different applications. The concept of such models was established by McCall for quality investigation in development processes (McCall, Richards et al. 1977), with additional models (such as the models published by Boehm and Deutsch (Boehm 1976; Deutsch and Willis 1988)) appearing later. A quality model defines characteristics and properties that need to be measured, enabling the use of software metrics to measure such. Software quality metrics (essentially a subset of software metrics with special focus on quality) have been classified into product metrics, process metrics and project metrics (Kan and Jones 2004). The first attempt to use metrics for software quality prediction was by Akiyama (Akiyama 1972) in a simple regression-based model (Fenton and Neil 1999).

SOA is an approach, not a product (Rud, Schmietendorf et al. 2006). It does not follow a specific development methodology process and furthermore SOA implementation can be achieved by a variety of different technologies, e.g., Representational State Transfer (REST), Web service (WS) and Distributed Component Object Models (DCOM). We believe that focusing on the implementation of services means that product metrics are more appropriate to SOA than project or process metrics. The features and properties of a product (service) represent software quality attributes (Perepletchikov, Ryan et al. 2005); typically classified as external and internal attributes (Costagliola, Ferrucci et al. 2005). The external attributes, called characteristics, relate to the product environment, for example, the ISO/IEC 9126-1:2001 standard defines external software quality attributes as usability, maintainability, efficiency, portability, functionality, and reliability (ISO/IEC 2001). The internal attributes are related to the product itself, for example, measuring the software size, coupling, cohesion, and complexity, and such an

attribute might impact one or more external attributes. At the enterprise level, quality-in-use can be used to measure specific needs in order to achieve specific goals effectively, productively, safely and satisfactorily in specific contexts of use, according to the ISO/IEC 25020 (ISO/IEC 2007). Fig. 2-4 shows the relationships between different quality attributes in the context of an enterprise system adopting SOA.

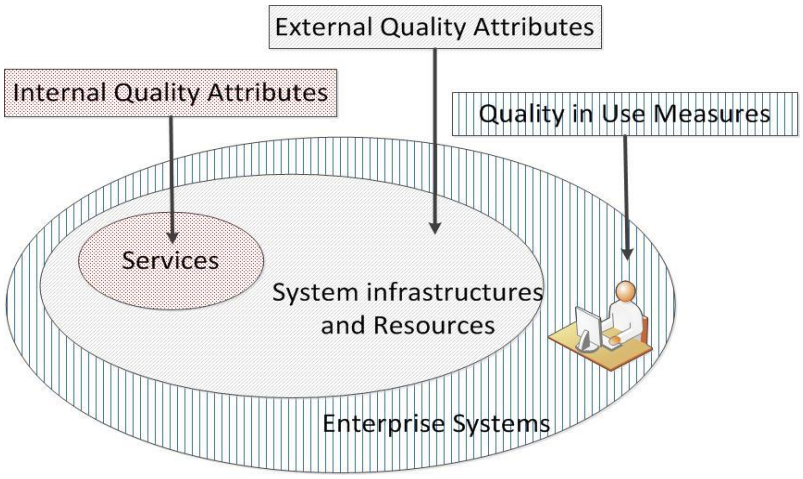


Figure 2-4 SOA Product Measurements

Currently, SOA is emerging as an innovative approach with considerable promise for improving common software quality concerns such as unacceptable inflexibility and complexity. Despite the extensive amount of research within the area of service quality (QoS), no agreed standards are currently available to evaluate the implementation quality of service-based systems. Indeed, the existing SOA quality models focus on broad measurements of external structural software service attributes (such as complexity, reusability and performance), neglecting the impact from internal structural software attributes, and in particular from service granularity.

2.2.4 Service Choreography WS-CDL

Web Services (WS) are currently a widely adopted implementation method for SOA (Barker, Walton et al. 2009). Web services can be composable and described by choreography languages from a global viewpoint. The choreography languages describe rules of collaborations between participants and help to ensure service interoperability between services. Despite the large number of existing choreography languages such as Web Services Choreography Description Language (WS-CDL), BPEL4Chor (Decker, Kopp et al. 2007),

Ontology Web Language for Services (OWL-S) (Martin, Burstein et al. 2004) and Let's Dance (Taylor, Shields et al. 2003), none has achieved acceptance as a *de facto* standard for describing WS composition (Cambronero, Diacuteaz et al. 2009). Nonetheless, the drivers of these choreography languages have been developed and refined based on various requirements For example, a detailed comparison of the existing literature on choreography languages can be found in (Bucchiarone and Gnesi 2006; Cambronero, Diacuteaz et al. 2009), giving a full semantic descriptions for all stages of the web service lifecycle. Based on our problem space, we found WS-CDL to be the most suitable choreography language because it is designed for describing abstract business processes and focuses on web service architecture (Bucchiarone and Gnesi 2006). Indeed, it concentrates on role representations that can be used to simulate roles in business processes for description of participant behaviour in a collaboration of services. Moreover, the WS-CDL is based on a formal language (derived from the π-calculus) which allows us to ensure the correctness of service behaviour based on behavioural type checking (Ross-Talbot 2004; Li and Miao 2008).

An overview of the elements and structure of WS-CDL, as described in the WS-CDL v1.0 specification (dated 9 November 2005), is at the W3C candidate recommendation stage (W3C 2005). WS-CDL is an XML-based language that describes the observable behaviour of peer-to-peer collaborations (i.e., multiple services), using message exchanges to accomplish a common business goal (Bordbar and Staikopoulos 2004). It defines the relationships among activities through executed interactions by means of message exchanges among web services described in WSDL. It is also an independent platform and business process implementation language, specifically designed for composing. Figure 2-5 shows an overview of the WS-CDL package in a set of type definitions, and it can be seen that the WS-CDL code consists conceptually of two parts: the package root elements, and the choreography definition.

The package root elements define both the exchanged messages and collaborating participants responsible for the observed behaviour. An *informationType* element specifies the type of exchanged messages and variables (e.g., capturing the state of a purchase order during the order creation routine of a business process). The *token* and *tokenLokator* elements refer respectively to relevant data pertaining to variable values, and how to access the token information in other resources. The *roleType* element represents the behaviour of the collaborating participant. It refers to one or more exhibited

behaviours (e.g., operations in WSDL file) and optionally identifies associates if the implementation supports web service interfaces. The *relationType* element consists of two roles (roleType), optionally including a subset of their collaborative behaviours. The *participantType* element groups roles (roleType) to which they will be executed by the same participant. The *ChannelType* element describes behaviours of a participant as a message recipient (rather than a requestor of messages) in order to specify both how exchanged information is passed and the target destination. Figure 2-5 illustrates a view of package root elements.

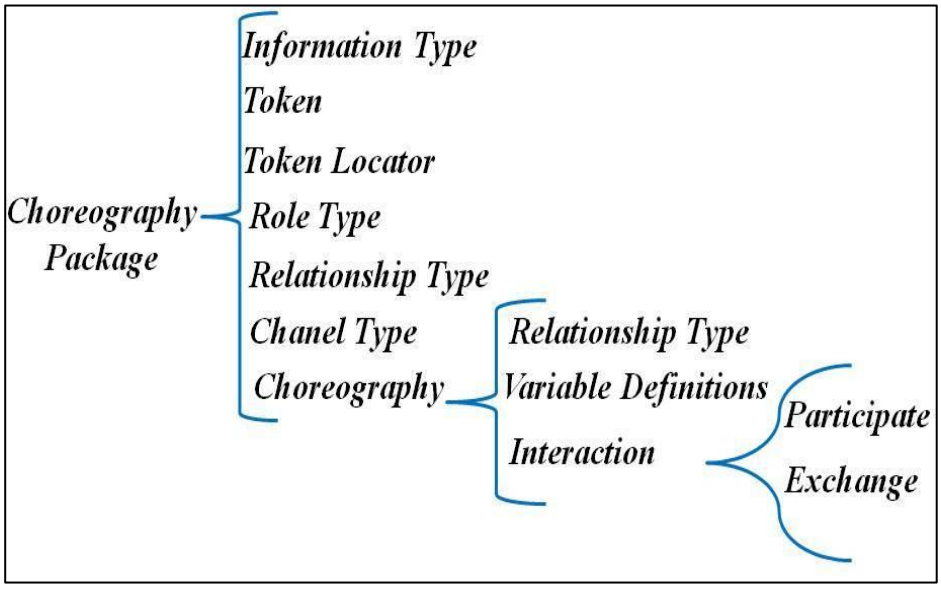


Figure 2-5 A View of the WS-CDL Package Root Elements

One or more choreographic definitions are included in every package. The choreographic definitions can be globally defined without the root package and other choreographies can invoke it when needed. The choreography section defines collaboration rules between two or more participants, and Alistair et al. (Alistair, Dumas et al. 2005) summarise activities in WS-CDL into three categories: control-flow activities, *workunit* notation, and basic activities. The first category can be subdivided into *sequence*, *parallel*, and *choice* elements, with these elements expressing the ordering structure by which interactions are executed. The second category, the *workunit* element, describes required conditions for successful execution of collaborations and synchronisation among activities. These conditions might include activity looping, guarding, exception handling, and coordination. Finally, basic activities include the following elements: *interaction, perform, assign, noAction, silentAction* and *finalize*. These

describe the lowest level actions performed within a choreography definition. Figure 2-5 illustrates a view of choreography definitions.

The W3C has promoted the suitability and stability of WS-CDL as a choreography language, based on web services from a global viewpoint (W3C 2005; Decker, Overdick et al. 2006), however there are some specific criticisms of the current version that could affect the definition of corresponding modelling notations in the context of SOA (Alistair, Dumas et al. 2005). An example is the integration of the XML syntax and semantic (meta-model) of service choreography into one specification, which affects the definition of an interchange format and modelling constructs (Alistair, Dumas et al. 2005). In addition, WS-CDL is bound to the WSDL interface with limited implementation (ISO/IEC 2007).

2.3 Business Process Modelling

A Business Process (BP) is a set of tasks or activities which is performed collaboratively to realize an overall business objectives (Medjahed, Benatallah et al. 2003). These objectives are achieved by using services which can adapt to requirements changes rapidly. Business Process Management (BPM) governs and controls BP in workflows, in order to improve agility and integrity. Business process modelling is the activity of representing and analysing business processes (Luo and Tung 1999), and a number of business modelling languages and tools have been proposed to model, implement, and execute these models. Among these modelling languages are the UML EDOC Business Processes, the PCD (Process Chain Diagram) of ARIS, and the activity diagram of UML (Unified Modelling Language). There are also ebXML BPSS and BPMN which are intended to be mapped to execution languages such as Business Process Execution Language (BPEL), XML Process Definition Language (XPDL) (Coalition 2008) and Web Services Business Process Execution Language (WSBPEL) (OASIS Standard 2007). A model in BPMN can be executed in a process-executable environment on a process engine (Genon, Heymans et al. 2011). The adoption of process modelling using BPMN 2.0 as the modelling language in this research is motivated by several factors:

- Relevant research has confirmed that process-oriented modelling provides a good basis for SOA (Rolland and CentreKaabi 2007; Jamshidi, Sharifi et al. 2008).
- BPMN 2.0 supports rich constructers. There are limitations when modelling related resources and representing various types of control-flow constructs using other modelling languages such as UML 2.0 Activity Diagrams for business process modelling (introduced by OMG) (Decker, Overdick et al. 2006).
- BPMN 2.0 focuses on extensibility in choreography descriptions.

2.3.1 Business Process Model Notation (BPMN)

BPMN is the leading standard among modelling languages for business processes and workflows (Chinosi and Trombetta 2011). BPMN is an OMG specification, which was initiated by a working group within the Business Process Management Initiative (BPMI), and then completed and published by OMG in February 2006 (version 1.0) (Recker, zur Muehlen et al. 2009). The initial goal of BPMN was to provide a standardized graphical notation that is comprehensible by business analysts and developers, without a native serialization format. The updated specification of BPMN was released in January 2008 and January 2009 as versions 1.1 and 1.2 respectively. These updates included better-defined semantics, such as various types of events (OMG 2008; OMG 2009). The most recent specification is BPMN 2.0, in which the focus and capabilities from previous versions have apparently been changed and extended (OMG 2011). This version formalizes the execution semantic for BPMN elements, provides extensibility capacity for processing models and graphical data, refines event composition and correlation, enables mapping of business process models in BPMN to other models, updates the semantic and definitions of human interactions, and extends its scope to define choreography and conversation models (OMG 2011). It also resolves some of issues with previous versions such inconsistencies and ambiguities. Moreover, it defines a meta-model and a schema for diagram interchanges, unlike previous versions that failed to provide an official meta-model (List and Korherr 2006; Debnath, Zorzan et al. 2007; Korherr and List 2007; Recker, zur Muehlen et al. 2009). According to the BPMN 2.0 specification (OMG 2011), diagram types include:

1. Process Diagrams: these contain description of flow elements and attributes used in a stand-alone business processes (orchestration), private non-executable processes for documentation, private executable processes for modelling and execution, and public processes for describing interactions between a private business process and another process or participant (see an example in figure 2-6).
2. Collaboration Diagrams: these consist of two or more participants communicating via a communication route known as a *message flow*, which considers the internal behaviour within business processes. Participants representing other business processes are assigned a role in a business interaction. These diagrams are designed to show the relationship between choreography and orchestration processes (see an example in figure 2-7).
3. Choreography Diagrams: these define interactions and communication protocols among participants using sequences of message exchanges. In contrast to orchestration concepts, this interaction description is based on Message Exchange Patterns (MEPs - see an example in figure 2-8).
4. Conversation Diagrams: an informal description of a collaboration diagram focusing on a logical grouping of message exchanges based on a correlation key, e.g., grouping of message exchanges for a specific object. (See an example in figure 2-9).

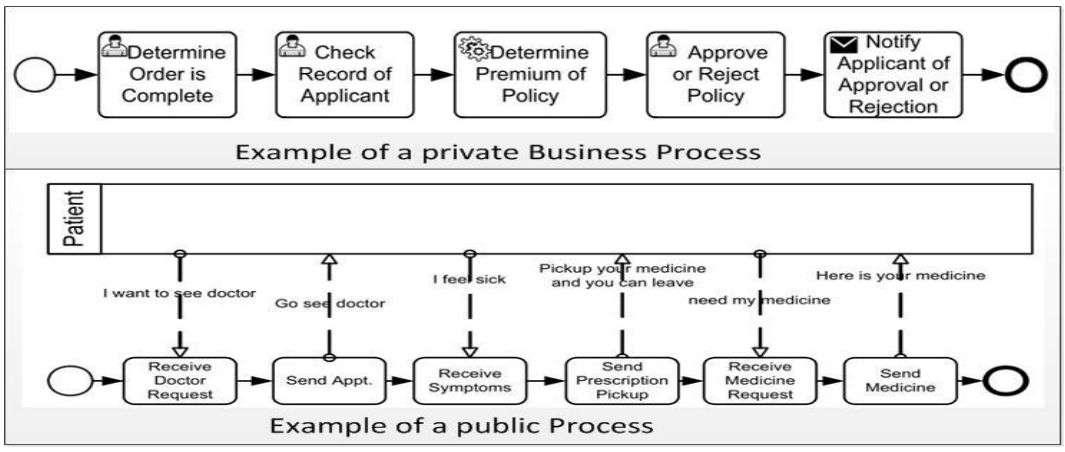


Figure 2-6 A Process Diagram Examples

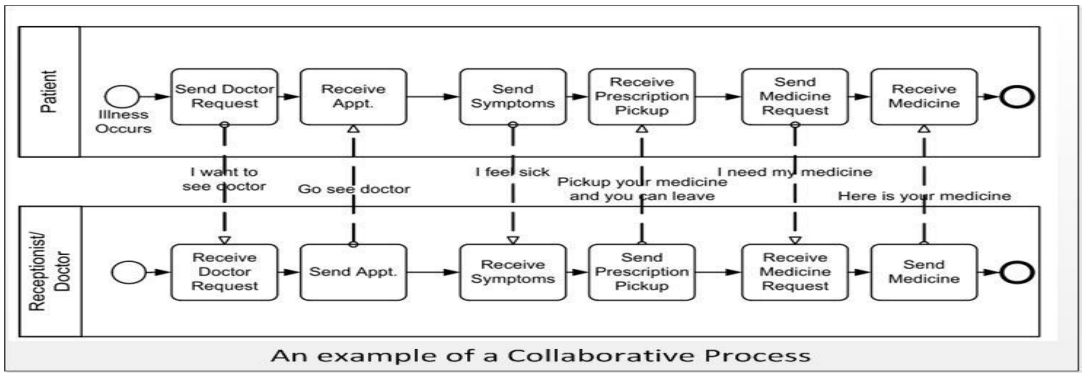


Figure 2-7 A Collaboration Diagram Example

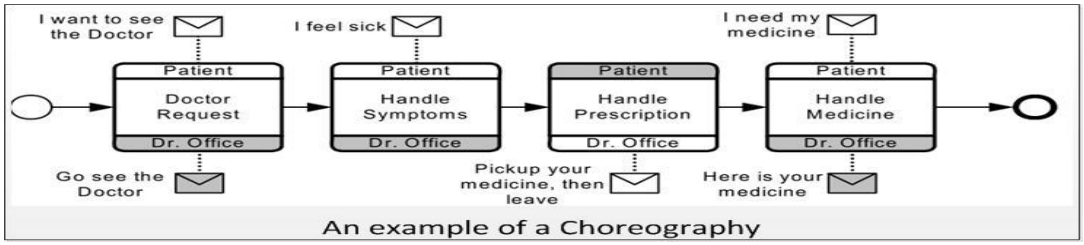


Figure 2-8 A Choreography Diagram Example

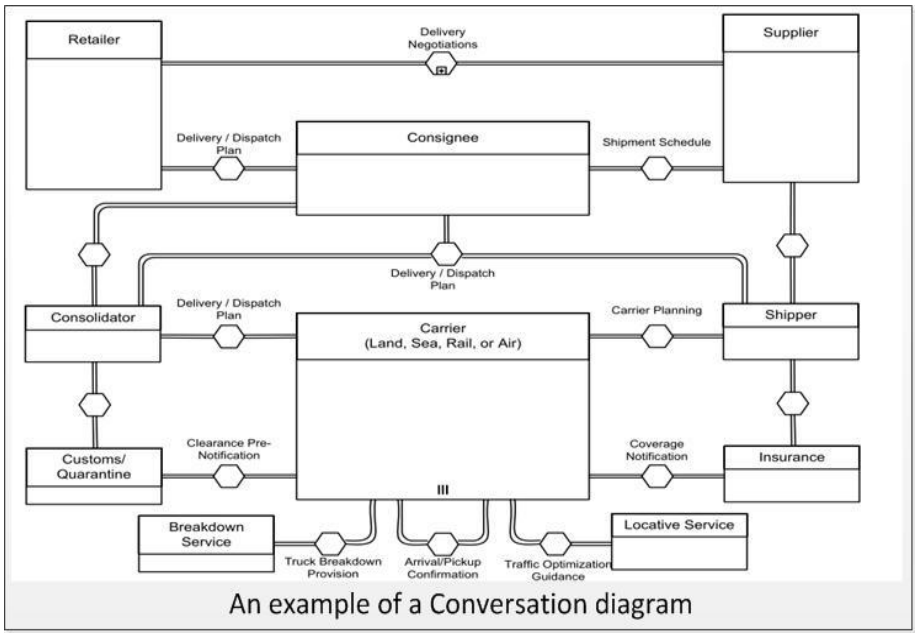


Figure 2-9 A Conversation Diagram Example

2.4 Model-Driven Transformation

The model-driven transformation (MDT) technique in MDA is used to develop a software program which can transform abstract models to code. Thus, the MDT can be used for SOA to generate service implementations from abstract models such as business process models. Although Software Development

Environments (SDE) (e.g., the integrated development environment (IDE)) have greatly improved in the past twenty years, software complexity and development costs continue to rise (Mellor, Scott et al. 2004). In order to develop software systems efficiently, the Object Management Group (OMG) has supported and defined the (MDA) as "software development processes based on a model" (OMG 2003). MDA is "an evolutionary step that consolidates a number of trends that have gradually improved the way we produce software" (Frankel 2003). To support a chain process of transformation, MDA requires the compliance and portability of standards such as Meta-Object Facility (MOF), Unified Modelling Language (UML) and XML Metadata Interchange (XMI) (OMG 2003).

2.4.1 Model Driven Architecture (MDA)

MDA is based on models that are defined using meta-meta-models, with every model based on a unique meta-model possessing precise vocabularies and auxiliary properties (Bezivin, Hammoudi et al. 2004). A model, as a primary artefact, presents statements about a system for a specific goal (Bezivin and Gerb 2001; Seidewitz 2003). Different forms can be used to describe a model, such as a general-purpose modelling language (which is a specific meta-model dependent) e.g., using a UML class model to describe detailed design of software systems. The aim is to have a model presenting a system, and defined according to a recognized standard. Models are transformed to other models, executable code or text using transformation languages.

An MDA increases the level of abstraction by separating the specification and business logic of a system from its software platform (Kleppe, Warmer et al. 2003). Conceptually, the level of abstraction in an MDA is designed according to three levels: the Computation Independent Model (CIM), the Platform Independent Model (PIM), and the Platform Specific Model (PSM) (OMG 2003). Models defined on the level of CIM correspond to business models in that they have a pure business specification; the focus is on the system environment, with little relevance afforded to how the software system is built. The PIM describes a system from a platform independent viewpoint, showing that the model description is sufficient to define system behaviour, e.g., a class diagram presenting the structure of a system. If the CIM separates business

specification from the design, the PIM separates the design of the system from implementation. The modelling languages used (e.g., plain UML, Executable UML (Mellor and Balcer 2002) combining UML with OCL) are an important factor in the quality of PIM models, however, the model must also have a high level of completeness and consistency (Kleppe, Warmer et al. 2003). The PSM describes a software system from a specific platform (OMG 2003), combining the PIM specifications with additional information about a specific platform, i.e., information about a specific operating system that impacts software systems.

A meta-model describes the properties and constructs of every model precisely. For the definition of such concepts, the OMG determines a meta-model architecture definition based on four layers of abstraction: M0, M1, M2, and M3. Figure 2-10 shows classical metadata for a (place order) business process modelled in BPMN. According to the definitions of these levels, M0 presents runtime-environment instances (e.g., a Customer with id=AAA places order_id=10 into a shopping cart_id=AAA100), M1 presents the model (e.g., a business process defined using a BPMN model), and the meta-model resides on level M2 where the transformation rules are defined, i.e., rules defined using OVT (OMG 2002) or OCL (OMG 2006). Those meta-models are always dependent on a common meta-meta-model (MOF) which is represented at level M3 (OMG 2008). Any meta-model frameworks of MOF dependent comprises of the four meta-layers.

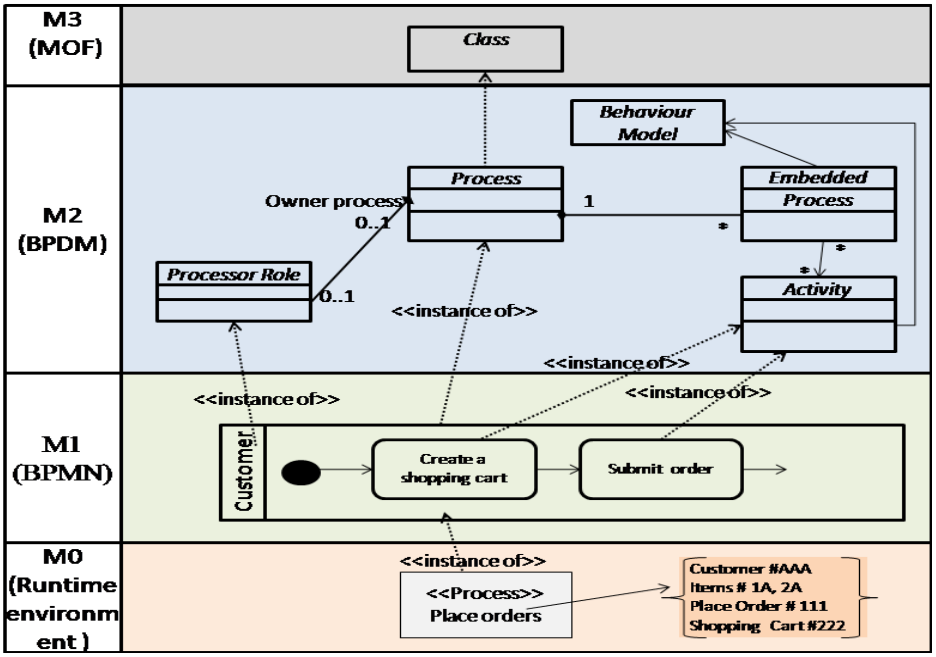


Figure 2-10 MDE Architectural Abstraction Levels

2.4.2 Meta-Modelling Supported Standards

MOF represents a set of modelling elements used in the specification and development of meta-models in a domain-specific modelling environment, and exists at level M3 (Frankel 2003). The definitions of the meta-meta-models are MOF dependent, and MOF can be also used to define non-Object-Oriented using meta-meta-models (Frankel 2003), i.e., using the Rational Unified software development Process (RUP). It supports the metadata management which binds a model to its meta-model (OMG 2002). UML is aligned with MOF and based on a four-layer meta-model architecture (Frankel 2003). As a graphical modelling language, UML provides MOF with the basic constructs to define and visualize meta-models. XML based Meta-data Interchange (XMI) is a specification language that defines rules for exchanging interchange format (e.g., metadata). Figure 2-11 shows the MOF architecture in an example of definitions of a business process in BPDM.

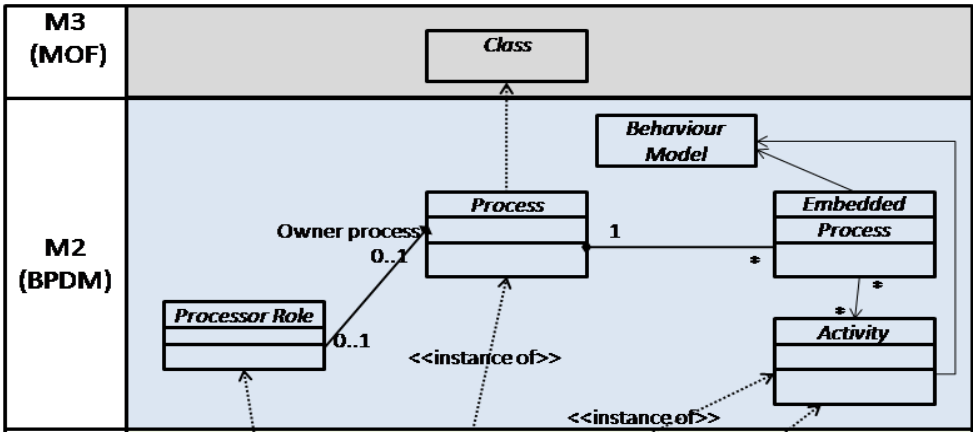


Figure 2-11 An Example of MOF Architecture

2.4.3 Model Transformations

The foundations for transformation in MDA come from theoretical computer science and practices within software engineering such as rewrite systems and complier construction (Davis, Sigal et al. 1994; Biehl 2010). In the context of the model-driven architecture (MDA), the Object Management Group (OMG) defines model transformation as "the process of converting a model into another model of the same system". Model transformations are a core element in Model Driven Engineering (MDE), providing a seamless way to process source models in order to generate, filter, and update target models. The transformation modelling languages achieve different types of transformation such as Model-To-Model or Model-To-Code. The transformation always depends on a model, to which it presents a set of statements about some particular systems.

The representation of these statements can be achieved graphically (Hidaka, Hu et al. 2009) e.g., a model might represent different level of abstractions of systems as views. A modelling transformation can be achieved either through a rule-based transformation (Debnath, Zorzan et al. 2007; Benaben, Touzi et al. 2008) or by the use of parameterized patterns (Brahe and Bordbar 2006; Delessy and Fernandez 2008). The transformation mechanism can be used in different phases of the software development cycle, for example, in development of a transformation program for software quality control to detect bugs (Bezivin, Bruneliere et al. 2005). Figure 2-12 shows some examples of model-transformation-mechanism use during different phases of a general System Development Life Cycle (SDLC) (e.g., transformation of functional

requirements to UML class diagrams, and then to Java skeleton code using model-to-code method transformation).

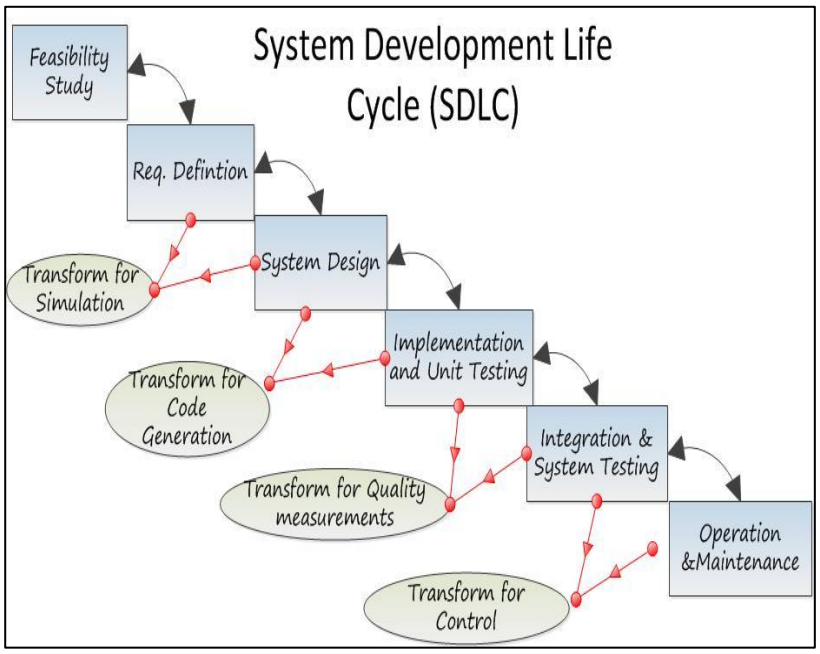


Figure 2-12 Model Transformation during System Development Life Cycle

As the applicability of model transformation has grown, a number of transformation languages offering many features have been proposed, under both open source and commercial licenses (Czarnecki and Helsen 2006; Milanovic 2007). These underlying transformation languages and approaches include Query/Views/Transformations (QVT) (OMG 2002), ATL (ATLAS Transformation Language) (Jouault and Kurtev 2006), Extensible Platform of Integrated Languages for model management (Epsilon (Kolovos, Rose et al. 2012), KerMeta (Moha, Sen et al. 2010), and XML Stylesheet Language Transformations (XSLT) (W3C 1999). The evaluation of different transformation languages and tools can be found in references (Czarnecki and Helsen 2006; Biehl 2010). In this thesis, we have adopted the ATL transformation language in an exogenous transformation (a type of transformation when source and target models are defined in different languages), for of the following reasons:

- ATL is described by an abstract syntax (MOF meta-meta-model).
- ATL provides a complete transformation model and supports several advanced features and complex transformations, e.g., it supports a number of source pattern elements.

- ATL has gained extensive support for development from the user community via discussion and available projects have been implemented in ATL for various examples and case studies.

2.4.4 ATLAS Transformation Language (ATL)

ATL (ATLAS Transformation Language) was first proposed by the Atlas Group and the TNI-Valisos Company as a model transformation language in response to the MOF/QVT for transformation implementation (BEzivin, Jouault et al. 2003). It also provides a modelling transformation platform to transform a set of source models into a set of target models (the semantics of involved models are defined in MOF meta-models or meta-meta-models.) It is a hybrid language supported by declarative constructs for less complex mappings, imperative constructs for advanced mappings, and offers the capacity to handle queries, views and transformations. In the context of model transformation, it consists of different rule styles (e.g., called rule and matched rules) dependent on the invocation method and targeted results, supported by concepts of polymorphism and inheritance. The "helper" construct defines global variables and functions expressed in the Object Constraint Language (OCL) standard. ATL is developed on top of the Eclipse environment as an Integrated Development Environment (IDE) supported with development tools (e.g., compiler and debugger, etc.), and an ATL transformation engine is used to compile and execute ATL programs. Figure 2-13 shows an overview of transformation models using ATL. A source model conforms to a specific meta-model, whereas a target model conforms additionally to a meta-model. The source meta-model and target meta-model conform to a standardised meta-meta-model (such as MOF or Ecore). The ATL program defines the transformation rules that enable generation of the target model based on the source model input.

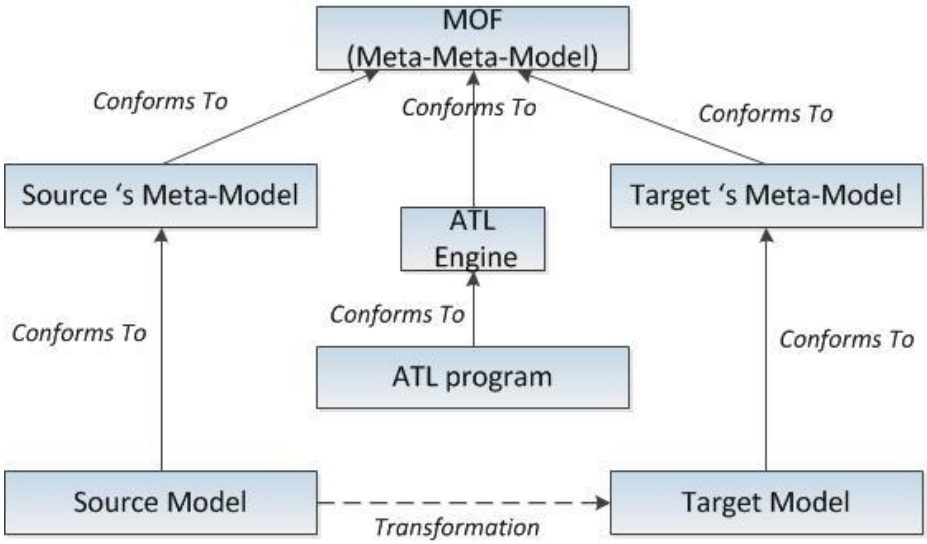


Figure 2-13 General View of Model Transformation

2.5 Summary

This chapter introduced the research fields of service modelling for service-oriented systems and MDA. We gave an overview of SOA, and showed how SOA covers a wide spectrum of enterprise architecture, as well as providing an illustration of the SOA hierarchical layers. We explained the traditional lifecycle phases of SOA development, focusing on service modelling that aims to use theoretical foundations to model service-oriented systems. Ambiguity remains about a universal-accepted definition for a service, however we adopted the service definition by W3C because it defines the key characteristics of a service (such as being self-contained). Then, we described the strategy of service-oriented decomposition process with elaboration on a service identification process to define the 'right' services with the appropriate level of granularity.

Service granularity might impact other software quality attributes, service granularity being classified into different types based on amount of exchanged data, functionality, and the level of abstraction. Software product metrics are possible methods to measure service granularity. WS-CDL is a suitable choreography language to bridge the gap between business process models and web service architecture. We also discussed the concept of business process modelling and introduced BPMN as a suitable business modelling language to depict business processes complex scenarios for service-oriented systems. We introduced the MDA approach and described the architectural aspects behind

it. We used a metadata example to demonstrate the model-driven engineering (MDE) four-layer of abstractions. We described the model transformation method that converts models of different types to new models or source code. Finally we introduced ATL as the adopted transformation language and implementation framework for this thesis.

The next chapter will focus mainly on current research approaches in service identification and will contain a comprehensive literature summary. We will also discuss how to measure service quality attributes and how these measurements can be used within the service identification process.

Chapter 3 Service Identification Current Approaches

In Chapter 2, we gave an overview about the research areas of SOA and MDA. In this Chapter, we provide an overview of current literature in the area of service identification and highlight the limitations of existing approaches.

This literature review is divided into three main sections. In Section 3.1, we discuss the area of service identification and classify existing approaches into three sub-sections: business-driven, ontology-driven and legacy system-driven. In Section 3.2, we briefly cover research into quality of service (QoS) in the context of service identification. In section 3.3, we evaluate the approaches identified in section 3.1 using a number of criteria. Finally, Section 3.4 concludes by summarizing the main findings.

3.1 Service Identification Methodologies

A considerable amount of literature has been published in the area of service modelling covering various different contexts. Although service modelling covers both the analysis and design phases, most of the current research focuses on either the analysis or design aspects individually. A number of systematic literature reviews and surveys have been undertaken on service modelling issues, mainly concentrating on service identification (Papazoglou and Van 2006; Bianchini, Cappiello et al. 2009; Kohlborn, Korthaus et al. 2009). The research methods of these reviews were based on predefining a set of SOA characterises or aspects to evaluate and then comparing the results, e.g., comparing supported phases of SOA life cycle and targeted types of service.

The important result for this research is to investigate these different delivery strategies and select a suitable strategy for service identification. The design strategy of service-based systems is generally classified into top-down, bottom-up and meet-in-the-middle approaches (further details in section 2.2.1). We can classify the research conducted in service identification for service-based systems into: business-driven, ontology-driven and legacy system-driven. Each of these approaches will be considered in the next sections.

3.1.1 Business-Driven Service Identification

A rapid response to changing business requirements is one of the important objectives of the SOA approach. Thus, some researchers argue that SOA does not merely integrate an IT infrastructure, it must also fully take into account the underlying business models (e.g., business process, use case, and activity diagrams) (Papazoglou and Van 2006; Kim and Doh 2007). A top-down analysis technique might identify services seamlessly mapping from business processes or use cases (Galster and Bucherer 2008). SOA can be specifically differentiated from other software methodologies because it is explicitly intended to be strategically aligned with the underlying business vision. The decomposition of business processes or business functions is a key technique to drive a top-down strategy.

Galsters and Bucherer propose a graph-based framework that discovers service granularity according to specified business goals during the design phase (Galster and Bucherer 2008). However, this approach does not define fine-grained services and only quantifies coarse-grained services using a non-technical description. Kim et al (Kim, Kim et al. 2008) focus mainly on how to define the right services in the analysis phase in respect of business change factors and goals. Rolland and CentreKaabi (Rolland and CentreKaabi 2007) introduce an approach that depends on exploring the purposes of a business process in order to identify a service. As a result, this approach defines a new type of service called an "Intentional Service" which considers business goals, pre- and post-conditions, and different interpretations instead of the technical aspects of interfaces, behaviour and composition of services respectively. Nayak et al. attempt to solve the gap between service provider and requester regarding the service agreements (Nayak, Nigam et al. 2006). A Unified Service Model (USM) is proposed along with a service operational model to specify business

services from a business perspective. Although the authors of this approach claim that the USM defines business services at multiple levels of granularity, no metrics or guidelines are provided to identify the service granularity.

Arsanjani and Allam (Arsanjani and Allam 2006) outline a set of activities in the analysis phase that lead to an adequate broad foundation for service identification. The authors classify service types into three layers; orchestration layer (process service), business layer (task and entity service) and application service layer. Even though the author emphasizes the importance of the key principles of service-oriented, no details have been provided as to how those principles can be applied to guarantee optimised services. Boerner and Goeken (Boerner and Goeken 2009) provide a general approach to identify services including economic aspects, e.g., service robustness decreases operations costs and SOA governance, e.g., considering SOA lifecycle to prevent service redundancy. It also emphasizes the importance of BPM as foundation for service identification concerning appropriate standards. The authors of this approach mention broad aspects implicitly with neither practical guidelines nor specific process details. Shirazi et al. (Shirazi, Fareghzadeh et al. 2009) attempt to categorize services based on the operational state of services and logical presentations, i.e., differentiating between applications and business services. Then they use this classification to build their method that consists of several instructional steps to identify services. However, the approach is incomplete because the authors did not consider important elements that affect the service identification phase such as granularity and complexity.

Nuffel (Van Nuffel 2007) focuses on deriving guidelines for service identification from business requirements by means of a BPM language definition and an analysis of relative artefacts. Stewart and Chakraborty (Stewart and Chakraborty 2010) use the value chain and prioritization analysis technique to model business service and software services from business strategies and a business process model respectively. Kim and Doh (Kim and Doh 2009) define a formal method using graph clustering techniques. Cost metrics are used to evaluate interaction patterns between activities; a UML activity diagram represents the business model as input of the method. Dwivedi and Kulkarni (Dwivedi and Kulkarni 2008) introduce a semi-automated approach to identify services in process-oriented systems. It converts UML-based business process models into XMI. The XMI reader (NSUML) is used to produce the MOF (Meta Object Facility) for mapping XMI

meta-model. The algorithm used runs over an XMI meta-model developed using a statistically-based approach which is used to create APIs to query candidate services. Although this approach provides a good definition of the service identification issues and presents an interesting tool, it fails to demonstrate how the tool will integrate the service hierarchy layers and properties.

In order to realize the potential of SOA and address the lack of detailed approaches, researchers have sometimes considered full-cycle approaches for SOA development and design. The objective of this approach is to support various phases of service-based system and resolve the issues related to the internal activities of every phase of the SOA development cycle, e.g., issues related to service identification process in the service modelling phase. Papazoglou and Van (Papazoglou and Van 2006) suggest a full cycle development methodology for web services, based on other development models such as Rational Unified Process (RUP), Component-based Development and Business Process Modelling. It is an iterative and incremental methodology with six phases: planning, Analysis and Design (A&D), construction and testing, provisioning, deployment, execution and monitoring. It also discuses characterises and principles of service-oriented design and development.

Erradi et al. (Erradi, Anand et al. 2006) introduce the Service Oriented Architecture framework (SOAF) approach with five conceptual levels: information elicitation, service identification, service definition, service realization and roadmap with planning. Each level requires inputs to proceed with a set of activities that deliver outputs as inputs for the next layer. It captures the "As-is" and "To-be" business models to identify business services and then maps the captured business processes of existing applications to determine potential functionalities within the candidate business services. To identify the optimal services, it capitalizes on the top-down approach for domain decomposition and the bottom-up approach for application portfolio analysis using manual techniques (e.g., interviews and questionnaires) and also uses automation tools (e.g., IBM's Asset Analyser). For service identification, it defines the design tasks to be performed (e.g., by specifying a service policy). It also scales service granularity levels by means of grouping the number of invoked components or services via one operation on a service interface and number of updated sources. Transformation strategies are defined along with the plan for service implementation. The SOAF illustrates merely these steps at

conceptual level, which neglects explaining details of relevant design issues of every level.

Evaluation and validation play an important role in the applicability of the proposed methodology, especially with significant differences between SOA development compared to formal software development approaches (Haines and Rothenberger 2010). Erradi et al. (Erradi, Kulkarni et al. 2009) extend the service design concepts of the SOAF framework (Erradi, Anand et al. 2006) with a business-driven approach built on top of a meta-model based on a practical service design process from a real case study. It highlights broad guidelines for enhancing the service granularity such as reusability, business alignment, designing for assembly, and reducing the ripple effects of application changes.

From experience with industry practices and implementation of several real projects, IBM (Arsanjani 2004) introduced Service-Oriented Modelling and Architecture (SOMA) as a service-oriented modelling methodology. For modelling services, it defines three steps: identification, specification, and realization, and includes flows and composition of services. Although SOMA was successful in highlighting the broad architectural aspects, it could not provide detailed implementation guidance. This methodology explains how to identify the important aspects of service modelling, not how to implement them. However, we believe that SOMA has gained acceptance in industry (Lane and Richardson 2011) because it is driven from real case studies, which increases its validity and applicability (in contrast to many other methodologies) (Rolland and CentreKaabi 2007; Galster and Bucherer 2008; Kim, Kim et al. 2008). Moreover, at the time that SOMA was published there was very little available research in the area of service modelling. Further research based on the SOMA methodology was conducted to enhance SOMA by learning from its adoption and past usages which have turned it into a "fractal model" for service-oriented software development (Arsanjani, Ghosh et al. 2008). The fractal model refers to enablement of the SOMA method to evolve in an approach as needed during different phases of the software development life cycle. Recent work advances SOMA usage to leverage method components and patterns. Zhang et al. (Zhang, Zhou et al. 2008) also extend SOMA to providing SOMA-ME as a platform for model-driven design to provide tools and design and development environment for SOA solutions. This

is an integrated development environment (IDE) which facilitates the evaluation, design, and validation of service models.

3.1.2 Ontology-Driven Service Identification

The service identification process always concurs with decomposition mechanism between architectural layers which results in model (semantic) transformations. During the transformations of models, semantic inconsistency might occur. To cope with semantic inconsistency, some researchers have used ontology-based approaches to identify services (Klose, Knackstedt et al. 2007).

Semantic web concepts, standards and technologies (such as the Web Ontology Language OWL) have wider applicability in the world-wide web and can also be used for model automation and validation (Tetlow, Pan et al. 2006). Several research projects have utilised the concept of ontologies in the service identification process to build quality models as well as understand and capture essential elements from legacy systems (Yang, Cui et al. 1999; Dobson, Lock et al. 2005). Yousef et al. (Yousef, Odeh et al. 2009) propose a framework called "BPAOntoSOA" which defines a service-oriented model from a Business Process Architecture (BPA) based on two ontologies: BPAOnt (semantic definitions of business processes and candidate services) and QoSOnt (defining an ontology for quality of service) (Dobson, Lock et al. 2005). This framework has been developed for a specific domain (healthcare systems) and does not provide comprehensive supports for other domains. DongSu et al. propose a method to identify services based on semantic relationships derived from mapping an ontology and feature model (DongSu, Chee-yang et al. 2008). The tree-like structure that the feature model depends on does not clearly show the level of granularity, e.g., services that reside at similar level of granularity in the tree could offer different level of functionalities which means the granularity varies on one level.

Feng et al. (Chen, Zhang et al. 2009) use three different ontologies: the Domain Concept Ontology (DCO) provides knowledge about an application domain, the Functionality Ontology (FO) describes the functionalities of applications and the Software Component Ontology (SCO) describes software design patterns developed in the approach. These ontologies attempt to bridge the gap between the traditional technologies in legacy systems, and software and service-oriented technologies. Bianchini et al. capitalise on annotating

business processes to identify functionalities suitable to become candidate services semantically (Bianchini, Cappiello et al. 2009). A reference ontology that consists of atomic concepts and a set of semantic relationships between those concepts with a weight factor (assessing the degree of relationship) evaluates business process elements. However, this approach ignores a very important aspect of service identification which is granularity.

3.1.3 Legacy system-Driven Service Identification

In SOA, a green-field case often does not exist in software practices. Often a legacy system exists as a valuable asset that can be exposed and integrated with new developed services. There is a number of ways that such a legacy system can be used in SOA, e.g., developing a wrapper to shield legacy code.

Zhang and Yang (Zhang and Yang 2004) propose a hierarchical clustering algorithm to extract independent services from procedural software systems into an object-oriented (OO) models. This approach uses a grey-box strategy which is a combination of system wrapping together with the key business logic. It starts by first identifying services using domain analysis and then builds a domain model. The next stage is to build a process model after completing the assessment using a dendrogram to visualize results. A clustering technique is used to transfer procedural code to an object-oriented model, mapping between similar entities based on the underlying concepts. Finally, the candidate services from the Object-Oriented (OO) model and targeted constructed services are packaged with code refinements. Chen et al. (Chen, Li et al. 2005) discuss the transformation of legacy systems developed with Object-Oriented Design (OOD) or Component Based Design (CBD) into SOA applications using feature analysis. The feature analysis approach consists of three stages: identifying the system features, constructing feature models and tracing the relationship between the defined service operations and the source code using a feature location technique. To locate a specific feature in the source code, a re-engineering technology is required. The located source code is aggregated into a united module and the key features are associated with one or more services, as coarse-grained as possible. The identified service operations are exposed by class delegations using a tool called a Web Service Wrapper. Klose et al. propose a selective method which based on evaluation of methods from a business and technical perspective (Klose, Knackstedt et al. 2007). It defines a procedural model for service identification with three phases:

preparation, service analysis and service categorization. Each phase consists of tasks and related documents, integrating the aspect of stakeholders in the business process model to derive candidate business services at the service analysis phase.

Zou and Kontogiannis (Zou and Kontogiannis 2001) provide a framework to transform legacy systems into a web-enabled environment by means of a CORBA wrapper (consisting of a CORBA IDL, SOAP, WSDL, and UDDI). This approach is accomplished in three stages. Firstly, legacy code is decomposed based on application functionality. Secondly, the decomposed code is migrated using wrappers into CORBA distributed objects. Finally, SOAP/CORBA IDL is defined to unify the services. It suggests that legacy systems can be divided into four layers: standards and guidelines, basic common services, value-added functional services, and mission-specific services. This research does not provide enough detail on how to identify services along with the new business requirements and the targeted service characteristics. Jianzhi et al. (Jianzhi, Zhuopeng et al. 2005) develop a framework ICENI (Imperial College e-Science Network Infrastructure) to leverage the components of legacy systems into a grid environment. It then applies reverse engineering techniques to components using a Java Native Interface (JNI) wrapper to encapsulate code and the Commerce eXtensible Markup Language (CXML) to describe specifications for communication with the ICENI workflow.

Zhang et al. propose an architecture-based service-oriented reengineering approach that uses a hierarchical clustering method to identify services from legacy systems based on mapped requirements derived from UML models (Zhang, Liu et al. 2005). This approach requires human supervision to assist in determining the optimal service granularity along with the clustering technique. Aversano et al. suggest a approach that extracts description of services (WSDL) from legacy code as features (Aversano, Cerulo et al. 2008). An Information-Retrieval (IR) algorithm (Baeza-Yates and Ribeiro-Neto 1999) and matching algorithm (Kokash 2006) are used to evaluate candidate services. The IR algorithm is used to match the intended goal from the service to the extracted candidate features, whereas the matching algorithm calculates the lexical similarities and assesses the similarities between service elements. An extractor was developed that maps elements between source code and WSDL elements (class-to-service, method-to-operation and parameter-to-Message) and

textual documentation. An obvious drawback of this approach is neglecting important service designs aspects in service identification such as granularity.

Because of the complexity of most software systems, researchers often propose abstract models to simplify the descriptions of legacy systems. However, conceptual models describe only high-level activities in the core business processes (i.e., the business logic and rules are not included). As result, the resulting services are coarse-grained and have redundant functions. In a real-world project in Energy Management System (ESM), Wang et al. utilise specific enterprise service hierarchy patterns for selected business processes to determine the service granularity (Wang, HU. et al. 2007). This method focuses on a high level architecture which consists of four main service patterns: an execute pattern (i.e., a coordinating services), a broadcast pattern (i.e., to alert the enterprise when a business object is changed), a receive pattern (i.e., applying changes to a business object), and a retrieve pattern (i.e., responding to the consumer and returning data). Because of the simplicity of the implementation, only the broadcast and receive patterns were implemented. These patterns failed to provide effective guidelines to enhance the service granularity.

3.2 Quality of Service (QoS)

Most SOA researchers agree on the importance of software metrics to improve the quality of service–based systems. While the relative relationship between granularity and other SOA quality attributes has been discussed in recent research, few researchers have focused on measuring granularity as an independent factor which affects internal SOA structural attributes such as coupling and cohesion. 'Service granularity' is a measure of the exposed functionality of services. The service granularity of any service-oriented system indirectly affects typical SOA design qualities such as flexibility, reusability and performance. The granularity of service operations plays a key role in SOA quality attributes (Shim, Choue et al. 2008). This impact can be either positive or negative based on the tradeoffs adopted by the service provider. Coarse-grained services are usually advantageous because they improve overall performance, at the expense of reducing system flexibility. It is important that we differentiate between different types of granularity in order to analyse relative quality attributes. Haesen et al. and Karmarkar et al. (Erl, Karmarkar

et al. 2008; Haesen, Snoeck et al. 2008) propose different types of granularity which require different measurements (explained previously in section 2.2.2). The service types and the architectural level at which a service resides together can be used to define types of service granularity.

Shim et al. (Shim, Choue et al. 2008) propose a set of metrics for general SOA design including service and parameter granularity. In this research paper the service granularity metric is based on the number of operations and similarity between operations in a service. Parameter granularity is used to evaluate the ratio of operations with fine-grained parameters to the total service operations. However, these measurements lack any precise definitions for fine and coarse parameters in addition to any mechanism to define similar messages. Sindhgatta et al. (Sindhgatta, Sengupta et al. 2009) suggest a metrics suite for measuring the SOA quality attributes of service cohesion, coupling, reusability, composability and granularity supported with two real-life SOA design models. The proposed granularity metric counts number of services, operations, and messages, but is not particularly designed to quantify the granularity of a specific service. Senivongse et al. (Senivongse, Phacharintanakul et al. 2010) focus on the capability granularity which is the functional scope of a service. It traces fine-grained capabilities through web service invocations using association rules and the "Apriori" algorithm to guide a service designer to an appropriate implementation. Although invocation methods (synchronous and synchronous) play an important role in web service design, they are not specifically considered.

Measuring service granularity is also used as an indicator of SOA quality attributes such as complexity in compound services by counting the number of services in every individual component node (Zhang and Li 2009). Xiao-jun uses information theoretic principles to propose SOA metrics for coupling and well-chosen granularity (Xiao-jun 2009). The granularity metric in this case is based on the mutual information content of relative service operations and their usage occurrences. This metric groups operations that are used together into a single service. However, the metric does not provide any clarification of the appropriate information content it considers which could refer to several different aspects of SOA quality (e.g., dependencies between service, shared messages and invocation methods). Dobson et al. (Dobson, Lock et al. 2005) suggest a set of ontologies about QoS vocabularies, relative concepts, metrics, quality attributes (e.g., dependability, performance). To leverage QoSOnt

approach, the authors propose a prototype tool, called the Service QoS Requirements Matcher (SQRM), which is demonstrated with synthetic scenarios. QoSOnt develops a single ontology for every quality attribute aiming for extensibility and generality.

3.3 Analysis Comparison of Existing Approaches

In order to compare the methodologies that are proposed for service modelling (and in particular for service identification), specific criteria can be adapted from the relevant literature. Klose et al. (Klose, Knackstedt et al. 2007) provide criteria relating to the business-driven perspective using general SOA design principles, e.g., the starting point of the modelling such as business process model or software components. Kohlborn et al. (Kohlborn, Korthaus et al. 2009) suggest some criteria that suited mostly service analysis (rather than identification), although the number of approaches considered was significantly larger than similar reviews in (Klose, Knackstedt et al. 2007; Ramollari, Dranidis et al. 2007; Boerner and Goeken 2009). Gu, et al. (Gu and Lago 2010) (a more detailed review of service identification methods) define several classifications for methods, techniques, process, input, and outputs of service identification methodologies from a range of literature, providing a holistic overview. Classification types for every criterion are defined and applied for thirty collected heterogeneous approaches and with different scope. However, the criteria used when comparing service identification methods needs to be more focused on the way that services are actually delivered.

We defined a number of criteria: the criterion for delivery strategy, technique, lifecycle coverage, service types, quality aspects and granularity. In addition, we adopted the criteria of input and output of the modelling phase used by Gu et al in reference (Gu and Lago 2010). The descriptions and analysis of each criterion as follows:

Delivery strategy criterion: This is an important aspect of service analysis and design. This strategy is primarily used in the existing literature that applies comparative analysis. The three key strategies for SOA development mentioned detailed in section 2.2.1. The top-down strategy begins with a business analysis of requirements and business processes which can be implemented as business services. In contrast a bottom-up strategy analyses existing legacy systems and then defines technical services (Rosen, Lublinsky et

al. 2008), while the meet-in the-middle strategy combines both approaches. From these criteria, we can show the impact of every strategy on the process of service modelling. As a matter of fact there is no particular *de facto* strategy that can identify the "optimum" services in all possible application domains with all possible requirements. We found that two attributes affect the decision about which delivery strategy the enterprise should adopt: the status of the resources and the targeted service types.

Firstly, the "green field" (i.e., develop software from scratch) case does not usually exist in SOA, thus making effective use of existing assets such as legacy code has become an important part of the service development process. Erradi et al. (Erradi, Anand et al. 2006) classify approaches in integrating legacy systems as services into two broad categories: legacy integration (non-invasive) and legacy transformation (invasive). Legacy integration is a cost-effective and short-term solution (i.e., the business logic wrapping approach). The legacy transformation approach is more modular and typically uses an incremental migration process with both refactoring and consolidation of the business logic.

Secondly, consideration of the functional scope of different services is a key element required to construct a service taxonomy (Braunwarth and Friedl 2010). It is not about developing monolithic services; in contrast a service should accomplish certain goals that can be quantified and that correspond to a specific business or technical requirement. The top-down approach reflects business requirements and enterprise goals but will frequently deliver coarse-grained business services (Nayak, Nigam et al. 2006; Galster and Bucherer 2008; Kim, Kim et al. 2008). This approach is dependent on the representation and decomposition of business models which lacks the ability to capture the full requirements that can be seamlessly transformed to software artefacts. In other words, most authors agree that using the top-down strategy to transform business models directly to candidate services does not provide usable explicit service definitions. The decomposition can be achieved based on domains, processes, goals and requirements, and make use of particular analysis techniques. DongSu et al.; Yousef et al. (DongSu, Chee-yang et al. 2008; Yousef, Odeh et al. 2009) employ ontology approaches to business processes to conceptualise the requirements and relevant architectural aspects into one model of knowledge representation. Analysis techniques (such as clustering suggested in references (Zou and Kontogiannis 2001; Zhang and Yang 2004; Kim and Doh 2009) and feature extraction in (Chen, Li et al. 2005; Aversano,

Cerulo et al. 2008)) are also used to support the top-down strategy to achieve the final identification of candidate services along with decomposition of business models. In contrast, the bottom up strategy uses the existing legacy systems to define IT services (finely-grained services) (Braunwarth and Friedl 2010). The integration of existing legacy code into SOA can be achieved by integrating via adapters, which shields the legacy systems from the web service interface; this is sometimes called the "black-box" approach (Zou and Kontogiannis 2001; Zhang and Yang 2004; Chen, Li et al. 2005). Where appropriate, the important business logic of the existing code will be implemented as WS (Arsanjani, Ghosh et al. 2008; Aversano, Cerulo et al. 2008). A combination of a WS wrapping technique and the development of key business logic is widely adopted in the meet-in-the-middle strategy (Erradi, Anand et al. 2006; Papazoglou and Van 2006; Shirazi, Fareghzadeh et al. 2009). Meet-in-the-middle, as a hybrid approach, is the strategy most often suggested in the references (Kohlborn, Korthaus et al. 2009). The focus of approaches based on meet-in-the-middle is to deliver both business services and IT services (Brereton and Budgen 2000; Papazoglou and Van 2006; Arsanjani, Ghosh et al. 2008; Erradi, Kulkarni et al. 2009). The IT services require to combine outputs of both strategies top-down and bottom-up in order to enable service integrity by means of specific algorithms application (Zhang and Yang 2004) portfolio analysis (Dwivedi and Kulkarni 2008; Jamshidi, Sharifi et al. 2008). Decomposing the enterprise architecture of a system into different hierarchical level of abstractions defines various level of granularity (Erradi, Anand et al. 2006; Dwivedi and Kulkarni 2008). It is an approach to consider the scope of different services, e.g., the scope of utility services residing on an infrastructure layer, which responds robustly to provide specific granular functional scope to composite services rather than in business services.

Technique criterion: This describes the method that is used to implement the selected strategy. There are various techniques that could be adopted e.g., an approach using a top-down strategy might use a formal method and a graph clustering technique (Kim and Doh 2009) or components and RUP models (Papazoglou and Van 2006). Some approaches start from an enterprise perspective to achieve a set of strategic goals (often described as 'goal-driven') (Erl 2005; Galster and Bucherer 2008; Kim, Kim et al. 2008). According to Gu et al (Gu and Lago 2010), existing techniques for service identification approaches can be classified into six distinct types: algorithm,

guidelines, analysis, ontology, patterns and information manipulation. This classification is ambiguous because these types of techniques are sometimes combined and used at various phases of SOA development cycle, e.g., analysis techniques (such as clustering and features), performed to define business goals and processes repositories, are used initially at an early stage of several proposed approaches (Erradi, Anand et al. 2006; Klose, Knackstedt et al. 2007; Stewart and Chakraborty 2010) along with other techniques such as developing algorithms (Dwivedi and Kulkarni 2008) or guidelines (Van Nuffel 2007). As shown in the literature review, the majority of research is based on the use of business models to represent software requirements and understand the key business requirements. Furthermore, service properties and SOA design principles are already defined by SOA practices. Although the separation of modelling details from implementation is a key design principle, the current proposed SOA modelling approaches suffer from a rigorous separation of concerns (Haeng-Kon 2008) which increases the abstraction gap between the models represented and their implementation. Therefore, a successful technique should be able to transform a business model to a set of service with appropriate implementation and integrate the two phases seamlessly. MDA appears to be the appropriate technique to maintain the balance between levels of details in the different level of abstractions.

Lifecycle coverage criterion: To a limited extent this approach covers the complete SOA development lifecycle (discussed in section 2.1.1). This criterion is primarily defined in the literature of service analysis and expressed using different terms by different authors (Klose, Knackstedt et al. 2007; Kim and Doh 2009; Kohlborn, Korthaus et al. 2009). It is noticeable that some approaches limit their scope to specific phases (such as modelling) (Arsanjani and Allam 2006; Boerner and Goeken 2009; Chen, Zhang et al. 2009), while very few approaches attempt to fulfil all potential SOA lifecycle (e.g., references (Papazoglou and Van 2006; Arsanjani, Ghosh et al. 2008)). With reference to the modelling phase, because there are no standardised approaches, typical activities depend on the focus of the approach adopted and the specific technique used. Klose et al. (Klose, Knackstedt et al. 2007) make use of this approach by identifying business services from a business perspective using a manual stakeholder in three phases: preparation, service analysis and service categorization. One mature approach proposed by industry is based on extensive empirical evidence and defines three main phases for service-oriented

modelling : 1) service identification (this identifies candidate services based on goal-service modelling, domain-decomposition or existing asset analysis; 2) service specification which constructs service elements specifications (both interface and message) together with service dependencies and interactions; 3) service realization which implements details specifications of service elements and components. These phases are widely adopted by later approaches according to a recent systemic literature review on process models for service-based application (Lane and Richardson 2011). Whether or not proposed approaches consider the SOA lifecycle fully or partially, it is important to bridge the gap between the modelling phase and other SOA lifecycle phases in order to identify the right candidate services.

Service Types criterion: Achieving the definitions of the candidate services is the goal of the modelling phase. There are several different classifications proposed to define service types from various viewpoints. The service classification is often defined based on the added value of the service from the business or IT perspective (Gu and Lago 2010) or alternatively by layering the enterprise architecture into hierarchical levels (Erradi, Anand et al. 2006; Rosen, Lublinsky et al. 2008). While process services are derived depending on the collaboration of several business services, IT services are required to support the operational goals of business services. Kohlborn et al. refer to this criterion as the "SOA Concept" which indicates whether the focus of an approach is the business services or the software services or both (in this context 'software services' refers to the execution of business services). In other words, software services represent all service types apart from business services (despite the different levels of abstraction among software services such as data services, infrastructure services, etc.). Gu and Lago (Gu and Lago 2010) define four types of services, whereas Kulkarni and Dwivedi (Kulkarni and Dwivedi 2008) classify services into seven types. This difference in definitions of service types deduces the architectural layering adopted in the approach. In a more business-goal-oriented interpretation of the service type, Rolland and CentreKaabi (Rolland and CentreKaabi 2007) define a new type of service called an "Intentional Services" which ignores completely the functionality provided by the service. A comprehensive classification for service types in terms of properties and characterises is required rather than a modification of an existing architectural layering with the addition of special-purpose services.

The existing classifications are misleading because they are based on the level of decomposition that has already been adopted in an enterprise.

Design Input criterion: the type of resources available affects the decision about which strategy to use, e.g., legacy code sometimes represents a valuable asset for enterprises and this needs to be taken into account. Thus, the process of service identification needs to start with detailed analysis techniques (Chen, Li et al. 2005; Wang, HU. et al. 2007; Aversano, Cerulo et al. 2008) or reengineering methods (Jianzhi, Zhuopeng et al. 2005; Arsanjani and Allam 2006; Papazoglou and Van 2006; Erradi, Kulkarni et al. 2009) or both of these techniques used together (Zou and Kontogiannis 2001; Zhang and Yang 2004; Erradi, Anand et al. 2006; Arsanjani, Ghosh et al. 2008) to extract valuable code. Using the same strategy does not imply identical inputs, i.e., different types of representations and semantic of business models will provide different level of detail. For example, some researchers use a top-down strategy with similar types of input (intended requirements and goals) and they all result in different types of outputs - from a very abstract description (a list of services) to complete service profiles (detailed descriptions of services) (Rolland and CentreKaabi 2007; Galster and Bucherer 2008; Kim, Kim et al. 2008). In the case of a "green-field" SOA project (i.e., completely from scratch), the goals and business requirements in the form of business models are used to provide structural (Kim and Doh 2007; Rolland and CentreKaabi 2007; Kim, Kim et al. 2008; Stewart and Chakraborty 2010) and behavioural descriptions (Rabhi, Yu et al. 2006; Kim and Doh 2009) of software systems (e.g., standards for business process modelling often used are Petri-Net, UML 2.0 activity diagrams and BPMN). However, there is a wide acceptance of business process representation for modelling service-oriented systems (Linthicum 2003; Zhang and Yang 2004; Chen, Li et al. 2005; Jamshidi, Sharifi et al. 2008) to describe behavioural descriptions. It seems that the adoption of behavioural descriptions in modelling is not only to depict business requirements but also to assist with bridging the gap between business models and the service implementation. However, the model languages currently available are not yet capable enough to provide a complete representation for modelling business functions and requirements into suitable models to facilitate service implementation for service-oriented systems.

Design Output criterion: Service identification approaches typically intend to identify candidate services at the end of the modelling phase.

However, the final context and details of the identified services are essential for the efficiency and completeness of any proposed design approaches. The detailed outputs of the different approaches vary considerably, i.e., approaches that result in a formal service specifications (Arsanjani and Allam 2006; Rabhi, Yu et al. 2006; Dwivedi and Kulkarni 2008) are more detailed than those that simply list potential candidate services (Rolland and CentreKaabi 2007; Galster and Bucherer 2008; Kim and Doh 2009; Shirazi, Fareghzadeh et al. 2009) or just provide an explanation of the challenges and guidelines (Arsanjani 2004; Van Nuffel 2007; Boerner and Goeken 2009). The outputs of these design approaches are affected by the techniques used more than any other defined criteria; even a similar type of input might not result in a similar type of outputs. For example, approaches that start the process of service identification with a business process can generate results in several different outputs: a service profile, service implementation and a list of candidate service respectively (Arsanjani and Allam 2006; Shirazi, Fareghzadeh et al. 2009). The strategy adopted also affects the output criterion, e.g., a bottom-up strategy eventually results in web services (WS) (Zhang and Yang 2004; Chen, Li et al. 2005; Jianzhi, Zhuopeng et al. 2005; Aversano, Cerulo et al. 2008). In contrast, approaches that use a meet-in-the-middle strategy advocate service specification and models (Arsanjani and Allam 2006; Rabhi, Yu et al. 2006; Klose, Knackstedt et al. 2007; Arsanjani, Ghosh et al. 2008). In case of adopting the meet-in-the-middle strategy, the feasibility of outputs of this strategy needs to be assessed.

Quality of Service (QoS) criterion: quality aspects such as flexibility and reusability are important factors that support the use of SOA in preference to other development styles. In fact, it is not always possible to meet the desired quality aspects for SOA projects because there are inevitable trade-offs in any implementation. However, specifying quality aspects that are essential to meet for such system precisely helps to achieve SOA benefits. Furthermore, quality attributes should be considered and specified at an early stage of the modelling process. There is a wide variation in meeting the desired software quality attributes in the published literature. There are existing approaches that do not cover the quality aspects (Arsanjani and Allam 2006; Dwivedi and Kulkarni 2008; Kim, Kim et al. 2008; Ma, Zhou et al. 2009; Shirazi, Fareghzadeh et al. 2009) and others that explicitly investigate external architectural quality attributes (e.g., performance, flexibility, and

interoperability) (Wang, HU. et al. 2007; DongSu, Chee-yang et al. 2008; Kim and Doh 2009). Others focus on one particular attribute of QoS, e.g., DongSu et al. (DongSu, Chee-yang et al. 2008) attempt to depict the level of reusability using a range of semantic distance measurements within the service identification process. Wang et al. (Wang, HU. et al. 2007) stress the impact of performance in legacy systems integration with SOA in data translation and payload transportation and suggest possible design criteria to be considered. Loose coupling and high cohesion as primary characteristics of SOA are recommended without clear directions on how to achieve them (Papazoglou and Van 2006; Dwivedi and Kulkarni 2008; Erradi, Kulkarni et al. 2009). What seems missing in many current approaches is a failure to consider the main SOA quality attributes that affect the service identification process. They also fail to define service quality measurements that can be used to determine the quality of candidate services.

Granularity criterion: The granularity of the services implemented is always a design issue, whatever the design approach adopted. Achieving the appropriate level of granularity is very challenging; services are often either coarse-grained or fine-grained. With no explanations as to how service granularity is being assessed, Boerner and Goeken (Boerner and Goeken 2009) add also "middle grained" as an additional granularity type. Furthermore, it is not clear what the best assessment method for assessing the service granularity should be. Classifying various types for service using a hierarchical architecture is one mechanism to assess candidate services individually (Dwivedi and Kulkarni 2008) (e.g., the granularity of business services is coarser than that in infrastructure services because business services reside at higher level of enterprise architecture layers). A granularity metrics tool is being used to quantify service granularity factors to decide appropriate service implementation (Bell 2008). We found that the granularity for defined services varies considerably from one approach to another, even though different approaches have used the same delivery strategy. For example, approaches that use a top-down strategy, but the proposed services have very different granularity levels (varying from coarse-grained to multiple levels of granularity), which demonstrates that multiple criteria affect granularity decisions (Nayak, Nigam et al. 2006; Dwivedi and Kulkarni 2008; Galster and Bucherer 2008). The underlying service identification process in SOA specifically depends on defining the "right" services with an appropriate level of

granularity. A considerable amount of literature has proposed methodologies for identification of such services with the appropriate granularity (Erradi, Anand et al. 2006; Papazoglou and Van 2006; Kim, Kim et al. 2008; Kulkarni and Dwivedi 2008; Zhang, Zhou et al. 2008). Although these approaches have all used different techniques, none of them has achieved a perfect design, agreeing instead on the difficulty of delivering a set of services with appropriate granularity. Furthermore, in service design, the impact of granularity on quality of service (QoS) aspects must also be considered. The candidate services with appropriate level of granularity that are identified should not interfere with the potential benefits of SOA such as flexibility, reusability, and functionality.

In conclusion, although a lot of research has been conducted in service modelling in particular in the service identification, the real design challenges of the service identification phase such as granularity and the abstraction gap between the business models and service implementations have not been solved. The proposed criteria are used to analyse current literature and to address the research gap in the service identification problem. Tables 3-(1, 2, 3, 4, and 5) provide an analysed summary of current approaches using the criteria above.

Table 3-1 Comparison of Service Identification Approaches

CRITERIA	(Galster and Bucherer 2008)	(Kim, Kim et al. 2008)	(Rolland and CentreKaabi 2007)	(Nayak, Nigam et al. 2006)	(Arsanjani and Allam 2006)
Delivery strategy	Top-down	Top-down	Top-down	Top-down	Meet-in-the-middle
Technique	Goal-driven using a graph-based method	Goal-driven using a goal-scenario modelling	Goal-driven using a map-based modelling	Goal-driven	Business process decomposition
Lifecycle Converge	Modelling	Analysis	Modelling and Discovery	Modelling	Modelling
Service Types	Business services	Business services	Business services	Business services	Business Services
Modelling Input	Requirements and goals	Requirements and goals	Requirements and goals	Service agreement	Business processes
Modelling Output	Service capabilities	Service Profile	Composite services	A service model	Service implementation
Quality Aspects	none	none	none	none	none
Granularity	Coarse-grained	Coarse-grained	Coarse-grained	Multiple granularity	Coarse-grained

Table 3-2 Comparison of Service Identification Approaches

CRITERIA	(Boerner and Goeken 2009)	(Shirazi, Fareghzadeh et al. 2009)	(Kim and Doh 2009)	(Dwivedi and Kulkarni 2008)	(Papazoglou and Van 2006)
Delivery strategy	Top-down	Meet-in-the-middle	Top-down	Top-down	Meet-in-the-middle and bottom up
Technique	Business process decomposition	Business functions, goals	clustering using cost metric	An Algorithm	RUP, CBD, BPM
Lifecycle Converge	Modelling	Analysis	Analysis	Modelling	Full SOA cycle
Service Types	Business services	Business and IT services	Business and IT services	Business and IT services	Business and IT services
Modelling Input	Business processes	Business processes	UML activity diagram	Business processes	Business processes
Modelling Output	Guidelines	A list of services	A list of services	Service profile	Service profile
Quality Aspects	none	none	Coupling and cohesion	none	Coupling and cohesion
Granularity	Coarse-grained	Coarse-grained	Multiple granularity	Multiple granularity	Coarse-grained

Table 3-3 Comparison of Service Identification Approaches

CRITERIA	(Jianzhi, Zhuopeng et al. 2005)	(Erradi, Anand et al. 2006)	(Erradi, Kulkarni et al. 2009)	(Arsanjani 2004)	(Arsanjani, Ghosh et al. 2008)
Delivery strategy	Bottom-up	Meet-in-the-middle and bottom-up	Meet-in-the-middle	Meet-in-the-middle	Meet-in-the-middle with focus on top down
Technique	Component-Functional decomposition	Process decomposition and analysis	Domain decomposition	Goal-service modelling	Goal-modelling and process decomposition
Lifecycle Converge	Analysis and Development	Modelling and Development	Design	Modelling	Modelling
Service Types	IT services	Business and IT services	Business and IT services	Business and IT services	Business services
Modelling Input	Legacy code	Requirements, goals,	Business process and existing assets	Business process and existing assets	Business domain and processes
Modelling Output	Service interface (WS)	Technology architecture	Service architecture and guidelines	Guidelines	Service architecture and guidelines
Quality Aspects	none	none	Coupling and cohesion	none	none
Granularity	none	Multiple granularity	Coarse-grained	none	Coarse-grained

Table 3-4 Comparison of Service Identification Approaches

CRITERIA	(Yousef, Odeh et al. 2009)	(DongSu, Chee-yang et al. 2008)	(Zhang and Yang 2004)	(Chen, Li et al. 2005)	(Zou and Kontogiannis 2001)
Delivery strategy	Top-down	Top-down	Bottom-up	Bottom-up	Bottom-up
Technique	Ontology driven / process decompositio n	Ontology-driven/ business decomposition	business functions and existing assets decomposition	Feature analysis	Component decomposition using Clustering techniques
Lifecycle Converge	Modelling	Modelling	Modelling	Modelling	Modelling and development
Service Types	Business and IT services	Business services	Not clear	Not clear	Not clear
Modelling Input	Business process	Service features model	Legacy code	Legacy code	Legacy code
Modelling Output	Service model	Service profile	Service interface (WS)	Service interface (WS	Service interface (WS)
Quality Aspects	NFR	Reusability	Coupling	none	none
Granularity	none	Multiple granularity	Coarse-grained	Coarse-grained	Coarse-grained

Table 3-5 Comparison of Service Identification Approaches

CRITERIA	(Aversano, Cerulo et al. 2008)	(Wang, HU. et al. 2007)	(Klose, Knackstedt et al. 2007)	(Van Nuffel 2007)	(Stewart and Chakraborty 2010)
Delivery strategy	Bottom-up	Bottom-up	Meet-in-the-middle	Top-down	Top-down
Technique	Feature analysis using information retrieval technique	Goal and requirements driven within a transformation method	Clustering analysis using a profound prioritization	Analysis of business requireme nts	value chain and prioritization analysis technique
Lifecycle Converge	Modelling	Modelling	Modelling	Analysis	Modelling
Service Types	Not clear	IT services	Business and process services	Business service	Business and IT services
Modelling Input	Legacy code	Proprietary data	Business processes	Business processes	Business strategies
Modelling Output	Service interface (WS)	SOPA messages	Service profile	Guidelines	A list of services
Quality Aspects	none	Performance	none	none	none
Granularity	Coarse-grained	Coarse-grained	Coarse-grained	Coarse-grained	Coarse-grained

3.4 Summary

In this chapter, we have discussed the different methodologies available for the identification of suitable services during the modelling phase of the SOA development life cycle. We have shown that current methodologies suffer from key limitations, such as a gap between business model and service design and do not consider internal quality aspects that affect the overall quality of service (QoS). These limitations contribute to the failure of the current approaches to identify the "optimum" services, in terms of when services should be coarse grained or fine grained. Although the approaches investigated usually conclude with several service design principles, they do not provide well-defined and effective steps to achieve these principles. From evaluating the relevant design criteria, we can see that a meet-in-the-middle strategy using a business process decomposition technique leads to a detailed service specification which assists considerably in the construction of better candidate services. In addition, service granularity is a key architectural attribute of the service design that will inevitably affect important external architecture attributes of quality of service (QoS) such as reusability, maintainability, performance and flexibility. Indeed, establishing appropriate measurements for service quality is still not present in almost all current approaches. However, these approaches have nevertheless agreed on the complexity of considering all applicable factors to fulfil both the business and the technical aspects (Papazoglou, Traverso et al. 2007).

Against this background, Chapter 4 presents a potential architectural design using the choreography concept and model transformations that can be used to bridge the abstraction gap between business process models and service interface designs. The Chapter also explains the underlying meta-models for source and target models used in the model transformation development which can be used to generate service interface designs automatically.

Chapter 4 Choreography and Model Transformation Design

Having introduced the existing methodologies for service identification and the importance of achieving service quality in Chapter 3, we now present the first part of our framework design for optimum service identification. This Chapter develops a theoretical base of using the choreography concept to bridge the abstraction gap between the business process model and service interface design. Based on the choreography concept, the underlying meta-models used for the model transformation are constructed.

In section 4.1, we formalise the choreography concepts between the business process model and service interface design. This is followed by a discussion of the choreography concept in section 4.2. In section 4.3, we explain the architectural analogy between business process modelling and the choreography concept, and propose an extension to the BPMN 2.0 standard. In section 4.4, we describe the architectural analogy between service choreographies and service implementation and describe the semantics of service choreographies WS-CDL. In section 4.5, general choreography requirements are introduced. In section 4.6, we cover the semantic of service interface in WSDL used during the transformation model. The framework design is summarised in section 4.7.

4.1 Introduction

In the service-oriented computing environment, the concept of choreography appears at two different levels of the SOA development lifecycle: service

modelling and service composition. Firstly, to explain the concept of choreography in service modelling, note that the developers start with a model that is often expressed as collaborative business processes that will eventually be implemented as a service-oriented system. These business processes must work collaboratively in a number of complex interactions to achieve the required business goals. The "Business process choreography" describes and formalises these interactions between the business processes (participants). In business process modelling, a choreography model describes an observable behaviour of a participant (e.g., a company) or participant's role (e.g., a buyer or seller) in an interaction.

Secondly, the concept of choreography in service composition refers to the aggregation of services to achieve new functionalities (Rosen, Lublinsky et al. 2008), assuming identified candidate services are appropriate services that meet user business requirements. A peer-to-peer description of the global of observable interactions between aggregated services is called a "Service Choreography." Complex conversations between peer-to-peer services are described with interactions using messages that conform to behavioural specifications.

Fig. 4-1 illustrates the conceptual model of SOA business process choreographies and service choreographies. Business processes (BPs) capture business and user requirements, which are subsequently implemented as candidate services. These business processes describe a flow of internally sequenced activities within control flows to achieve a business goal, i.e., "business process orchestration." In Fig. 4-1 there are four BPs, each of which is a representation of business process orchestration. On the other hand, "business process choreographies," which describe the external behaviour of BPs based on interactions, concentrates on interactions between BPs (as participants) from a global point of view are shown as a green-curved with double-headed arrows. After identifying candidate services, service interactions can be further broken down into concepts: service choreographies and service orchestrations. Service choreographies describe interactions between different services (participants) using exchanged messages, whereas service orchestrations describe the internal actions and interactions from the point of view of a single service (participant).

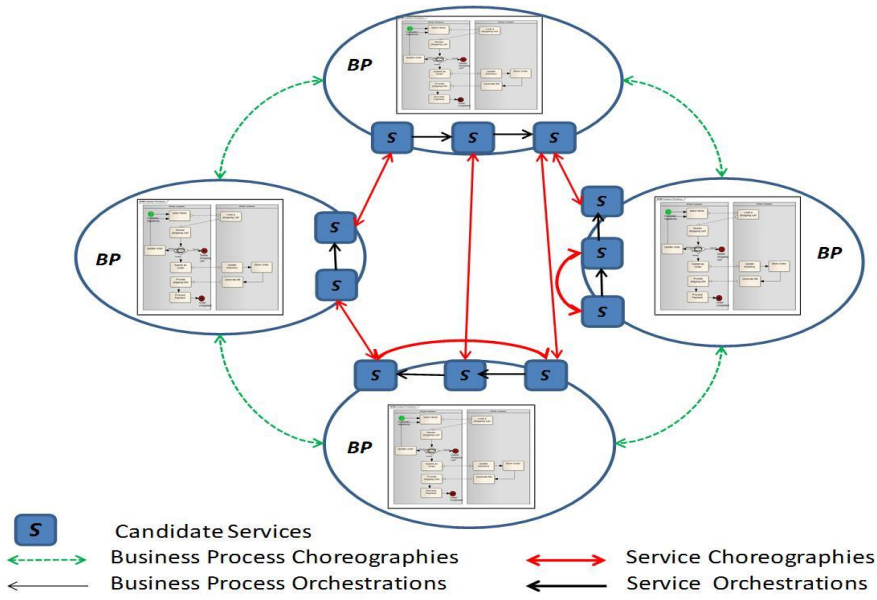


Figure 4-1 The Conceptual Model of SOA Business Process Choreographies and Service Choreographies

There are two main modelling approaches for choreographies: interaction models and interconnected interface models (Decker, Kopp et al. 2008). These approaches are used to describe choreographies at both levels of business processes modelling and service composition. Interaction models describe primary specifications of interactions and are supported by BPMN 2.0 choreography diagrams (OMG 2011), Let's Dance (Zaha, Barros et al. 2006) and the Business Process Schema Specification (BPSS) (Clark, Casanave et al. 2001) choreography languages, e.g., a request-response message is exchanged between two participants. The interconnected interface model describes the internal behaviour of choreography elements and is supported by WS-CDL (W3C 2005), BPEL4Chor (Decker, Kopp et al. 2007) and BPMN 2.0 collaboration diagrams (OMG 2011) as choreography languages, (e.g., a complex interaction between participants that requires a control flow to evaluate outcomes of other interactions to decide next steps). Decker et al. (Decker, Kopp et al. 2008) also consider implementation-independent and specific levels besides the two main paradigms of choreography modelling to distinguish between choreography languages. However, there is still debate about clearly distinguishing between these two approaches and whether they overlap in certain circumstances (Kopp, Leymann et al. 2010).

Both choreography modelling approaches can be supported by one choreography language, such as BPMN 2.0 (i.e., it provides representation as a

collaboration and also a choreography diagram) and WS-CDL (Kopp and Leymann 2009). To evaluate the suitability of a modelling language to model an efficient service composition approach, a number of service interaction patterns are proposed in (Alistair, Dumas et al. 2005); these patterns are derived from the existing literature, relative standard activities (e.g., BPEL and WS-CDL), and "use case" scenarios. According to these patterns, Decker et al. establish key requirements of service choreographies that can be used to evaluate choreography languages (Decker, Kopp et al. 2009). After applying the patterns suggested in (Decker, Kopp et al. 2009) against WS-CDL and in reference (Kopp, Leymann et al. 2011) against BPMN 2.0 collaboration and choreography diagrams, the results suggest that collaboration and choreography diagrams of BPMN 2.0 and WS-CDL as choreography languages fulfil similar requirements. As a result, transformation between these different choreography languages appears to be feasible. This feasibility motivates us to draw a theoretical grounding for using the choreography concept to fill the abstraction gap between business process modelling and service implementations.

Choreography languages have the common goal of describing interactions between participants. Thus, they depend on definitions of the two underlying elements, interactions and participants. Interactions can be represented in a set of patterns that are defined from classic scenarios, such as service patterns (Alistair, Dumas et al. 2005). The way the choreography described is semantic-dependent of the selected choreography languages, i.e., collaborating parties "participants" perform interactions; there are different viewpoints for participants. For example, "Participant" element in WS-CDL includes different types, such as "Role Type." Although the modelling of business processes is an isolated task from service implementation, business processes will eventually the implemented as services.

4.1.1 Service Meta-model

To assist the bridge of the abstraction gap between the definitions of business processes and the description of service interface, a service meta-model was proposed (Fig. 4-2). The service meta-model represents the relationship between BP characteristics and different service types. The model provides a comprehensive understanding of two major concepts: BP modelling and service modelling. Each BP consists of one or more activities. A BP may also be

composed of other BPs (or activities). Each activity either has one or more atomic activities, or is a compound activity that can be broken down to one or more tasks. A compound activity includes an atomic activity that is described by several operations. One or more activities belong to a role, which could be a person or organization. One or more activities use one or more data entities, which could be a transitional data entity or a master data entity. On the other hand, a service includes one or more operations, which will be implemented as either business logic or a as CRUD function (Create, Read, Update, and Delete). In the context of SOA, business logic and CRUD functions can be defined as services with specific types that reside in particular architectural layers (section 2.1.1). Furthermore, CRUD operations can process data entities of BPs as transactional or master data, each of which would have different level of granularity.

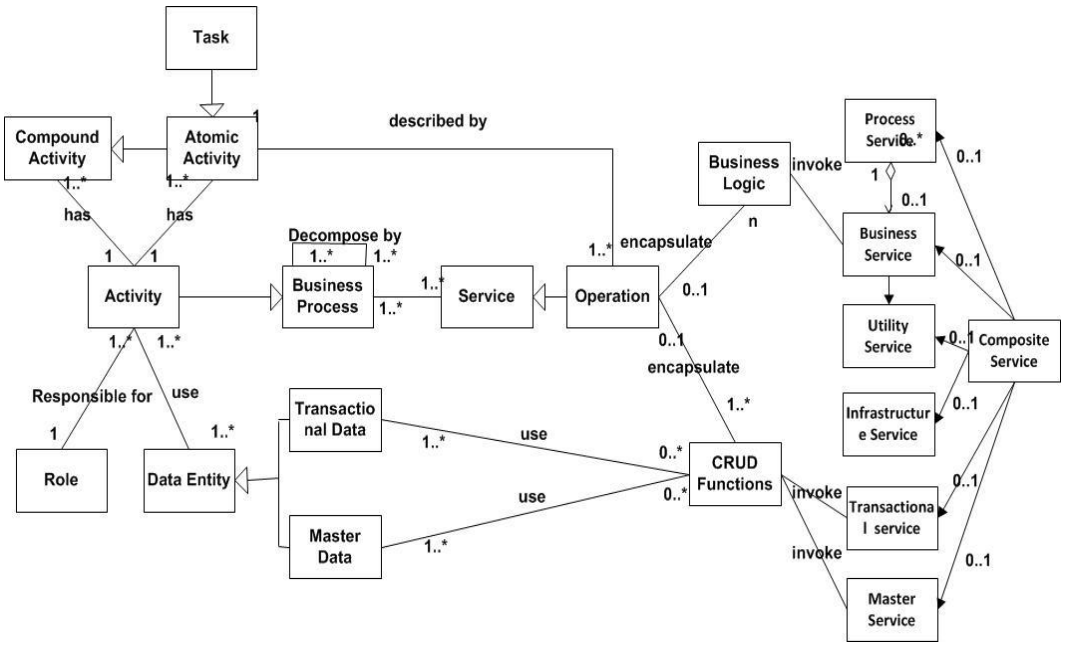


Figure 4-2 The Service Meta-model View

Two essential issues must be considered to identify the correct services for a process-oriented system:

- The abstraction gap between the BP model and service implementation causes separation between the way the business model is described and the way services are implemented. In this context, we can bridge this gap using the choreography concept at two different levels: the BP modelling level and the service modelling level. An explicit representation of these two levels is required to assist with the

implementations of the business process and service choreographies transformations. (Note: this issue will be discussed in this chapter.)

- The service quality attributes (e.g., interoperability, flexibility and agility) are important principles in service-oriented systems. Considering and quantifying these service quality attributes early in the design phase will assist with implementing the optimal set of services in the service domain. The term "quality of service" (QoS) is used here to refer to the internal service quality attributes applied to web services. (Note: this issue will be discussed in chapter 5.)

4.2 Why Choreography?

The majority of research in service composition in particular choreography languages has focused on designing and evaluating semantic and syntax issues. Here, we focus on choreography at two different levels of abstraction: the BP model and service choreographies. That is, we use the choreography concept not only to bridge the abstraction gap between a business model and a service interface, but also as a mediator to implement service interfaces, e.g., a skeleton through which web services or orchestration can be generated. The description of choreographies can be also considered as an initial basis for implementing orchestrations (Decker, Kopp et al. 2008; Hwang, Liao et al. 2010; Kamari and Khayyambashi 2010). However, this view is implicitly supported by a number of studies (Alistair, Dumas et al. 2005), i.e., BPMN 2.0 specifications isolate the definitions of service interface from other choreography modelling conformances. At the service choreography level, achieving interoperability for services can be ensured through choreography by the conforming behaviour of multiple participants (services). Furthermore, it enables validation of services statistically and during run-time in accordance with the description of choreographies in the WS-CDL code. Although, WS-CDL and pi-calculus share a number of elements and pi-calculus can be used to validate WS-CDL code (Decker, Overdick et al. 2006), the WS-CDL must be based on formal language principles to enable proper validations for choreographies (Alistair, Dumas et al. 2005).

The nature of being stateless presents an interesting analogy between BP choreographies and a service interface (WS), both are always in favour of being stateless (Mendling and Hafner 2008). When service requesters invoke services,

the state is persevered, i.e., services do not differentiate between service requestors (clients). While the control of choreography is decentralized and exchanged messages are accomplished thoroughly in multi-part collaborations, the service interface specifies an input and output message for every operation. Therefore, we can theoretically say that using the choreography concept is essential for facilitating the service interface that is driven from a BP model.

4.3 Business Model versus Choreography

This chapter revolves around two key concepts: BP choreography and service choreography. The objective of this section is to define these concepts and their relationships as well as their meta-model. BP modelling languages, such as BPMN, can be used to depict choreographies graphically by linking BPs via message flows (Decker, Kopp et al. 2009). The specifications of these choreographies will describe the behaviour of participants (e.g., business partners). Support of choreography concepts in BP modelling was somewhat limited until BPMN 2.0 emerged (OMG 2011). Support has developed from a simple depiction of basic interactions between participants using BPs and message flows in BPMN 1.x (OMG 2008; OMG 2009) to rich semantics of choreography and collaboration diagrams in BPMN 2.0 (OMG 2011). BPMN 2.0 supports interaction models and introduces choreography diagrams that define a flowchart as sequenced activities of interactions between participants based on message exchanges (Kopp, Leymann et al. 2011). Where choreography is an extended type of collaboration (OMG 2011), collaboration diagrams define interactions between different participants (e.g., Pools and Processes elements), which ultimately support the interconnection of interface models (see section 2.3.1). Unlike the current BPMN 2.0, in order to cover both interaction models and interconnected interface models, we consider choreography and collaboration diagrams that are include all explanations of choreographies in BP modelling.

4.3.1 Preliminary: BPMN Choreographies and BPs Modelling

BP modelling choreographies revolve around key BPMN 2.0 artefacts: collaboration diagrams, choreography diagram, participants, message flows, and pools. The collaboration diagram is the core diagram that includes specifications for all interaction patterns between all participants in one or

more choreography diagrams. In general, the choreography diagram defines an interaction between two participants using sequences of message flows. The participant element represents a specific logic or physical entity involved in an interaction. The message flow element connects different participants and defines the transferred data in an interaction. The pool element presents a participant in an interaction that is represented in a collaboration diagram. Fig. 4-3 shows part of a collaboration diagram as a comprehensive diagram integrating collaboration and choreography definitions according to BPMN 2.0 specifications.

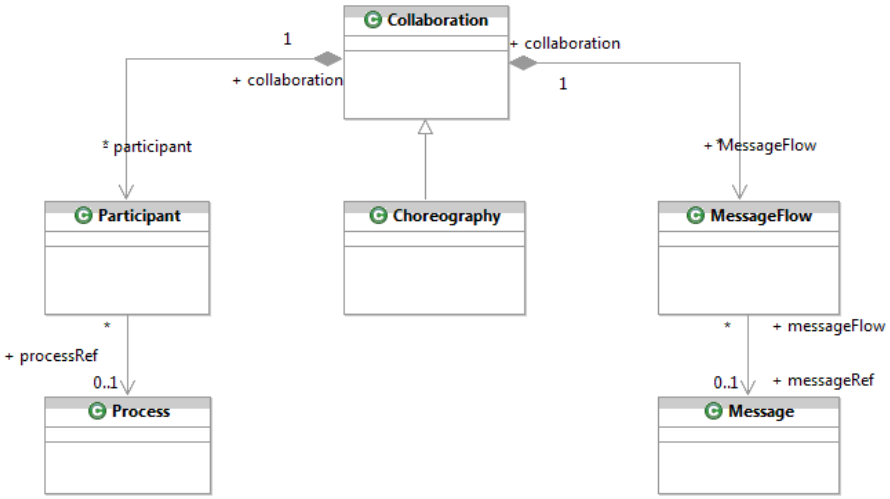


Figure 4-3 BPMN Meta-model

Although BPMN 2.0 has significantly emerged with new capabilities in choreography semantics, processing models, and graphical data, it still lacks important aspects for modelling choreographies, e.g., limited modularity and decomposition capabilities, incompatible control flow dependencies (Decker and Weske 2011), and lacks interchangeability of technical configurations (Kopp, Leymann et al. 2011). For example, for limited interchangeability, interface elements (which have no graphical representation) use associated choreography semantics with the attribute "portType," which must be changed based on the technical aspects of service implementation (Kopp, Leymann et al. 2011). While the aim of the BPMN 2.0 choreography diagram is to implement independent and interchangeable models (Decker, Kopp et al. 2008), we have extended BPMN 2.0 to enhance the interchangeability of choreography semantics from the BP modelling level to the service choreography level. In fact, the BPMN 2.0 standard provides a robust extensibility mechanism that permits users to

extend the standards by creating new attributes and elements. We can classify our extensions into views.

Extending current BPMN 2.0 elements: Current elements are essential for completing the semantics of choreographies and are linked to specific existing constructors. This thesis adopts the extension mechanism available in BPMN 2.0 that allows users to construct new meta-model classes as formal specifications. The BPMN extension mechanism consists of four elements: Extension, ExtensionDefinition, ExtensionAttributesDefinition, and ExtensionAttributesValue. The Extension element connects the new ExtensionDefinition element with the main BPMN model definition through the Definition element. The ExtensionDefinition element defines and groups the extension attributes, while the ExtensionAttributesDefinition element contains newly defined attributes. Finally, ExtensionAttributesValue holds the values of the new attributes. The Message element in BPMN 2.0 specifications is created mainly to show a graphical representation. To define the direction of exchanged messages at the BP modelling level and to enable the correct tracing of exchange messages at the service choreography level, we added a new enumerated class construct that presents three enumeration expressions (Request, Response, and Request-Response). These enumerated expressions correspond to the types of actions associated with exchanged messages. The association relationship between the message flow element and the message element must be changed to one-to-many because a message flow element might have more than one message depending on the action type. For example, a message for an action type "request'" will have one message, whereas an action type "'request-respond" has two messages.

Fig. 4-4 shows the new extension of message types within the BPMN 2.0 meta-model using the available extension mechanism. Three new elements, "MessageTypesDefintion," "MessageTypesAttributes-Definition," and "MessageTypesAttributesValue" extend the existing "Message" elements. The "MessageTypesDefintion" element defines new types of message elements that group definitions of the new attribute, the "actionID" of the enumerated class of data exchanged methods (e.g., Request, Response, Request-Response) using a composite relationship. The "MessageTypesAttributes-Definition" element contains the new attribute definitions, such as the attribute "attributeKind" that refers to the XML schema type (i.e., complex, simple, and driven). The

"MessageTypesAttributesValue" contains values and types that correspond to extended attributes. These new elements are linked to the "BaseElement" element using a new composite relationship to provide values and model associations.

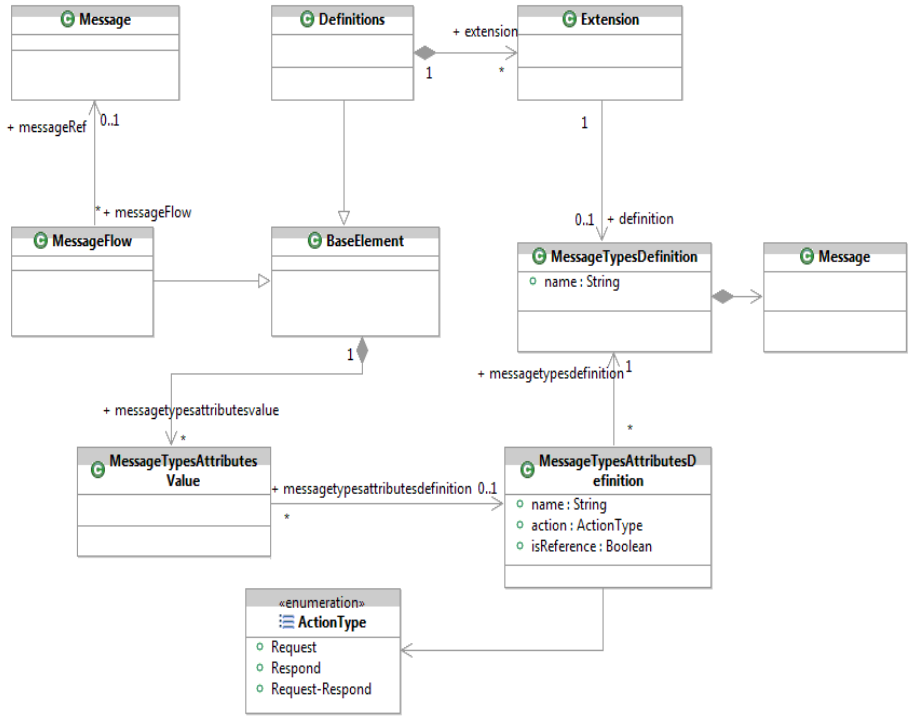


Figure 4-4 Message Types Extension Meta-model Class Diagram

New relationships and attributes for current BPMN 2.0 elements: New relationships are required for connecting new elements to the existing BPMN 2.0 elements. The attributes are defined as specific elements that are required to complete the transformation from the BP modelling level to the service choreography level. This enhances scalability for interchanging BPMN 2.0 choreography representations due to limitations in the semantics of elements and is also required to cope with the representation of early specifications of BPMN 1.x. For example, the element "Pool" in BPMN 1.2 represents participants, whereas the element "MessageFlow" connects boundaries of the element "Pool." There are no connections between the elements "Pool" and "MessageFlow" in the XMI schema interchange, so we construct the composite relationship "PoolMessageFlow" to capture an interaction of a particular message flow involved in the case of missing participant schema. BPMN 2.0 depicts interactions explicitly in a choreography diagram using the "choreography activity" element, which is an abstract

element and represents the point where an interaction occurs in a choreography flow. In the case of the choreography within a collaboration diagram, semantics of participants and message flow elements connect interactions in the choreography within a collaboration diagram. According to BPMN 2.0 standard, an interaction is created when a message flow initiates, thus there is an interaction for every message flow. However, this design might cause redundancy when a message flow occurs twice to initiate a request and then the response to the particular request between interconnected models in one interface . This thesis links the message flow element with the message type element via the attribute "*messageRef*", where the "*actionID*" attribute is associated with the Messageflow element specifying the appropriate data exchanged method. This new relationship allows us to minimize redundancy by creating a message flow element in response to one interaction. Fig. 4-5 shows the new relationships in the context of the BPMN 2.0 meta-model.

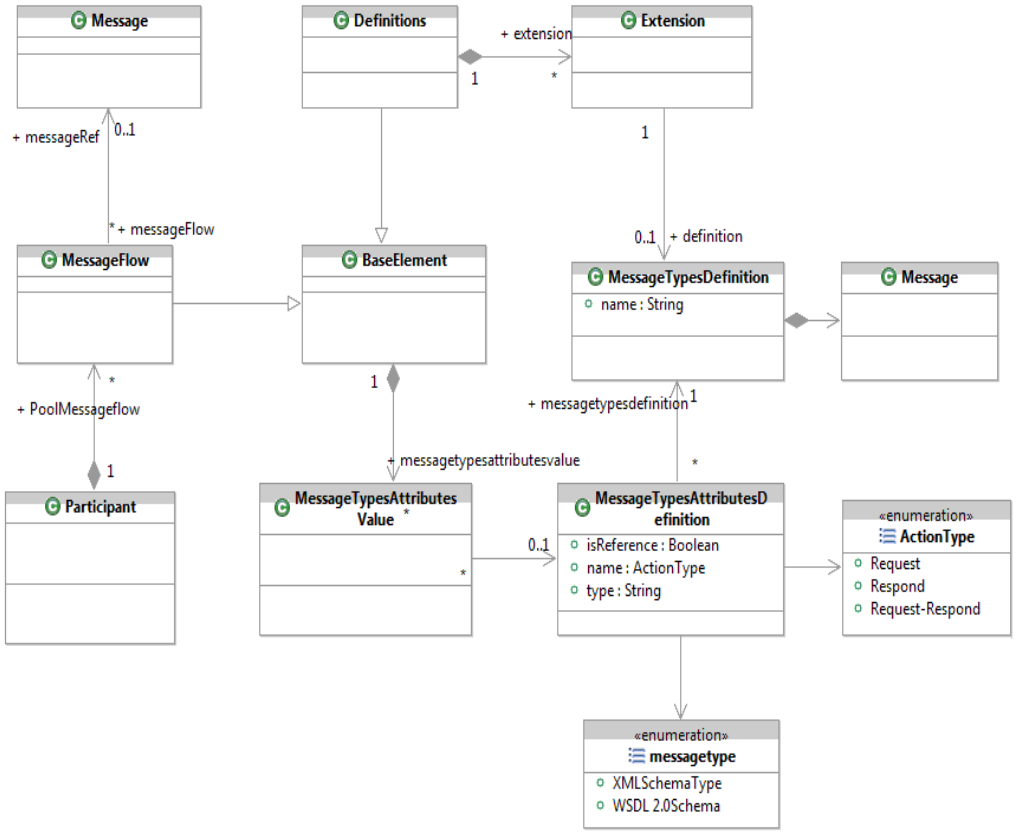


Figure 4-5 New Attributes and Relationships Extension Meta-model Class Diagram

4.4 Choreography versus Service Choreographies

Service composition can be described in the rules of service interactions as orchestration or choreography (Peltz 2003). Orchestrations are descriptions of interactions that occur by one party (web service), including orders of interactions (Rosen, Lublinsky et al. 2008). Choreography is a specification for conversations between different parties (web services) from a global viewpoint (Decker, Kopp et al. 2008). A service is always driven from a BP or function. This means that it is appropriate for the BP definitions and design to be used in the process of service identification. Choreography defines the externally observable behaviours of a BP (Fischer 2005). We have adopted the WS-CDL specification standards for the description of service choreographies.

4.4.1 Preliminary: The Service Choreography Concept and WS-CDL

The WS-CDL code can be conceptually categorized into parts: the package root elements and the choreography definition. We select elements and attributes in WS-CDL that are capable of capturing the semantics of BP models, in particular in BPMN 2.0 models. A brief description of WS-CDL has been provided in section 2.2.4. Alistair et al. (Alistair, Dumas et al. 2005) propose WS-CDL meta-model designed in UML class diagrams that covers the concepts of package and choreography. The focus of our research is not to construct a complete design of WS-CDL meta-model. Rather, it is to demonstrate the ability of WS-CDL to respond to the semantics of BPs. Hence, we illustrate in the detailed WS-CDL meta-model that is implemented in the transformation from the BP modelling level to the service choreography level. Alistair et al. (Alistair, Dumas et al. 2005) define the comprehensive WS-CDL meta-model which we adopted in this thesis. The package elements provide descriptions of participants and captured data within interactions of the observed behaviour. The description of the WS-CDL meta-model is presented in the main package depicted in Fig. 4-6 and the choreography is shown in Fig.4-7.

The main WS-CDL elements (i.e., the un-highlighted elements in Fig. 4-6) of package definitions that are used in the implementation are as follows:

- **InformationType:** This element defines the data types used within defined choreographies and activities, types of exchanged messages, and variables to which schema it uses, i.e., “xsd:name” is used to refer to the XML schema. Further descriptions of the exchanged information can be defined in the

"Variables" element (e.g., capturing the state of a purchase order during the order creation routine of a BP). There is a composition relationship between the element "InformationType" and package definitions. This element is essential as the container of the exchanged data when participants interact. In particular, it includes the definition of a new attribute defined as "attributeKind," in addition to the default attributes of *name* and *element*. The value of "attributeKind" refers to the weight of exhibit data granularity of exchanged messages, which is used for deciding service quality.

- **RoleType:** The RoleType element represents collaborating participants as roles, every role associated with observable behaviour is linked to a specific WSDL interface type. This element will eventually refer to a logical representation of a service; similarly, the representation of a participant (role) in BPs might envisage a process interface.

- **RelationshipType:** This element combines two roles into a specific behaviour or relationship; defined relationships will be further described through an interaction definition within choreography.

- **Choreography:** The Choreography element represents the core of a collaboration, which defines rules that manage the sequence of a message exchange. The Choreography definition can be set locally within a root choreography package definition or globally as a separate top-level element specified in a different choreography package (see Fig. 4-7). It defines a unique name for the choreography within a package.

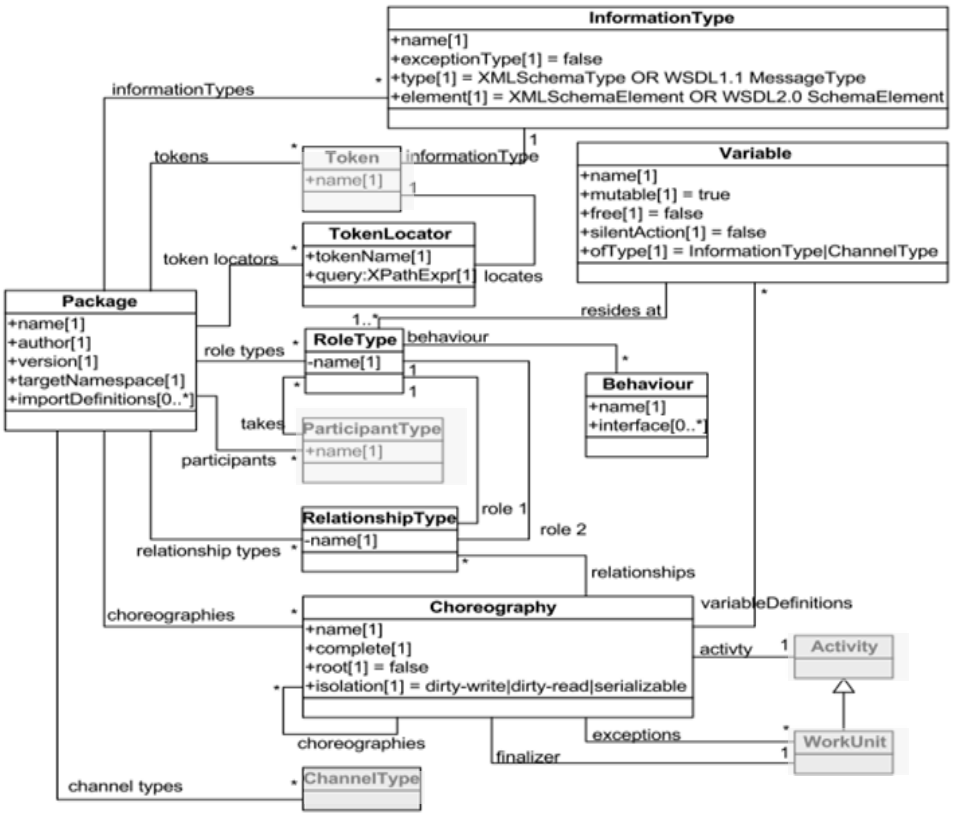


Figure 4-6 The WS-CDL Meta-model (part 1) (Alistair, Dumas et al. 2005)

Fig. 4-7 shows the detailed description of the choreography element (un-highlighted elements in Fig. 4-7) of package definitions that are used in the implementation as follows:

- **Interaction:** The Interaction element is the most important element of choreography languages. It constructs descriptions of the exchanged messages between services. It defines the default attributes of name and operation that specify a unique name for the interaction and its invoked operation. The new attribute "actionType" defines a weight for exhibiting functional granularity of the operation, which is used for deciding service quality. Interaction includes further definitions through linkages to the Exchange and Participate elements.

- **Variable:** The Variable element declares an object's information, such as the state of capturing object and capturing channels. In particular, we use this element to prescribe the definitions of the InformationType element within a specific choreography.

- **Activity:** The Activity element explains the actions carried out within a choreography activity. It is like an abstract class for different activity types of ordering structure, work-unit, and basic activity; each type is covered individually.

- **Sequence:** The Sequence element (ordering structure type) enables sequential definition of the activity notations. It is essential when there is more than one activity notation to control the flow.

- **Parallel and Choice:** The Parallel and Choice elements (ordering structure types) enable concurrent and implicit selection of one or more activity notations within a choreography. According to the WS-CDL specifications, no attributes are defined for these two elements. However, we add a name attribute for readability and keep consistency of the transformation within different models.

- **WorkUnit:** The WorkUnit element prescribes the conditional execution within a choreography, defining a unique attribute name for the element within the choreography element. It checks a conditional statement using the attribute guard and based on the evaluation of the guard condition (i.e., true or false), the next execution is performed. The attribute repeat specifics the repetition of the execution within the WorkUnit element.

- **Exchange:** The Exchange element provides more detailed information about the operation attribute of the Interaction element. It prescribes the definitions of the type of action used via the action attribute and the exchange of messages (i.e., send and receive attributes), which are essential when specifying the granularity of every operation in a service.

- **Participate:** The Participate element defines the sender and receiver roles based on the defined RoleType element and the name of the associated relationship. It shows the source and is responsible for the operations via three attributes, RelationshipType, fromRoleTypeRef, and toRoleTypeRef.

We did not cover every element in the WS-CDL specification because some elements have semantics that are irrelevant for the transformed model within our framework. In this thesis, the WS-CDL meta-model was defined according

to the semantics of the choreography requirements and the requirements in selected application scenarios.

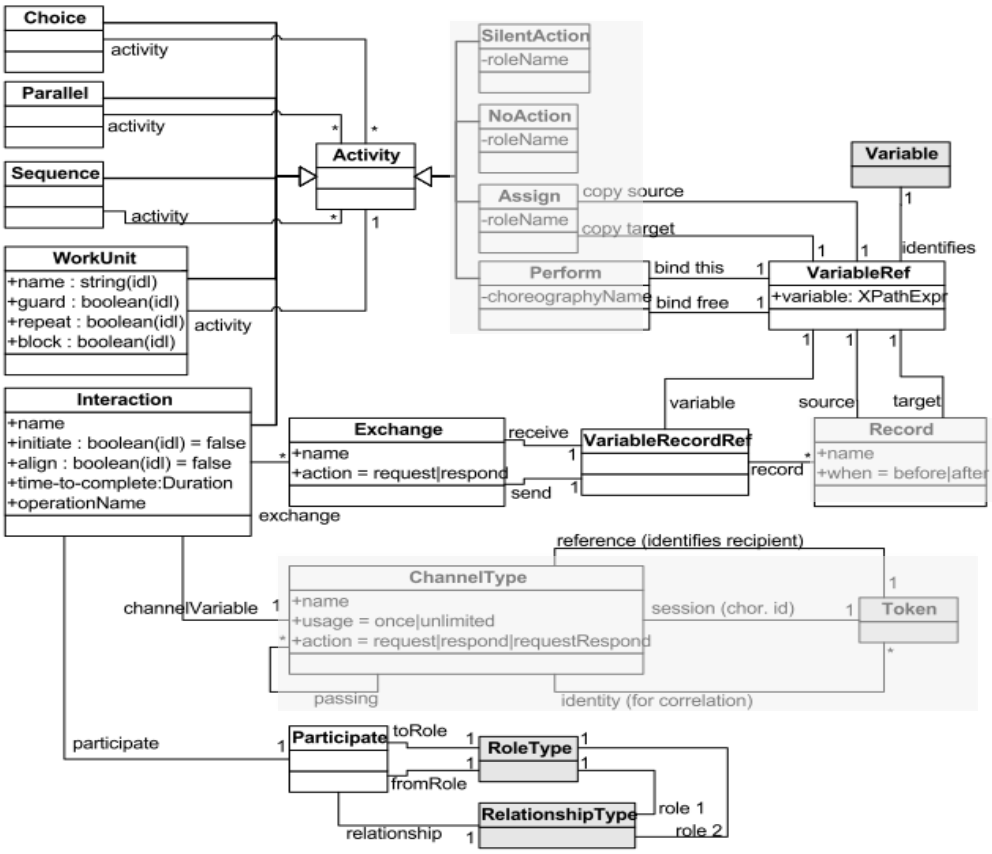


Figure 4-7 The WS-CDL Meta-model (part 2)(Alistair, Dumas et al. 2005)

4.5 Choreography Requirements

To evaluate the capabilities of choreography languages, Decker et al. (Decker, Kopp et al. 2009) provide a set of requirements that can be used to asses both BPMN 2.0 and the WS-CDL language. Their assessment investigated the capabilities of both languages to describe choreographies at their own level of abstractions. In addition, it sought to find similar transformation patterns between the two languages by evaluating them against similar requirements. When we discuss the capabilities of BPMN 2.0, we will consider the interchangeability between choreography and collaboration diagrams, where they support different interactions, paradigm interactions, and interconnections respectively. The requirements of choreography languages are as follows:

- **Multi-lateral interactions (RQ1):** This is the capability to handle the descriptions of more than two participants that interact in more than one interaction. This requirement is fully met in both languages. In BPMN 2.0,

the choreography diagram supports multi-lateral interactions through the definitions of the behaviour of two or more participants in collaboration; the interactions between participants are shown in a number of choreography tasks. In collaboration diagrams, the *Pool* element represents participants and the *Message Flow* element shows interactions. A participant can interact with more than one participant using message flows. In the case of WS-CDL, a choreography description enables a definition of several scenarios of interactions using one or more *RelationshipType* elements.

- **Service (participant) topology (RQ2):** Having a structural vision of how different services (participants) collaborate and the types of services (participants) that exist is an important choreography requirement that is supported fully in BPMN 2.0 but only partially in WS-CDL. In BPMN 2.0, a choreography diagram provides a *choreography activity* element, which supports the definition of the interactions of different participants; each interaction is presented as a *ChoreographyTask* element. The types of services (participants) can be defined using the *PartnerEntity* element. The *ParticipantMultiplicity* element defines the maximum and minimum number of participants. In the BPMN 2.0 collaboration diagram (Figure 2-7 A Collaboration Diagram Example), the *pool* element represents participant types. In WS-CDL, the Roletype elements can be counted, which represents service topology. However, the enumeration relationship between the Roletype element and service participant is not clear because a role-type can be defined for one or more services.

- **Service sets (RQ3):** Supporting several services that are defined with the same type of participant is a requirement that is met fully in BPMN 2.0 and partially in WS-CDL. In BPMN 2.0, there is a graphical sign (three black parallel lines) that indicates that a participant has multiple instances. The WS-CDL specification does not support multiple executions to priori runtime, but it is possible to provide support during design time only.

- **Selection of services and reference passing (RQ4):** All services are made aware of the selection during the design time and runtime. This requirement is partially supported in BPMN 2.0 and WS-CDL. In BPMN 2.0, the *messageRef* attribute is defined for tasks type receivers, which

indicate there is an incoming message. The exchanged messages (data) mechanism between participants makes a service aware of the selection.

- **Message formats (RQ5):** Exchanged messages that are used to communicate between participants must be in the same formats, e.g., XML-based messages. BPMN 2.0 offers the ability to define message formats in the XML scheme using a specific attribute with the message element and thus supports this requirement. The WS-CDL also fully supports this requirement since it uses the standard WSDL message formats.

- **Interchangeability of technical configuration (RQ6):** Using WSDL as the structural interface description with message definitions that influences the choreography language, e.g., changes in the port types or binding should not cause significant changes in the choreography descriptions. Neither BPMN 2.0 nor WS-CDL supports this requirement. The BPMN 2.0 specification states that the structural interface description must be in WSDL. WS-CDL binds to the WSDL configuration, which makes changes in WSDL document that causes changes in WS-CDL.

- **Time constraints (RQ7):** It is important to control the time of exchanged messages, e.g., to allow timeouts to be specified as a type of request-responding message. BPMN 2.0 fully supports this requirement by means of the multiple-event element with the attribute "TimerEventDefinition." The Interaction element in WS-CDL allows the specification of the time taken to complete the interaction.

- **Exception handling (RQ8):** It is possible to halt collaboration of participants under defined constraints. This requirement is met partially in BPMN 2.0 and fully in WS-CDL. Within the interconnection models (collaboration diagrams) of BPMN 2.0, exception handling can disrupt the flow of a process by using intermediate or error events. In a choreography diagram in BPMN 2.0, the exceptions increase based on individual participants, which means other participants remain un-notified. There are various types of exception handlers in the WS-CDL, such as interaction failures, validation errors, and protocol-based exchange failures.

- **Correlation (RQ9):** Different conversations between participants must be uniquely identified by identifiers. This requirement is met fully in BPMN 2.0 and WS-CDL by means of a correlation key element and the token element (respectively).

- **Integration with service orchestration languages (RQ10):** The ability to integrate a standard language for BPs, such as BPEL, is partially supported only in BPMN 2.0; WS-CDL has no such support. However, the support of BPMN 2.0 does not cover all patterns of the orchestration and choreography languages.

Table 4-1 provides a comparison of BPMN 2.0 and WS-CDL. In addition to the results of the comparison of relevant research conducted in (Decker, Kopp et al. 2009; Kopp, Leymann et al. 2011), we found that both BPMN and WS-CDL satisfy most choreography requirements, except RQ6 and RQ10. However, RQ6 is currently difficult to satisfy because of the integration with WSDL. Regarding RQ10, the maturity of BPEL as an orchestration language is still undetermined.

Table 4-1 Assessment of BPMN 2.0 and WS-CDL Support for Choreography Requirements

Items	Requirements	BPMN 2.0	WS-CDL
RQ1	Multi-lateral interactions	+	+
RQ2	Service (participant) topology	+	-/+
RQ3	Service sets	+	-/+
RQ4	Selection of services and reference passing	-/+	-/+
RQ5	Message formats.	+	+
RQ6	Interchangeability of technical configurations	-	-
RQ7	Time constraints	+	+
RQ8	Exception handling	-/+	+
RQ9	Correlation	+	+
RQ10	Integration with service orchestration languages	-/+	-
+ fully supported, -/+ partially supported, - not supported			

4.6 Service Interface in WSDL

WS-CDL specifications include a reference to the service definitions in WSDL code. This reference is the name of the interface which defines messages, operations, binding styles and services. The definitions required to construct WSDL code can be derived from the WS-CDL (see Chapter 6).

WSDL is an XML-based language to describe web services. A web service is the service implementation, an application programming interface (API) invoked over a protocol. We selected WSDL 2.0 standards over the former specifications 1.2 and 1.1 because of the new features, including the interface inheritance feature that results in high reusability and an extensibility capability for Message Exchange Patterns (MEPs). In particular, the extensibility of MEPs is essential for specifications such as WS-CDL and WSDL that use messages exchanged for communication. However, the support of WSDL 2.0 is still limited in regard of tools. A WSDL service interface description document consists of two main components: abstract and concrete. The abstract component defines relevant elements of a service, e.g., definitions of exchanged messages and associated operations. The concrete component defines how and where the service is accessed. Fig. 4-8 shows the incomplete WSDL 2.0 meta-model that includes the main elements of a service interface as follows:

- **Description:** The Description element is a container of the document declarations and WSDL 2.0 elements, such as types, interface, bindings, and services. It defines definitions of the target namespaces that include declarations for semantics of all components.

- **Types:** The Types element defines data types using *Input* and *Output* elements in the meta-model that describes the XML schema definitions of all messages (parameters) accessed by operations defined in the WSDL document. Current XML schema in element types used in WSDL 2.0 defines two data types: *complexType* or *simpleType.* According to XML schema data types (Biron, Permanente et al. 2004), the simple type can be further classified as either a primitive or derived type, each with different constraining facets, such as length and pattern. We applied this classification to layer new levels of data granularity used in message exchanged (for further details see section 5.2.1).

- **Interface:** The Interface element defines a set of performed operations, specifying the messages that are accessed. As a new feature in WSDL 2.0, an interface can be optionally extended and derived from one or more interfaces. Along with this feature, the property attribute defines the behaviour control of the features.

- **Operation:** The Operation element defines actions performed by a service, accessing definitions through a sequence of input and output messages (parameters) used in the operations and the defined types element. We extended the semantics of the operation definition through the *ActionType* element that refers to the purpose of the operation and functional granularity (for further details see section 5.2.2).

- **Binding:** The Binding element defines the underlying protocol, associated operations and message format, e.g., SOAP and HTTP. Every binding links to an interface.

- **Service:** The Service element specifies one or more end points that define the network address where the service can be invoked. A service definition can have one or more interfaces; an interface might have one or more bindings.

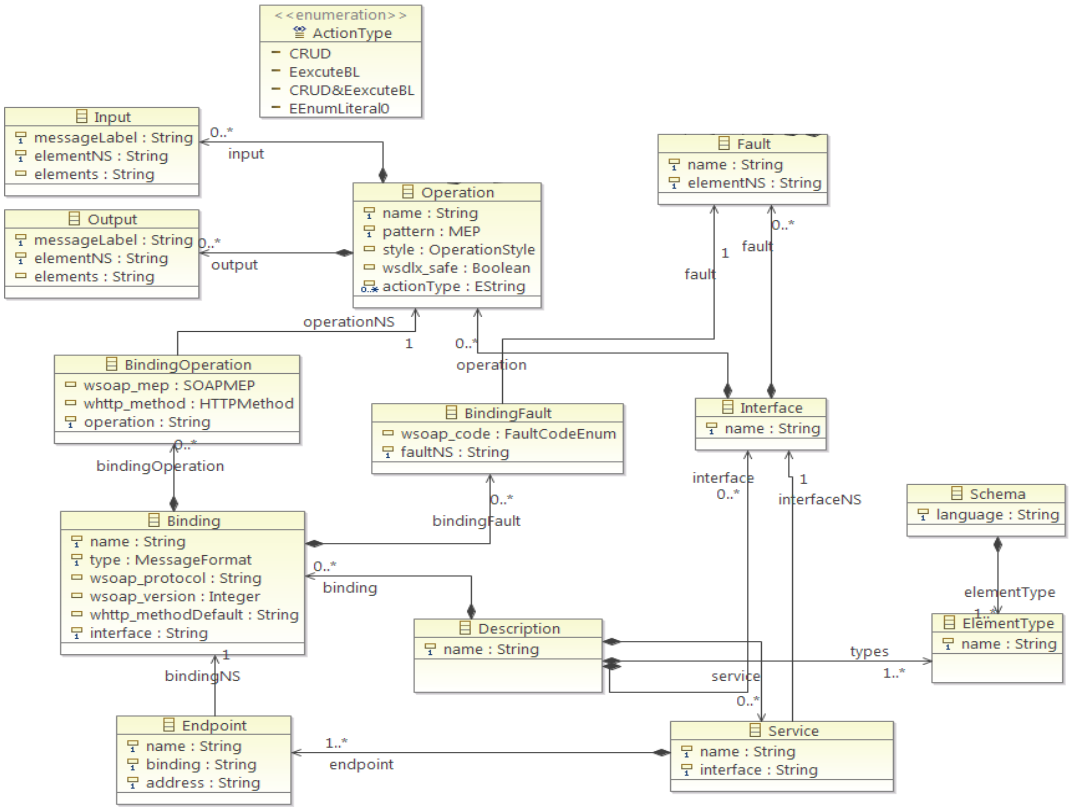


Figure 4-8 WSDL 2.0 Meta-model

4.7 Summary

This Chapter presented the choreography concept within the context of business process modelling and service choreography, which form the basis of the implementations of the transformation model in our framework. The discussion covered the semantics of business process choreography in BPMN 2.0, service choreographies in WS-CDL and service interface design in WSDL.

We concentrated on bridging the abstraction gap between a business processes model and the implementations as set of services. We formalised new extensions within the BPMN 2.0 choreography model for underlying BPMN 2.0 meta-models. The proposed BPMN 2.0 extensions provide missing properties that are mandatory for facilitating the transformation from a business process choreography model to the service choreography semantics in WS-CDL. It will be also used to define the quality attributes. The extension consists of new relationships between current elements of the choreography model in BPMN 2.0 and new elements such as “attributeKind” and “actionID”.

We represented the WS-CDL meta-model, focusing on the main elements that enhance interchangeability between the BP choreography model in BPMN

2.0 and service interface in WSDL 2.0. A list of requirements for choreography languages in the literature shows the suitability of using the semantics of the choreography model in BPMN 2.0 and WS-CDL for choreography modelling. To a great extent, they cover similar choreography patterns as described in the choreography requirements, either fully or partially. We represented an incomplete meta-model of the service interface in WSDL 2.0 that will be used in the transformation of the implementation between WS-CDL and the service interface in WSDL 2.0.

In Chapter 5, we will present the service quality model, which is the second essential aspect of service identification process. The Chapter also discusses the software metrics that can be used to evaluate service interface designs.

Chapter 5 Service Quality Model

Chapter 4 introduced the concept of choreography to bridge the abstraction gap between the business process models and the implementation of the service interface design. Chapter 4 presented the first part of the framework design. This Chapter explains the second part of the framework design proposing a service quality model that can assist in selecting the "optimum" services. The selection will be based on computations of service metrics of service granularity and service quality attributes of complexity, cohesion and coupling.

In section 5.1, we explain the underlying theory of the service quality model. This is followed, in section 5.2, by a description of the basic metrics of service data granularity and functional service granularity that comprise the granularity metrics of the average service operation, and which provide a measurement for service granularity. In section 5.3, we introduce three metrics for the architectural quality attributes of complexity, cohesion and coupling that are based on the service quality model and show the impact of service granularity on other architectural attributes. In section 5.4, we conduct a theoretical validation for the proposed metrics against mathematical properties. This chapter concludes with a summary of the service quality model in section 5.5.

5.1 Service Granularity Quality Model

It is essential to identify the appropriate level of granularity of services in the early phases of SOA quality design as well as the identification of service quality attributes for a set of services. While a key objective of software engineering is the enhancement of software quality, the focus of the SOA

quality metrics that currently exist is on the broad measurements of external structural software service attributes (e.g., complexity, reusability, and performance). They neglect the impact from internal structural software attributes, in particular from service granularity. Although several researchers have attempted to develop an assessment of SOA quality attributes, very few provide specific details in terms of service granularity metrics. Our goal is to analyse the granularity of service operations for service-oriented systems from the perspective of a service provider. We have developed syntactic metrics that are driven by the service code syntax.

"Service granularity" is a measure of the exposed functionality of services. The service granularity of any service-oriented system indirectly affects typical SOA design qualities such as flexibility, reusability, and performance. The granularity of service operations plays a key role in SOA quality attributes (Shim, Choue et al. 2008). This impact can be positive or negative, based on the trade-offs adopted by the service provider. Coarse-grained services are usually advantageous because they improve overall performance, but they do so at the expense of reducing system flexibility. SOA designs a set of services that communicate with each other, each service having a number of specific operations that each contribute to the definition of the functional scope of the service (Erl, Karmarkar et al. 2008).

In order to measure the granularity of a service, we analyse the component service elements, using the definitions of granularity types proposed by Haesen et al. (Haesen, Snoeck et al. 2008). A service granularity quality model is proposed in this thesis with an explanation of how our definitions are driven by levels of granularity in a service-oriented system (see Fig. 5-1). Our quality model has two levels, the service level and the operations level. The service level always favours coarse-grained services. We ignore any service type classifications because the objective is to measure the granularity of operations for that particular service. At the operations level, we consider the purpose of the operation as well as the amount of data exchanged to define the operation and data types respectively. Here, we propose a set of metrics for measuring the internal structural attribute of service granularity in service-oriented systems. We also attempt to measure the impact of service granularity on other internal attributes of complexity, cohesion and coupling.

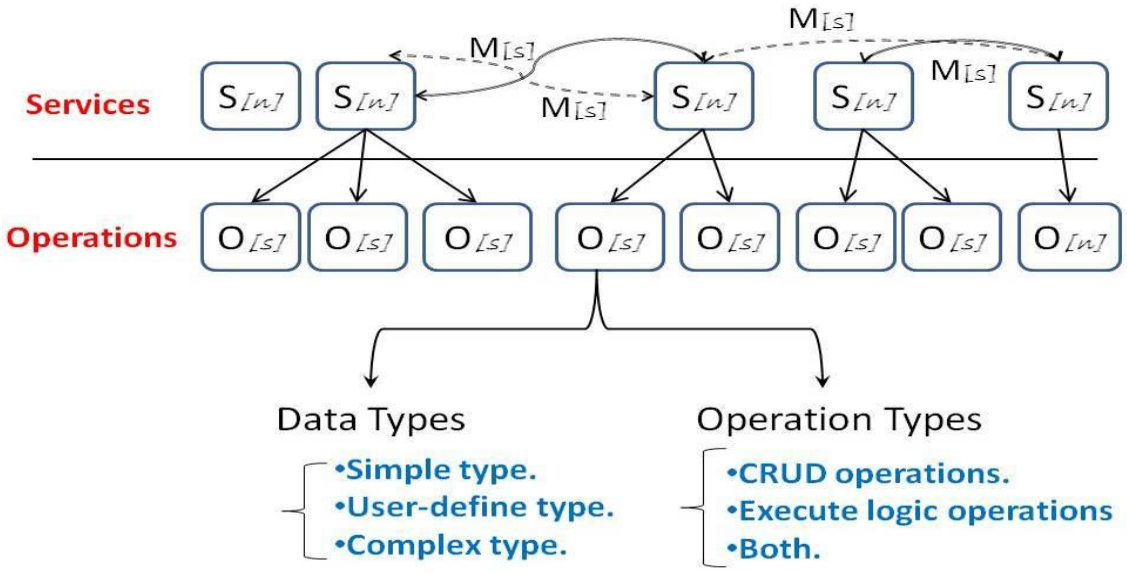


Figure 5-1 The Service Granularity Quality Model

5.2 Basic Metrics of Service Granularity

The metrics proposed have been devised to address the key aspects of a service business functionality and data manipulation. These aspects are considered individually, together with metrics derived from the service interface. The definitions that will be used for the proposed metrics are set out below:

- N –the domain of services.
- $S[n]$ – the set of services in the domain $n \in N$.
- $O[s]$ – the set of service operations in the service $s \in S$.
- $M[s]$ – the set of messages in the service $s \in S$.
- $P[o]$ – the set of parameters in service operations $o \in O$.

5.2.1 Metrics for the Data Granularity Score

The type and size of data elements manipulated by service operations can impact several internal structural software attributes: complexity, coupling, and cohesion; some researchers refer to this effect as "data granularity" (Haesen, Snoeck et al. 2008). In coarse-grained data, such as a structure data type, passing such data types minimizes communication overhead and improves performance. On the other hand, passing fined-grained data elements as individual parameters (i.e., primitive types such as string, integer, long,

decimal, etc.) might require additional work to complete all of the necessary computations. Of course, the use of elements with high-granularity improves overall system flexibility because each data element can be manipulated individually as required. The data granularity adopted as part of the service operations indirectly affects the service qualities. For example, the data size of a customer's record is more coarse-grained than that of a customer's identification element.

Previous research in SOA metrics has considered different ways to evaluate input and output parameters, depending on coarse-grained parameters (Shim, Choue et al. 2008). Dmytro et al. (Rud, Schmietendorf et al. 2006) suggest using the absolute size in bytes to measure the size of elements. This is unsatisfactory because the size of the service refers to self-contained functionality. Type definitions of data elements can be defined as complex types or simple types based on the XML schema for data types as well as user-defined data types (Biron, Permanente et al. 2004). A "complex type" parameter has attributes presented as a data structure or objects. A "simple type" parameter refers to a built-in type as defined in the XML specification, and can be either a primitive type (i.e., one that holds a single value, e.g., a float, string or double) or a derived type (e.g., a token, entity, unsigned long). User-defined data types are defined by individual schema designers. We define three different weights for these three data parameter types based on a comparative scale, these difference weights can be defined for input and output parameters, where:

$$FPW(p_o)\ or\ CPW(p_o) \begin{cases} = 1 \text{ if parameter is a simple type of parameter (primitive type or derived)} \\ = 5 \text{ if parameter is a user – defined data type} \\ = 10 \text{ if parameter is a complex type} \end{cases}$$

The given weights 1, 5 and 10 are alternatively selected; however these weights must have a consistent difference between them. We propose that complex data types should be assigned a heavier weight because they result in additional communication overhead compared to primitive and derived data types. Some user-defined data types might have a heavier weight than simple data types such as a primitive type because they can require additional computation. The data granularity score (DGS) measures the degree to which

an operation uses "excessive" data. The definition of DGS is based on fine-grained and coarse-grained parameters. To measure the data granularity of input and output parameters in an operation of a service, we define the operation data granularity (ODG) metric as follows:

$$ODG(O_s) = \left(\frac{FPW(p_o)}{\sum_{i=1}^{n} FP_i} + \frac{CPW(p_o)}{\sum_{i=1}^{n} CP_i} \right)$$

- FPW is the weight value assigned for an input parameter (FPW > 0).
- CPW is the weight value assigned for an output parameter (CPW > 0).
- FP is a function to sum the total weight of all input parameters of a service.
- CP is a function to sum the total weight of all output parameters of a service.

The valid range of ODG is between zero and unity because the value of the numerator (e.g., $FPW(p_o)\ or\ CPW(p_o)$) is a fraction of the total of denominator $\sum_{i=1}^{n} FP_i$ for each data element. A value close to unity indicates a low granularity (i.e., the data granularity of the operation service is coarse-grained) and a value close to zero indicates fine granularity.

5.2.2 Metrics for the Functionality Granularity Score

The functionality granularity of an operation service refers to the logic encapsulated by an operation or operations within a service. Various operations offer various levels of logic, which can be described as the "capability" of the operation (Hirzalla, Cleland-Huang et al. 2009). The functionality of a service consists of both business logic and CRUD functions. The CRUD functions can be implemented within service areas or separately within specific services. Operation services executing business logic can also be implemented separately or implicitly with CRUD functions. In this context, other researchers have suggested entity-centric business services called "entity services" to support the CRUD function interface and manage business entities (Cohen 2007; Hirzalla, Cleland-Huang et al. 2009). Thus, we define three different weights for the three different types of operations with different levels of granularity, using a comparative scale where:

$$OT(o_s)\begin{cases} = 1 \text{ if a service operation has CRUD operations} \\ = 5 \text{ if a service operation executes business logic} \\ = 10 \text{ if a service operation executes business logic and has CRUD operations} \end{cases}$$

We propose that service operations that execute business logic and CRUD functions have the heaviest weight because they result in additional computation overhead compared to those execute business logic or CRUD. A service that implements only CRUD functions has a lower granularity than that of a service that executes both business logic and CRUD functions (Erl, Karmarkar et al. 2008). We assume that the weight of service operations is based on both the value and scope offered by the service operations. To measure the granularity of the functionality of a service operation, we define the operation function granularity (OFG) metric as follows:

$$OFG(O_s) = \frac{OT(o_s)}{\sum_{i=1}^{n} O_i}$$

- OT is the scale weight value for functionalities in a service operation (OT > 0).
- O is a function that sums the total weights of functionalities for all operations in a service.

5.2.3 Metrics for Service Operations Granularity Score

A service consists of a set of operations that provide the self-contained functionality of the service. In order to estimate an accurate measurement for the service granularity, we begin by measuring the size of an operation service based on the ODG and OFG metrics for each service operation. We then define a metric to measure the total granularity of a service operation. We define the service operation granularity (SOG) as follows:

$$SOG(O_s) = \sum_{i=1}^{n} \big(ODG(i) \times OFG(i)\big)$$

Where ODG and OFG $\neq$ 0, and n is the number of operations in a service (n $\geq$ 1). We can also evaluate the granularity of every service operation individually based on our proposed scale definitions: low/average/high as shown

in Table 5-1. This table shows three arbitrary ranges: $0.00 < value \leq 0.33$, $0.33 < value \leq 0.66$, $0.66 > value$. This table is used to evaluate the data and functional granularity for a specific service operation. Thus, the level of the granularity considers both data and functional granularity together to define an appropriate scale for service operations in a service-oriented system.

Table 5-1 Evaluation of the Granularity Level for a Service Operation

	$0.00 < ODG \leq 0.33$	$0.33 < ODG \leq 0.66$	$0.66 < ODG$
$0.00 < OFG \leq 0.33$	Low	Low	Average
$0.33 < OFG \leq 0.66$	Low	Average	High
$0.66 < OFG$	Average	High	High

To measure the service granularity for all services in a service-based system, an average is calculated based on SOG, where SOG > 0 and NS is the number of services in a domain (NS > 0), we define the Average Service Operation Granularity (ASOG) metric as follows:

$$ASOG = \frac{\sum_{i=1}^{n}(SOG(i))}{NS}$$

- SOG is the value of service granularity of an operation in a service.
- ASOG is the cumulative total for the size of granularity of all services in a service domain.

5.3 The Impact of Service Operation Granularity

Service granularity influences a number of different internal and external structural software attributes (Perepletchikov, Ryan et al. 2005). We analyse the internal structural software attributes of complexity, cohesion and coupling that are influenced by service granularity. These internal attributes will eventually be used to analyse the external software attributes of reusability, flexibility, and portability. To achieve the key features of SOA in a particular domain, we need to derive a balance between several different quality attributes of the service implementation; in other words, we need to establish "trade-offs". The measurement of granularity has been extensively discussed in the context of Object-Oriented (OO) development. Many existing SOA metrics have been derived from former research into both OO and procedural programming

(Perepletchikov, Ryan et al. 2007; Perepletchikov, Ryan et al. 2007). In this section, we will analyse the SOA internal structural quality attributes that are affected directly by service granularity. These attributes are the complexity, cohesion and coupling that are essential for the service quality and need to be considered during the service implementations.

5.3.1 Service Operation Complexity

Service complexity refers to the effort required to maintain and to comprehend the implementation of a service or set of services. Although complexity metrics for service-based systems typically have four dimensions (i.e., data complexity, system complexity, service complexity and process complexity (Zheng and Keung 2010)), metrics derived from the concept of service granularity are our main concern. Complexity levels in SOA are a result of key design decisions that are directly related to service granularity (Fenton and Neil 1999). For example, developing many fine-grained services might increase the complexity of service governance. We define complexity as a dependent variable of the independent variable, service granularity. In other words, any changes in service granularity will impact the overall degree of complexity.

In a composite service, the average number of dependency relationships per atomic service might be considered (Liu and Traore 2007). Here, network cohesion among system nodes that have services, the number of services in composite services, and the count of dependent service pairs are proposed to quantify the complexity of an SOA infrastructure (Rud, Schmietendorf et al. 2006). Simply, the number of operations and messages in a service interface can also be used as indicators for complexity (Sindhgatta, Sengupta et al. 2009). Those metrics are broadly correlated to the size of service operations and complexity; they are essentially adaptations of the classic fan-out complexity metric.

We will focus on the aspects of functional complexity that are directly related to service granularity (SOG). We suggest that an appropriate measure of the effort required to comprehend a service implementation would be a metric based on the exponentiation of SOG as a^n, where the base a = SOG and the exponent n= 2. NS is the number of services in a domain (NS > 0), we define the Average Service Operation Complexity (ASOM) metric as follows:

$$ASOM = \frac{\left(\sum_{i=1}^{n}\left(SOG(i)\right)^2\right)}{NS}$$

5.3.2 Service Operation Cohesion

The degree to which service elements are related to functionality expressed in a service is an important measure that is needed to demonstrate the complexity of service levels and eventually of the overall system. Unlike fine-grained services, coarse-grained services have a significantly higher probability of being cohesive with a larger number of service operations. We define service cohesion as service operations that have similar types of exchanged messages (e.g., complex type) and operations (e.g., CRUD operations). The higher the cohesion, the less maintainability effort that will typically be required during service development (Perepletchikov, Ryan et al. 2007). The service functional cohesion index (SFCI) metric is used to express the commonality of the key message(s) to define the cohesion of the operations of services (Sindhgatta, Sengupta et al. 2009). Our metric considers the size of data and the operation types mentioned previously. However, the size of data is more accurate than particular occurrences of a specific message and this presents a challenge. If the number of operations with a specific type o using a specific size of input/output data (ODG) and operations (OFG) is μ(OFG, ODG) where: $o \in O_{(S)}$, $OFG > 0$, $ODG > 0$, *and* NO_S is the number of service operations in a service $NO_S > 0$ then we define the service operation cohesion (SOC) metric as follows:

$$SOC(s) = \frac{max\left(\mu(OFG, ODG)\right)}{NO_S}$$

The SOC value indicates cohesion of a service operation with a range of zero to unity. If the SOC value is equal to zero, there is no cohesion among service operations in the service, implying high complexity. The closer the value of SOC is to unity, the lower the complexity. We assume that input and output messages have the same weight. Where there is more than one value that corresponds to the maximum function, we do not consider their service operations in NS. To measure service cohesion for service-based systems, the Average Service Operation Cohesion (ASOC) is calculated based on SOC(s) as follows:

$$ASOC = \frac{\sum_{i=1}^{n} SOC_i}{NS}$$

NS is the number of services in the domain (NS > 0), while SOC_i is the number of cohesive values in individual services. If NS is unity, this means that all service operations are implemented in one monolithic service. Additional factors relative to cohesion can also be considered to measure the overall cohesion of a service-based system.

5.3.3 Service Operation Coupling

High coupling between services is a result of many different aspects: independency, stateless, and self-contained (Qian, Jigang et al. 2006). From an architectural perspective, coupling can be measured at several different levels of abstraction, ranging from high-level design through to executable implementations (Perepletchikov, Ryan et al. 2007). Each aspect of coupling can also be affected by a number of different factors such as service types, innovation methods, and direct/indirect relationships. There are several alternative approaches to measure coupling. For example, one straightforward method is to determine the number of messages exchanged between services and clients (Xiao-jun 2009).

In the service operation coupling metric, we focus on measuring dependency between service operations through invocation methods because of the strong impact of the service size. Fine-grained services will have greater dependency issues than coarse-grained services because they offer less functionality. Thus, in order to accommodate the overall system requirements, fine-grained services might require additional collaboration efforts and orchestrating services. The greater the number of service operations in a service, the greater the number of invocation calls that might be expected. The assumption allows us to identify service dependency between services by means of invocation operations. Qian et al. (Qian, Jigang et al. 2006) depend on service components to show dependencies by counting asynchronous and synchronous invocations with different weights for each. In contrast, we measure the average number of direct invocations at a service level regardless of service types for both synchronous and asynchronous invocations based on the classical fan-out concept. Although the asynchronous invocation method has a lower coupling effect (Qian, Jigang et al. 2006) and is the most common

mechanism (Rud, Schmietendorf et al. 2006), allocation of different weights for different invocation types is not appropriate because we believe there is no reliable relationship between the size of service and type of invocation. We define the Average Service Operation Coupling (ASOU) metric as follows:

$$ASOU = \frac{\sum_{i=1}^{n} (S_{i,sync} + S_{i,async})}{NS}$$

NS is the number of services in the domain (NS > 0), $S_{i,sync}$ is the number of synchronous invocations in a service operation and $S_{i,async}$ is the number of asynchronous invocations in a service operation. The lower the ASOU, the higher the external attributes of performance and maintainability will be. If the service granularity is higher, more invocation operations can be expected. When ASOU is equal to zero, service operations can be implemented in a single coarse-grained service.

5.4 Metrics Validation

When considering metrics as software measurements of the quality attribute, metrics need to be validated rigorously. There are two main ways to validate metrics, empirically or theoretically. In this section, we concentrate on the theoretical validation framework based on the measurement theory suggested by Briand et al. (Briand, Morasca et al. 1996). This framework proposes instinctive properties that are defined mathematically for a number of internal-structural attributes such as size, length, complexity, cohesion and coupling. This framework has also been successfully adopted for use in metrics validation research (Rossi and Fernandez 2003; Costagliola, Ferrucci et al. 2005; Perepletchikov, Ryan et al. 2007; Basci and Misra 2009; Perepletchikov, Ryan et al. 2010). In the following, we perform the theoretical validation to evaluate our metrics (ASOG, ASOM, ASOC, and ASOU) against the properties proposed by Briand et al. (Briand, Morasca et al. 1996), such as length, complexity, cohesion and coupling measurements.

Prior to defining properties, we need to define the basic representations used in patterns similar to that defined in (Briand, Morasca et al. 1996). The representation needs to be modified to represent our problem space. For example, the term “module” needs to be replaced with the term “service” because the service-oriented system is not based on modules; rather, it is based

on service compositions. Due to the fact that the service composition refers to a set of services that we have already represented as being in the service domain, we replaced the term module with the term service. In fact, a service itself might be a coarse-grained service with large functionalities that can be re-factored into a set of services. The basic definitions as follow:

Definition_1: Representation of systems and modules. *A service domain (consisting of one or more services) S will be represented as a pair <E,R>, where E represents the set of elements of S, and R is a binary realization on* E *(*$R \subseteq E \times E$*) representing the relationships between S's elements.*

Definition_2: *Given a service domain S =* <E, R>*, a service* $s = < E_S , R_S >$ *is a service of S, if and only if* $E_S \subseteq E , R_S \subseteq E_S \times E_S$ *, and* $R_S \subseteq R$.

Definition_3: Representation of service composition. *The 2-tuple* $SC =< E, R, s >$ *represents a service composition if S =* <E, R> *is a service domain that consists of a set of services according to definition˙1 and s is a collection of service operations.*

A) Average Service Operation Granularity (ASOG)

With respect to the general definition of the service granularity as software size, the size property appears to be suitable for validating the average service operation granularity (ASOG) metric. However, the ASOG metric does not satisfy the size measurement according to the third property of "module additivity," which states, "the size of services in a service domain *S = <E, R>*, is equal to the sum of the sizes of two of its services $s_1 =< E_{s1} , R_{s1} >$ and $s_2 =< E_{s2} , R_{s2} >$ such that any element of *S* is an element of either s_1 or s_2." Indeed, the calculated value of ASOG is always different because we eventually calculate the average of all services in a service domain to reach the overall value of granularity of all services in a service domain. Therefore, the length measurement is selected, since it considers more than one aspect of calculating a metric. The ASOG metric is based on the calculation of two aspects, the OFG and ODG. The ASOG is evaluated against five properties of the length measurement as follows:

- **Non-negativity:** The ASOG value has a non-negativity property. The ASOG value of a set of services S = <E, R> is non-negative in all cases, ASOG (S) ≥ 0.

- **Null Value:** The ASOG value satisfies the null value property. The ASOG value of a set of services $S = <E, R>$ is null when E is empty, $E = \emptyset \Rightarrow (ASOG\ (S) = 0)$.

- **Non-increasing Monotonicity:** The ASOG value satisfies the non-increasing monotonicity property. If S is a set of services and s is a service of S such that s is represented by a linked component of the graph representing S. Appending new (R) relationships between elements of s does not increase the ASOG value of S. For example, decomposing a service s with a number of operations $O(s)$ into fine-grained services will not increase the overall size of functionality (ASOG) in S.

  $(S = < E, R > \ and\ s = < E_S, R_S > \ \ and\ s \subseteq S$ *and s "is a linked component of S" and*

  $S' =< E, R' > \quad and\ R' = R \cup \{< e_1, e_2 >\} \quad and\ < e_1, e_2 > \notin R\ and\ e_1 \in E_{s1})$

  $$\Rightarrow ASOG(S) \geq ASOG(S')$$

- **Non-decreasing Monotonicity (non-linked services):** The ASOG value satisfies the non-decreasing monotonicity property. If S is a set of services and $s_1\ and\ s_2$ are two services of S such that $s_1\ and\ s_2$ are represented by two unlinked components of the graph representing S. Appending new (R) relationships between elements of s_1 to elements of s_s does not decrease the ASOG value of S.

  $(S = < E, R > and\ s_1 = < E_{s1}, R_{s1} > \ and\ s_2 =< E_{s2}, R_{s2} > and\ s_1 \subseteq S\ and$

  $s_2 \subseteq S"\ are\ unlinked\ services\ of\ S\ and$

  $S' =< E, R' > \ and\ R' = R \cup \{< e_1, e_2 >\} and\ < e_1, e_2 > \notin R$

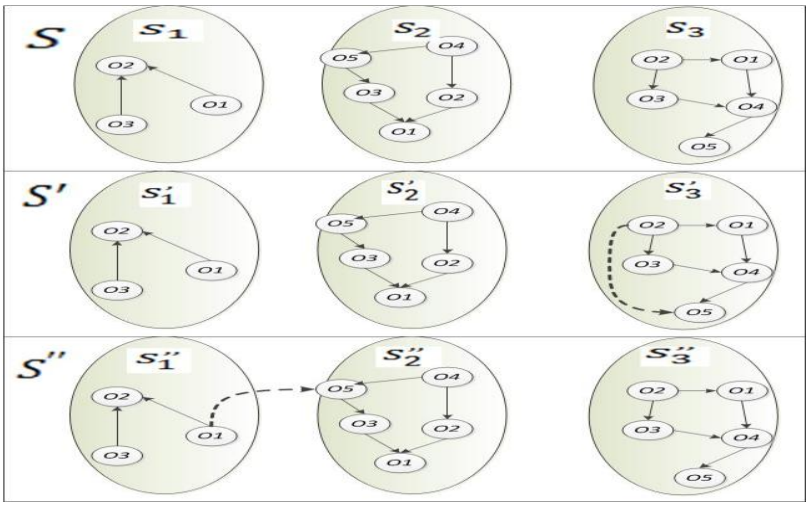
$and\ e_1\ \in E_{s1}\ and\ e_2\ \in E_{s2}\ )$

$$\Rightarrow ASOG(S) \leq ASOG(S')$$

- **Disjoint Services (modules):** The ASOG value satisfies the disjoint services property. The ASOG value of a set of services $S = < E, R >$ decomposed into two disjointed services $s_1\ and\ s_2$ is equal to the maximum of the ASOG value of $s_1 and\ s_2$. For example,

$$(S = s_1 \cup s_2\ and\ s_1 \cap s_2 = \emptyset\ and\ E = E_{s1} \cup E_{s2})$$

$$\Rightarrow ASOG(S) = \max\ \{ASOG\ (s_1), ASOG(s_2)\}$$



5-2 ASOG metrics evaluation using the properties of length

Fig. 5-2 demonstrates the three length properties: non-increasing monotonicity, non-decreasing monotonicity and disjoint services. Every *s* consists of E (elements), which represents a number of operations (o). The length of the service domain S is the maximum value among the lengths of $s_1, s_2\ and\ s_3$ which are services linked to S. The length of the service domain S′ is not greater than that of service domain S, where a new relationship $< o_2, o_5 >$ (represented by dashed arrow) links two elements of S, s_3. The length of the service domain S″ is not less than that of service domain S, where a new relationship $< o_1, o_5 >$ (represented by dashed arrow) links two elements of S, $s_1\ and\ s_2$.

B) Average Service Operation Complexity (ASOM):

The ASOM is evaluated against five properties of the complexity measurement as follows:

- **Non-negativity:** The ASOM value satisfies the non-negativity property. The ASOM value of a service domain $S = <E, R>$ is non-negative in all cases,

$$\Rightarrow ASOM\ (S)\ \geq 0$$

- **Null Value:** The ASOM value satisfies the null value property. The ASOM value of a service domain $S = <E, R>$ is null when $R = \emptyset$

$$\Rightarrow (ASOM\ (S)\ =\ 0)$$

- **Symmetry:** The ASOM value satisfies the symmetry property. The ASOM value of a service domain $S = <E, R>$ is flexible to select representation conventions between E of S. ($S = < E, R > and\ S^{-1} = < E, R-1>$,

$$\Rightarrow ASOM\ (S) = ASOM\ (S^{-1})$$

- **Service (module) Monotonicity:** The ASOM value satisfies the service monotonicity property. The ASOM value of a service domain $S = <E, R>$ is greater or equal to the sum of the values of ASOM of any two of its services that have no relationships in common.

$$(S = < E, R > and\ s_1 = < E_{s1}, R_{s1} > and\ s_2 =< E_{s2}, R_{s2} > and$$

$$s_1 \cup s_2 \subseteq S\ and\ R_{s1} \cap R_{s2} = \emptyset)$$

$$\Rightarrow ASOM\ (S) \geq ASOM\ (s_1) + ASOM(s_2)$$

- **Disjoint Services (module) Additivity:** The ASOM value satisfies the disjoint services property. The ASOM value of a set of services $S = < E, R >$ composed of two disjointed services $s_1\ and\ s_2$ is equal to the sum of the values of ASOM of $s_1\ and\ s_2$

$$(S = < E, R > and\ S = s_1 \cup s_2\ and\ s_1 \cap s_2 = \emptyset)$$

$$\Rightarrow ASOM\ (S) = ASOM\ (s_1) + ASOM\ (s_1)$$

C) Average Service Operation Cohesion (ASOC):

Since cohesion refers to the degree to which service elements are related to functionality expressed in a service, the concept of cohesion is examined at the level of services. To cover the validity on the service level as well as the a set of services (service compositions), we used a alternation symbol as " | ". e.g., the notation (S| SC), where S and SC present a cohesion for service and a service composition, respectively (Briand, Morasca et al. 1996). The ASOC is evaluated against four properties of the cohesion measurement:

- **Non-negativity and Normalization:** The ASOC value satisfies the non-negativity and normalization property. The value of ASOC where [service $s = < E_s, R_s >$ of a service composition $SC = < E, R, s >$ $| \ service\ composition\ SC < E, R, s >$] will fall within the interval between zero and unity. The ASOC meets the normalization property since the ASOC values of all services are comparable to the equivalent interval.

$$\Rightarrow [ASOC\ (s) \in [0, Max] \mid ASOC(SC) \in [0, max]\ ]$$

- **Null Value:** The value of ASOC satisfies the null value property. The ASOC value of [service $s = < E_s, R_s >$ of a service composition SC= $< E, R, s > | \ service\ composition\ \ SC =< E, R, s >$] is null when $[R_s \mid CE]$, where CE refers to common data and functional elements of exposed operations in a service.

$$[R_s = \emptyset \Rightarrow ASOC\ (s) = 0 \mid CE = \emptyset \Rightarrow ASOC\ (SC\ ) = 0]$$

- **Monotonicity:** The ASOC value satisfies the monotonicity property. The ASOC value for a service s_1 will not decrease when a new data or function element is added to operations of that service. In fact, the addition may increase the value of ASOC.

- **Cohesive Modules:** The ASOC value satisfies the cohesive modules property. If there are two unrelated (i.e., they share no common data or function types), services s_a and s_b are integrated into $s_{ab} = s_a \cup s_b$, and the value of the ASOC for s_{ab} is not greater than the maximum value of the ASOC for s_a and s_b.

$$\Rightarrow ASOC\ (S) \leq max(ASOC\ (s_1) + ASOC\ (s_1))$$

D) Average Service Operation Coupling (ASOU):

Measurement of coupling (ASOU) focuses on the invocation methods (synchronous and asynchronous) between service operations in terms of dependency. The ASOU is evaluated at the level of services and against four properties of the cohesion measurement as follows:

- **Non-negativity:** The ASOU value satisfies the non-negativity property. The ASOU value of a service [$s = <E_S, R_S>$ *of a service composition SC* | S] is non-negative in all cases where there is no dependency between service operations nor eventually between services.

$$\Rightarrow [ASOU\ (s) \geq 0 \mid ASOU(SC)\ \geq 0\ ]$$

- **Null Value:** The value of ASOU satisfies the null value property. The ASOU value of a service [$s = <E_S, R_S>$ *of a service composition* $SC = <E, R, s>$] is null in cases where there is no dependency between service operations nor eventually between services.

- **Monotonicity:** The ASOU value satisfies the monotonicity property. The ASOU value for a service s_1 or a service composition SC will not decrease when a new data or function element is added to operations of that service. In fact, this addition may result in an increase in the value of ASOU.

- **Merging of a Service or Service Composition.** The ASOU value satisfies the merging of a service or service composition property. The value of ASOU for a set of services $S = < E, R >$ will decrease when a pair of services is merged because relationships may exist between those services, thereby causing those relationships to disappear.

- **Disjoint Services (module) Additivity:** The ASOU value satisfies the disjoint services additivity property. The ASOU value of a set of services $S = < E, R >$ composed of two disjointed services $s_1\ and\ s_2$ is equal to the sum of the value of ASOU of $s_1\ and\ s_2$

$$(\ S = < E, R >\ and\ S = s_1 \cup s_2\ and\ s_1 \cap s_2 = \emptyset\ )$$

$$\Rightarrow ASOU\ (S) = ASOU\ (s_1)\ +\ ASOU\ (s_1)$$

We can state that our proposed ASOG, ASOM, ASOC, and ASOU metrics used the mathematical properties of length, complexity, cohesion and coupling, as suggested in reference (Briand, Morasca et al. 1996). Therefore, our metrics are applicable and can provide ratio scale measurements. It is important to state that our metrics satisfy the specific properties noted in each metrics proposed. However, an empirical validation will also be conducted as part of the framework validation (later in the thesis in chapter 8).

5.5 Summary

In this chapter, we proposed a service quality model that aims to provide a grounding theory to quantify service granularity. We focused on the concept of service granularity in service designs, using service and service operations as the key design constructs for analysing and deriving complexity, cohesion and coupling. We began by defining metrics for the service granularity based on two aspects of the service model, e.g., data and functional granularity. The purpose of the suggested metrics is twofold. First, the proposed metrics ASOG, ASOM, ASOC, and ASOU aim to quantify service granularity, complexity, cohesion and coupling for a given service interface. Second, and more significantly in the context of this thesis, the service metrics need to be applied at an early phase of SOA development to guide the service identification process effectively.

All metrics proposed, except coupling (ASOU), were described explicitly in an unambiguous way using the definitions of the service granularity quality model. The ASOU metric is based on measuring dependency between service operations through invocation methods because of the strong impact of the service granularity. The defined metrics were evaluated using the mathematical properties of length, complexity, cohesion and coupling as defined by Briand et al. (Briand, Morasca et al. 1996), and can therefore be considered to be theoretically valid measures. Furthermore, the metrics will be evaluated empirically when investigating the relationship between service granularity as an independent variable and the architectural quality attributes of complexity, cohesion and coupling used in the study (the empirical evaluation will be discussed in chapter 8).

After describing the framework design of the model transformation and service quality model in chapter 4 and 5. In chapter 6, we will explain the overall implementations of the framework of the service identification process

presenting into the choreography and model transformation and the service quality model.

Chapter 6 Service Identification implementation

Chapter 4 discussed the design aspects of the concept of choreography and model transformation and Chapter 5 proposed a service quality model. Both Chapters 4 and 5 represent the foundation necessary to implement the framework to deliver the optimum service interface designs. This Chapter discusses the implementation from the perspective of the research question: is it possible to generate service interface designs automatically from business process model using service choreography?

This chapter presents details of the implementation of the model transformation (model-to-model) that is based on the choreography concept and the service quality metrics. Although there are a number of approaches stated the importance of using MDA in SOA approaches, most of them do not provide implementation of MDA principles. The service identification process can be defined (as a model-driven) a separate a model for every architectural layer. These models are representative of the semantics defined in accordance with MOF specifications. Furthermore, we demonstrate how the service quality model can be implemented based on a number of service quality metrics. The implementation of the service quality model is essential to evaluate the service quality attributes for the service interface designs and to select the optimum service interface design.

In section 6.1, we begin by presenting the overall architecture of the implementation, which is divided into two architectural parts: model transformation and service quality model. Following this, section 6.2 gives a brief review of implementation issues related to model transformations. In

section 6.3, we introduce the semantic mapping between the business process choreography model, service choreographies and the service interface design; in addition, we present an algorithm for re-factoring WS-CDL code to several service interface designs in WSDL. In section 6.4, we explain the technical implementations of the transformation chain from BPMN 2.0 to WS-CDL and from WS-CDL to WSDL 2.0 using the Atlas Transformation Language (ATL). In section 6.5, we describe the detailed implementation of the second key principle of the framework architecture, which is service quality. The core component of this implementation is a parser that consists of three Java packages: service element extractor, syntax analyser and metrics calculator. This chapter concludes in section 6.6 with a summary of the important points and characteristics of the implementation.

6.1 Framework Architecture

The aim of this framework implementation is to demonstrate the possibilities of using the WS-CDL specification to enable an automated transition from a business process choreography model in BPMN 2.0 to a service interface design in WSDL. The objective is to automate the development process to the extent that it can generate the optimum service interface designs. Figure 6-1 below shows the overall implementation, which consists of two main parts: the model transformation and service quality model. These are implemented in three main phases. The implementation of the model transformation is based on the choreography and model transformation concepts as discussed in Chapter 4 and the service quality model implementation depends on the service quality attributes introduced in Chapter 5. We discuss each architectural part separately below.

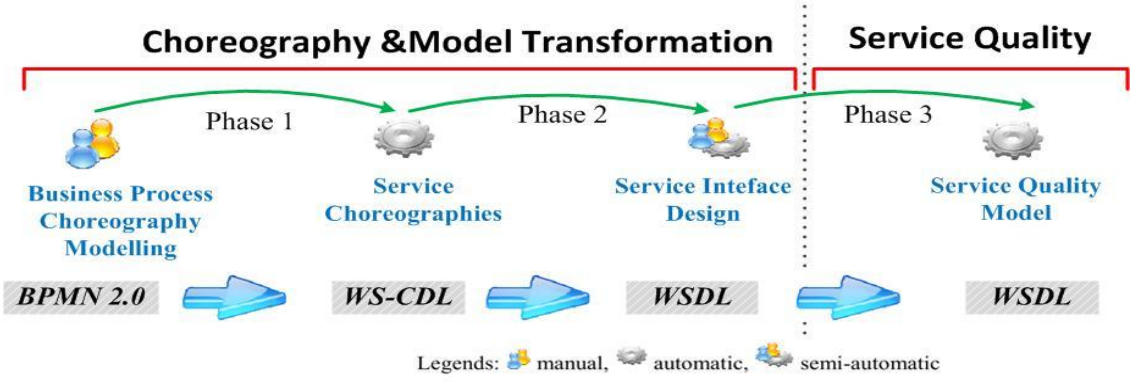


Figure 6-1 Overall Architecture of Service Identification Framework

6.2 Choreography and model transformation

In this section, we discuss the three models defined in different representations at three levels of abstraction, as well as the model representations and related issues that were considered during the implementation. This discussion is important for the underlying construction of the semantic mappings and eventually to the final technical implementation.

6.2.1 Business process choreography modelling

At an early stage of the software development cycle, business analysts construct a Business Process (BP) model corresponding to functional requirements. This model is constructed manually and independent of later development phases. In other words, the design of the BPs cannot be validated against service implementations. Furthermore, the BP design depends on a business analyst's expertise and detailed understanding of the functional requirements. Whether the design of the BP meets the functional requirements or the design of BP is completed, the representation and semantics of the design of BP in BPMN 2.0 standards cannot adequately be transformed to the next level of abstraction. As a result, in the semantics of BPMN 2.0 and the existing BPMN tools for modifying the interchange format of the BPMN diagram, we manually modified the BP diagrams (in the XMI format adapted for this thesis using the Altova XMLSPY tool). The mandatory modifications essentially covered the three elements shown in listing 6-1 and can be explained as follows:

1. The semantic of the Message element was modified through two attributes: the *messageTypedefinition* and *attributekind*. These attributes are explained in the BPMN 2.0 extension (section 4.3.1). It is important to know the schema type (e.g., XML schema, WSDL 1.1 and WSDL 2.0) used for selecting the proper interchange format when we need to define data types. In this particular case, the value assigned for the *messageTypedefinition* is "*WSDL 2.0 schema*", which is applied to element types in this case. The *attributeKind* attribute's value holds a weight value assigned based on proposed service quality model in (section 5.25.2.1). It shows the level of granularity of data exchanged and eventually allows quantification of the data granularity in a service, e.g., a

Message element with the value of *attributekind* element equals to "3" refers to the complex data type.

2. The semantic of the MessageFlow element is modified with the attribute "*actionID*", which is added through the BPMN 2.0 extension outlined in (section 4.3.1). This attribute helps to specify the appropriate data-exchange methods, such as request and respond. A given value is assigned for different data-exchange methods; in this case the value of the "*actionID*" equals "3", referring to request-respond. Although the attribute "*messageRef*" is already defined for several elements in BPMN 2.0 such as MessageFlow, the feature to associate the Message and MessageFlow through elements "*messageRef*" is not implemented. Therefore, we added this attribute manually.
3. The semantic of the Choregraphytask element was modified with the attribute "*actionType*", referring to the functional type of an operation, e.g., CRUD or executing business logic or CRUD and executing business logic. Every functional type has a different corresponding weight, as explained in section 5.2.2.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xmi:XMI xmi:version="2.0" ..........>
.....
<bpmn:Message xmi:id="...._3159F957F4D6" name="Request Form"
messageTypesdefintions="tns:WSDL2.0Schema" attributeKind="3"/>
.....
.....
<bpmn:MessageFlow xmi:id="_6-1033" name="Request Form"
messageRef="...._3159F957F4D6" actionID="1" sourceRef="....."
 targetRef="....."/>
.....
.....
<bpmn:ChoreographyTask xmi:id="....." name="Order Request"
initiatingParticipantRef="....." actionType="1" MessageFlowRef="_6-1033"
ParticipantRef="....  /  ....."  incoming="....." outgoing="....."/>
.....
.....
```

Message element modifications

MessageFlow element modifications

Choreography Task element modifications

Listing 6-1 An Example of XMI Schema with BPMN 2.0 Extension Modifications

6.2.2 Service choreographies

Although the premise of service choreography is well discussed in relation to service composition, implementation of the choreography concept is still at an early stage. Because of this lack of implementation expertise and support, we developed a transformation to generate service choreographies automatically from a higher abstraction (the business process level) using simple ATL

transformation rules. At this stage, the code of the service choreography is generated from the business process choreography (in BPMN 2.0) and then it is used to derive the designs of service interfaces (i.e., potential service implementations) confirming consistency between different models. In this way, conformance between service implementation and design is guaranteed in regard to the collaborative behaviour described in the WS-CDL code. The WS-CDL meta-model used includes semantic definitions of all the WS-CDL elements introduced in the W3C specification (W3C 2005), where the implementation of WS-CDL is presented as a mediator between the BPMN 2.0 (semantic of business process) and WSDL (service interface). There are a number of design decisions that we took in the transformation to and from WS-CDL as follows:

1. Every business process choreography process was transformed to piece of WS-CDL code, in particular to one choreography package.
2. Finding the appropriate mapping of choreography actions based on the WS-CDL specifications among the various semantic levels of business process and in regard to the design of service interfaces is essential because this is at the core of the choreography process. Generally speaking, a choreography process defines a sequence of choreography tasks, which can each be considered as an interaction block. When the semantic of the choreography task is not defined explicitly in a collaboration diagram, we consider the semantic of a message flow as an interaction block.

It is worth noting that there are some WS-CDL elements for activities notation that cannot be semantically transformed into service interfaces, such as Parallel, WorkUnit, and Sequence. In general, their behaviour is not relevant to the service interface. These elements are mainly used in the previous phase to express the behaviour of participants and activities as gateways with constraint conditions. For example, the BPMN ExclusiveGateway element was transformed to the WorkUnit element at the service choreography level, where the activities of the WorkUnit element were then transformed into basic activities (elements) at the service interface level. There are also other general elements in WS-CDL that are not especially considered in the transformation implementation between WS-CDL to WSDL, such as Participate and RelationsType element. These elements might not be applicable to service interface transformation, but they may be used in indirect service design

decisions, e.g., in developing heuristic rules for service modelling based on those services that shared the same participants (e.g., aggregating those services that are offered by one responsible participant department in an organisation).

6.2.3 Service interface design

It is difficult to design one service interface that implements the optimal functionalities of the service(s) and also satisfies the service design quality attributes. This is because considering and defining all the heuristic rules in the transformation implementation that can solve all problems is not often possible. In fact, the defined service quality attributes that are used to design a number of optimal service interfaces can be changed from time to time according to user requirements. Therefore, we developed an algorithm that generates a number of potential designs for service interfaces of service operations defined in the WS-CDL code. The number of all possible designs of service interfaces depends on the number of choreography tasks defined in the WS-CDL document, where in WSDL it is the number of service operations. This number can be calculated using the ***Bell*** number (B_n), which counts the number of all possible partitions (sub-sets) of a set with n members, where the n can be represented as the number of choreography tasks (Klazar 2003). For example, a choreography process consists of nine choreography tasks that are transformed to nine operation services in WSDL, $(B_9) = [1;\ 2;\ 5;\ 15;\ 52;\ 203 \ldots.; 21140] = 21140$. This means that there are *21,140* possible service interface designs for a service that consists of nine service operations. However, having the service interface generated automatically from the WS-CDL code, we need to consider the behaviour specifications defined for service choreographies in the generation process of all possible service interfaces. We developed an algorithm that generates five re-factored designs for service interfaces in WSDL based on the WS-CDL code. Currently, the number of generated service interfaces is limited to five re-factored designs.

As discussed previously, however, the optimum service interface design was not achieved with this transformation, although being able to measure the service quality attributes of service interface design is essential in designing the optimum service interface within specific requirements. The optimum service interface design was thus identified semi-automatically because service quality

metrics are developed as an independent application and has not fully been integrated with the implementations of model transformations.

6.3 Semantic transformation implementation

In this section, we present the semantic mapping of the two transformation phases: BPMN-to-WS-CDL and WS-CDL-to-WSDL (Figure 6-1). We also show the re-factoring algorithm that uses the semantic of WS-CDL to generate various service interface designs in WSDL. These transformation rules are implemented using ATL (ATL is explained in 2.4.4).

6.3.1 BPMN-to-WS-CDL transformation

We represent the transformation mapping from BPMN to WS-CDL in natural language. The respective transformation specification for this phase transformation was discussed in section 4.3. The transformation rules define how business process choreography in BPMN 2.0 is transformed into a service choreographies document in WS-CDL (the relevant meta-models of BPMN 2.0 and WS-CDL used in the implementation were introduced in sections 4.3 and 4.4). The implementation mapping can be described as follows:

- **Core element mapping.**
- **Specific element mapping**.

6.3.1..1 Core element mapping

The core element mapping represents the mapping of the main elements that are mandatory for establishing business process choreography between one or more participants. The elements mapping is discussed below:

- **BPMN: Definitions**

The BPMN Definitions element is an abstract class that defines the scope of the state and the namespace for all contained elements. It is a root element for diagram models; one or more definition elements are defined for the interchange of BPMN files. In WS-CDL, the Package element holds WS-CDL type definitions (e.g., informationtype and roletype elements) and namespace. Thus, we generated a WS-CDL:Package element for each instance of BPMN:Definitions and map the value of name and targetNamespace attributes of BPMN:Definitions to similar attributes in WS-CDL:Package. Concurrently,

we created initial instances for Roletype, Relationshiptype, Choreography and Informationtype elements to append their potential definitions within the elemnt WS-CDL:Package. These instances correspond to the elements of Participant, Messageflow, Choreography and Message respectively, as seen in the BPMN diagram.

1. For each BPMN ELEMENT_BPMN_Defintions : D
1.1 Read D.name, D. targetNamespace.
1.2 Create WSCDL:Package : P where
P.name = D.name , P. targetNamespace = D. targetNamespace
1.3 Collect all D instances of ELEMENT_BPMN_(Participant,
MessageFlow, Choreography, Message)
1.4 Create P instances of ELEMENT_WSCDL (roletypes,
relationshiptype, Choreography, informationtypes)

- **BPMN: Message**

The BPMN Message element is the way that participants communicate. We generated an Informationtype element in WS-CDL for every instance of the element Message and mapped the name and id attributes of the Message element to those in the Informationtype element. Furthermore, we define the appropriate data type of exchanged data based on the value of attribute "attributekind". The data schema also can be inferred from the attribute of messageTypedefinitions for communication. The definitions of the MessageFlow element were also used within the definitions of the Choreography elements in WS-CDL in order to define the Variable element.

2. For each BPMN ELEMENT_BPMN_Message : M
2.1 Read M.name, M.id.
2.2 Create WSCDL:InformationType I WHERE I.name = M.name ,I. id= M.id
2.3 Collect all S instances of ELEMENT_BPMN_ (messageTypesdefinitions)
3. Create I instance of ELEMENT_WSCDL (element)
3.1 Create WSCDL:Variable VA WHERE VA.name = M.name ,
3.2 VA.informationType= (messageTypesdefinitions) + M.name ,

- **BPMN: Participant**

The BPMN Participant element represents the role in the collaboration or choreography model. We generated the RoleType element in WS-CDL for every instance of element Participant and mapped the name and id attributes of the Participant element to that in the RoleType element. This is because the execution of the process is often the responsibility of the participant, which specifies the observable behaviour of the participant. To expose the behaviour via the WDSL interface, the interface attribute is required. Hence, we also

copied the name attribute of the Participant element to the attributes of behaviour and interface of the RoleType element.

4. For each BPMN ELEMENT_BPMN_ Participant : PA
4.1 Read PA.name , PA.id
4.2 Create WSCDL:RoleType RY WHERE RY.name = PA.name, RY.id = PA.id.
4.3 Create WSCDL:Behavior BH WHERE BH.name = PA.name and BH.interface =PA.name + 'Interface'

- **BPMN:MessageFlow**

The BPMN MessageFlow element demonstrates the flow of messages between participants. We generated the RelationType element in WS-CDL for every instance of the element MessageFlow. The two parties of the MessageFlow are defined within the attributes sourceRef and targetRef of MessageFlow, which are mapped to roletype1 and roletype2 respectively. These attribute together identify a mutual relationship between two participants.

5. For each BPMN ELEMENT_BPMN_ MessageFlow : MF
5.1 Read MF. id ,
5.2 Create WSCDL:RelationType RT WHERE RT.id = MF.id.
5.3 RT .Roletype1 = MF. SourceRef + RT. Roletype2 = MF. targetRef.

In cases where there is a collaboration diagram, the attributes of the MessageFlow element can be used to define the Interaction element within the WS-CDL, specifically the name attribute.

a. For each BPMN ELEMENT_BPMN_ MessageFlow : MF
a.1 Read MF. id ,
a.2 Create WSCDL:Interaction IT where IT.name = MF.name, IT.operation = MF.name.

- **BPMN: Choreography**

The BPMN Choreography element defines how participants interact; the interactions between participants are performed in collaboration or choreography diagrams. In WS-CDL, the choreography element defines collaborative behaviour between the interacting participants, encapsulating choreography definitions locally or globally. We generated a Choreography element in WS-CDL for the instance of the Choreography element in BPMN. In WS-CDL, the Choreography element encapsulates definitions of all activity notations (e.g., basic activity of Interactions, Order-Structures, VariableDefinition and ExceptionBlock) and these elements were defined concurrently in different transformation rules on the following defections:

6. For each BPMN ELEMENT_BPMN_ Choreography : CH
6.1 Read CH.na, CH.id
6.2 Create WSCDL:Choreography CY where CY.name = CH.name, CY.id = CH.id
6.3 Collect all CH instances of ELEMENT_BPMN (MessageFlows).
6.4 Create CY instances of ELEMENT_WSCDL (interactions).
6.5 Create CY instances of ELEMENT_WSCDL (variable).

- **BPMN: ChoreographyTask**

The BPMN ChoreographyTask element presents an interaction which results from a message being exchanged between two participants; the message exchanged is depicted as a MessageFlow element. We can consider the ChoreographyTask element to be a basic block of a choreography process, similar to that in the Interaction element in WS-CDL. Hence, we generated an Interaction element in WS-CDL for every instance of the Choreographytask element in BPMN. Since the Interaction element defines one operation and both attributes share similar behaviour, we mapped the attribute name of ChoreographyTask element to the attributes name and operation of the Interaction element. The value of the attribute actionType that refers to the operation type (e.g., CRUD) is transformed into a new similar attribute to the WS-CDL.

In fact, the Interaction element contains a number of references to the WS-CDL elements such as Roletype and InformationType. These references depend on the definitions of BPMN elements that are already defined above. In this thesis, we focused on two essential elements within the Interaction element in WS-CDL: Participate and Exchange. Firstly, the definitions of the Participate element were mapped directly from the ParticipantRef attribute of Choreographytask. In cases where the Choregraphytask element is not defined, the sourceRef and targetRef attributes of the MessageFlow element can be used to show the participants collaborating in an interaction. Secondly, the definitions of the Exchange element, which specifies exchanged data within an interaction, are transformed from Message and MessageFlow elements in BPMN. In this transformation, the data are specified using two definitions of two BPMN elements; we mapped the values of the attributes name from Message and actionID from MessageFlow elements.

7. For each BPMN ELEMENT_BPMN_ Choreographytask : CT
7.1 Read CT.name, CT.id, CT. actionType
7.2 Create WSCDL:Interaction IT where IT.name = CT.name, IT.id = CT.id ,
7.3 IT. actionType = CT. actionType
7.4 Read BPMN ELEMENT_BPMN_ Message:ME
7.5 Read BPMN ELEMENT_BPMN_ MessageFlow:MF
7.6 Create WSCDL: Participate PR WHERE PR.name = CT. ParticipantRef ,
7.7 Create WSCDL:Exchange EX WHERE EX.name = ME.name ,
and EX.informationType = MF.messageRef.
1.1 Create EX.action WHERE:
If MF.id = 1 then EX.action ='Request'
Else if MF.id = 2 then EX.action ='Respond'
Else MF.id = 3 then EX.action ='Request-Respond'

- **BPMN: EndEvent**

The BPMN Endevent element shows where a choreography process can end. WS-CDL has presented the element finalizerblock that defines confirmation of finalisation actions. Although, different patterns of the finalizerblock element can be used (i.e., the choreography may have one or more finalizerBlock) we focused on a simple pattern to indicate the completeness of the choreography process.

8. For each BPMN ELEMENT_BPMN_ EndEvent: ENE
8.1 Read ENE.name,
8.2 Create WSCDL: Finalizerblock WR where WT.name = ENE.name,

6.3.1..2 Specific element mapping

The specific element mapping represents the mapping of the elements that might not occur with every choreography process. The elements mapping is discussed below:

- **BPMN: ExclusiveGateway**

The BPMN ExclusiveGateway element defines alternative paths within a Process flow where only one path is eventually executed, in which one condition expression is evaluated (associated with the outgoing sequence flow). We generated the WorkUnit element in WS-CDL for every instance of the ExclusiveGateway element, with the name attribute of the ExclusiveGateway element mapped to that in the WorkUnit element. The outgoing sequence flow of the ExclusiveGateway element holds the expressions which can be mapped to the guard attribute of the element WorkUnit. The outgoing sequence flow of the ExclusiveGateway element points to the Choreographytask elements, which are defined explicitly within the definitions of the ExclusiveGateway element.

9. For each BPMN ELEMENT_BPMN_ ExclusiveGateway: EXG
9.1 Read EXG.name, EXG. outgoing
9.2 Create WSCDL:WorkUnit WR where WT.name = EXG.name,
9.3 Read WSCDL:SequenceFlow SF where (SF.name = EXG.Outgoing & SF.conditionExpression ≠ ∅)
9.4 EXG.guard = SF.conditionExpression
9.5 Create CY instances of ELEMENT_WSCDL_Iinteractions: IN where IN.name= SF.name

- **BPMN: EventBasedGateway**

The BPMN EventBasedGateway element defines a branching point in the process triggered by an event; the trigger is based on the receipt of a message from a participant (in greater detail, the gateways can be defined as parallel or exclusive). We generated a WorkUnit element in WS-CDL for every instance of the EventBasedGateway element, with the name attribute of EventBasedGateway element mapped to that in the WorkUnit element. When the event gateway is used to instantiate (with the instantiate attribute set as a true value) the EventBasedGateway might transform to Parallel or Exclusive elements depending on the value of the attribute eventGatewayType.

10. For each BPMN ELEMENT_BPMN_ EventBasedGateway: EBG
10.1 Read EBG.name, EBG. instantiate, EBG. eventGatewayType
10.2 Create WSCDL:WorkUnit WR where WT.name = EXG.name & EBG. instantiate = false.
10.3 Create WSCDL:Parallel PR where PR.name = EBG.name & EBG. Instantiate = true & EBG. eventGatewayType = parallel
10.4 Create WSCDL:Exclusive EX where EX.name = EBG.name & EBG. Instantiate = true & EBG. eventGatewayType = exclusive

- **BPMN: InclusiveGateway**

The BPMN InclusiveGateway element defines alternative and parallel paths within a process flow, where all condition expressions are evaluated. There are three potential elements in WS-CDL that can be used to correspond to InclusiveGateway based on the evaluation of the conditional expression: Sequence, Parallel and Choice. The outgoing sequence flow of the InclusiveGateway element points to the Choreographytask elements, which are defined explicitly within the definitions of the InclusiveGateway element.

11. For each BPMN ELEMENT_BPMN_ InclusiveGateway: IXG
11.1 Read IXG.name, IXG. outgoing
11.2 Read WSCDL:SequenceFlow SF where (SF.name = IXG. Outgoing)
If Evaluate (SF.conditionExpreseion)= SE then
Create WSCDL:Sequence SQ where SQ.name = IXG.name
elseIf Evaluate (SF.conditionExpreseion)= PA then
Create WSCDL: Parallel PA where PA.name = IXG.name
elseIf Evaluate (SF.conditionExpreseion)= CH then
Create WSCDL:Choice CH where CH.name = IXG.name
11.3 Create CY instances of ELEMENT_WSCDL_Iinteractions: IN where
11.4 IN.name= SF.name

- **BPMN: ParallelGateway**

The BPMN ParallelGateway element defines synchronised parallel paths within a process flow. We generated a Parallel element in WS-CDL for every instance of the ParallelGateway element, with the name attribute of ParallelGateway element mapped to that in the Parallel element. The outgoing sequence flow of the ParallelGateway element points to the Choreographytask elements, which are defined explicitly within the definitions of the ParallelGateway element.

12. For each BPMN ELEMENT_BPMN_ ParallelGateway: PA
12.1 Read PA.name, PA. outgoing
12.2 Create WSCDL: Parallel PAW where PAW.name = EXG.name,
12.3 Read WSCDL:SequenceFlow SF where (SF.name = EXG.Outgoing)
12.4 Create CY instances of ELEMENT_WSCDL_Iinteractions: IN where
IN.name= SF.name

- **BPMN: IntermediateEvent**

The BPMN IntermediateEvent element shows an event that happens during the process flow. This element has 12 types of intermediate events with different behaviour, especially when it has Intermediate Event as an output direction. Consequently, mapping this element directly to a specific WS-CDL element is not possible. One of these types is the IntermediateThrowEvent element, with one event definition which occurs at most once. We mapped the Choreographytask that the IntermediateThrowEvent element intends to trigger to the Interaction element in WS-CDL within the definitions of the generated order structure elements (Choice or Sequence or Parallel). Although repetition of a task might occur semantically in the WS-CDL specification, the task will be executed in the right order because of the sequential capability.

13. For each BPMN ELEMENT_BPMN_ IntermediateThrowEvent: ITE
13.1 Read ITE. outgoing
13.2 Read WSCDL:SequenceFlow SF where (SF.name = ITE.Outgoing)
13.3 Create CY instances of ELEMENT_WSCDL_Iinteractions: IN where
IN.name= SF.name

6.3.2 WS-CDL-to-WSDL transformation

In this section, we present the transformation mapping from WS-CDL to WSDL in natural language. The respective transformation specifications for this phase transformation are discussed in section 4.4. The transformation rules define how the service choreographies in the WS-CDL code are transformed into service interface designs in WSDL. The relevant meta-models of WS-CDL and WSDL used in the implementation are introduced in sections 4.4 and 4.6. The implementation mapping can be described as follows:

- **WS-CDL: Package**

The WS-CDL Package element defines the WS-CDL different type definitions (e.g., informationtype and roletype elements) and namespace. We generated a WSDL:Description element for the instance of WS-CDL:Package and mapped the value of the name and targetNamespace attributes of the WS-CDL:Package to similar attributes in the WSDL:Description. At this level, we also generated the instances of abstract definitions of Types and Interface elements corresponding to Informationtype and Choreography elements in WS-CDL, respectively. On the other hand, the concrete definitions of the Binding and Service elements were introduced individually.

1. For each WS-CDL ELELMENT_WS-CDL_Package: WSP
1.1. Read WSP.name, WSP.targetnamespace
1.2. Create WSDL:Description: DS where
DS.name = WSP.name, DS. targetnamespace = WSP.targetnamespace
1.3. Collect all Instances of ELMENT_WS-CDL(InformationType, Choreography)
1.4. Create all instance of ELEMENT_WSDL (Types, Interface)

- **WS-CDL: InformationType**

The WS-CDL InformationType element defines the data types of exchanged messages in the WS-CDL code. We generated an ElementType element in WSDL for every instance of the InformationType. To guarantee the right hierarchical structure for the XML schema definitions, we created *schema* and *elementDeclarations* attributes on the fly. We mapped seamlessly the value of the *name* attribute of InformationType to that in the ElementType, where the value of the attribute *attributeKind* defines the data types (e.g., simpleType, complexType or userDefined).

2. For each WS-CDL ELELMENT_WS-CDL_ InformationType: IT
2.1. Read IT.name, IT.attributeKind
2.2. Create WSDL:ElementType: ET where ET.name = IT.name, ET,
2.3. If IT. attributeKind = 1 then ET. attributeKind =' simpleType'
Else if IT. attributeKind = 2 then ET. attributeKind =' userDefined'
Else IT. attributeKind = 3 then ET. attributeKind =' complexType'

- **WS-CDL: Choreography**

The WS-CDL Choreography presents the definitions of collaborations that can be used to deliver the definitions of Interface, Binding and Services elements in WSDL. We generated an Interface and Service elements for the instances of the Choreography element as well as the required SOAP binding details via the Binding element. We mapped the name attribute of the Choreography element to name attribute of Interface element in WSDL. The details of the Interface element were derived from further transformations of the Interaction element to the Operation element, which we define below. We limited the generation of one interface for every service to maintain the definitions of choreographies consistent between the level of business process and service implementation.

The definitions of the Service element refer to the network addresses defined and the definitions of the Binding element including the value of the name attribute of the Choreography element mapped to the name attribute of the Service element, with the suffix "service" at the end. The definition of the Binding element is independent of the transformation process; it takes the operation attribute defined in the interface and specifies the required SOAP and HTTP binding style.

3. For each WS-CDL ELELMENT_WS-CDL_ Choreography: CH
3.1 Read CH.name, IT.attributeKind
3.2 Create WSDL:Interface: IN where IN.name = CH.name + 'Interface',
Collect all Instances of ELMENT_WS-CDL(Interaction)
Create all instances of ELEMENT_WSDL (Operation)
3.3 Create WSDL:Binding: BI where
BI.name = CH.name + 'Binding', BI.interface = CH.name + 'Interface',
BI.wsoap_protocol = "",
BI. whttp_methodDefault =" ",
Collect all Instances of ELMENT_WS-CDL(Interaction),
Create all instances of ELEMENT_WSDL (Operation),
3.4 Create WSDL:Service: SE where
SE.name = CH.name + 'Service', SE.interface = CH.name + 'Interface',
SE.endpoint = Collect all Instances of ELMENT_WS-CDL(Interaction),

- **WS-CDL: Interaction**

The WS-CDL Interaction element is at the core of the exchange of information between different services. We generated an Operation element in WSDL for every instance of the Interaction element and the values of the name and actionType attributes of Interaction are mapped to that in the Operation element. Within the interaction block, every operation processes the data exchanged (Input/Output), which is defined through the Exchange element. When Interactions are enclosed within an activity notation such as WorkUnit element, we were supposed to evaluate the guard condition to select the appropriate interaction. However, since we argue that filling the gap abstractions through transformation is not enough by itself to provide the optimal set of services, evaluating the service quality attributes is essential for delivering optimal service interface designs. Furthermore, given that we are showing how the transformation was implemented, in the next stage we will show the generation of various service interfaces with different designs.

4. For each WS-CDL ELELMENT_WS-CDL_ Interaction: IN
4.1 Read IN.name, IN. actionType
4.2 Create WSDL:Operation: OP where OP.name = IN.name,
OP. actionType =IN. actionType
Collect all Instances of ELMENT_WS-CDL(Eexchange).
Create instances of ELEMENT_WSDL (Operation).Output()
Create instances of ELEMENT_WSDL (Operation).Input()

- **WS-CDL: Exchange**

The WS-CDL Exchange element defines the data to be exchanged throughout an interaction; the exchanged data are then processed as input or output. We generated an Input or Output element in WSDL for every instance of the Exchange element. The value of the name attribute of Exchange is mapped to that in the Operation element, where the value of the action attribute is used to decide the input and output parameters of operations. For example, when the value of action is equal to "respond", the operation has an output parameter. Below we show how the input and output are implemented separately.

5. For each WS-CDL ELELMENT_WS-CDL_ Exchange: EX
5.1 Read EX.name, EX. action
5.2 Create WSDL:Input: INP where INP.name = EX.name,
If EX. action = "request" or "request-respond"
INP.messageLabel ="In",
INP.elements = EX.informationType,

6. For each WS-CDL ELELMENT_WS-CDL_ Exchange: EX
6.1 Read EX.name, EX. action
6.2 Create WSDL:Output: OUT where OUT.name = EX.name,
If EX. action = "respond" or "request-respond"
OUT.messageLabel ="Out",
OUT.elements = EX.informationType,

6.3.3 WSDL transformation (re-factoring)

We show below how the algorithm that develops five re-factored cases of service interface designs depends on the generated WS-CDL code. The five cases can be explained as follows:

First case: We created a monolithic service for all operations in a service domain. We generated a Service element with one Interface for every Choreography element in WS-CDL. Each Interaction element in WS-CDL defined within the Choreography element is mapped to an Operation element in the Interface and Binding elements. Relevant exchanged messages (parameters) are defined in the Types element.

```
int case = 5;
SWITCH  (Scenarios)
<<Case One>>
    case 1: Scenario = "1"
      FOR each Chorography_Elelement_WSCDL {
                CREAT Service_WSDL
                CREAT Interface_WSDL
          FOR each Interaction_Elelement_WSCDL {
                    DEFINE Operation_WSDL INTO Interface_WSDL
                    DEFINE Operation_WSDL INTO Binding_WSDL
                    DEFINE MESSAGE_WSDL INTO Types_WSDL
          }//END FOR
       } //END FOR

```

Second case: We created an initial Service with one Interface and then created an Operation element in the Interface and the Binding element. This had the required binding definitions for every Interaction element in the WS-CDL. Again, relevant exchanged messages (parameters) are defined in the Types element. When a WorkUnit element or ordering structure elements (e.g., Choice or Parallel) element exists, a new service was created with one Interface and Binding element. The Interaction elements were defined within those ordering structure elements in the WS-CDL as operation elements in an Interface. The required binding definitions are defined for the operations and relevant messages exchanged are also defined within Types elements. Those

Interaction elements that reside outside the ordering structures elements are defined as Operation elements, and added to the initial service interface.

```
<<Case Two>>
    case 2: Scenario = "2"
      FOR each Chorography_Elelement_WSCDL {
         CREAT Service_WSDL_1
                CREAT Interface_WSDL_1
                FOR each Interaction_Elelement_WSCDL {
                  DEFINE Operation_WSDL INTO Interface_WSDL_1
                  DEFINE Operation_WSDL INTO Binding_WSDL
                  DEFINE MESSAGE_WSDL INTO   Types_WSDL
                  IF Exist(WorkUnit_Elelement_WSCDL OR ParallelElelement_WSCDL){
                    CREAT New_Service_WSDL
                    CREAT Interface_WSDL
            IF each Interaction_Elelement_WSCDL Input =
              WorkUnit_Elelement_WSCDL Output  OR
              ParallelElelement_WSCDL Output{
                                DEFINE Operation_WSDL INTO Interface_WSDL
                                DEFINE Operation_WSDL INTO Binding_WSDL
                                DEFINE MESSAGE_WSDL INTO   Types_WSDL
                   }// END IF
                   else Exit
                  }//END else/IF
                }//END FOR
      }END// FOR -Case
```

Third case: This is similar to the second case, except that those Interaction elements residing outside of the ordering structures elements are defined as Operation elements within a new Service with one Interface element.

```
<<Case Three>>
    case 3: Scenario = "3"
      FOR each Chorography_Elelement_WSCDL
                CREAT Service_WSDL_1
                CREAT Interface_WSDL_1
                CREAT Service_WSDL_2
                CREAT Interface_WSDL_2
                  FOR each Interaction_Elelement_WSCDL
                    DEFINE Operation_WSDL INTO Interface_WSDL_1
                    DEFINE Operation_WSDL INTO Binding_WSDL
                    DEFINE MESSAGE_WSDL INTO   Types_WSDL
                    IF Exist(WorkUnit_Elelement_WSCDL OR ParallelElelement_WSCDL){
                              CREAT New_Service
                              CREAT Interface_WSDL
            IF each Interaction_Elelement_WSCDL Input =
            WorkUnit_Elelement_WSCDL Output  OR
            ParallelElelement_WSCDL Output{
                                DEFINE Operation_WSDL INTO Interface_WSDL
                                DEFINE Operation_WSDL INTO Binding_WSDL
                                DEFINE MESSAGE_WSDL INTO   Types_WSDL
                   }// END IF
                   else
                    DEFINE Operation_WSDL INTO Interface_WSDL_2
                              DEFINE Operation_WSDL INTO Binding_WSDL
                              DEFINE MESSAGE_WSDL INTO   Types_WSDL
                  }//END else/IF
                Exit For}//END FOR
            }END FOR -Case
```

Fourth case: We created an initial service and then created an operation for every interaction by mapping its exchange messages. It is similar to the third case when a WorkUnite element or ordering structure element (e.g., Choice or Parallel) exists. We created new services elements with relevant Interface, Binding, Operation and Types elements after the evaluation of the conditional expression defined in the *guard* attribute of the WorkUnit element.

```
<<Case Four>>
    case 4: Scenario = "4"
      FOR each Chorography_Elelement_WSCDL
                CREAT Service_WSDL_1
                CREAT Interface_WSDL_1
                FOR each Interaction_Elelement_WSCDL
                    DEFINE Operation_WSDL INTO Interface_WSDL_1
                    DEFINE Operation_WSDL INTO Binding_WSDL
                    DEFINE MESSAGE_WSDL INTO   Types_WSDL
                    IF Exist(WorkUnit_Elelement_WSCDL OR ParallelElelement_WSCDL){
                              IF each Interaction_Elelement_WSCDL && Value = "FALSE"{
                 CREAT New_Service
                                CREAT Interface_WSDL
                                DEFINE Operation_WSDL INTO Interface_WSDL
                                DEFINE Operation_WSDL INTO Binding_WSDL
                                DEFINE MESSAGE_WSDL INTO   Types_WSDL
                     }// END IF
                     else
                      CREAT New_Service
                              CREAT Interface_WSDL
                      DEFINE Operation_WSDL INTO Interface_WSDL
                              DEFINE Operation_WSDL INTO Binding_WSDL
                              DEFINE MESSAGE_WSDL INTO   Types_WSDL
                    }//END else/IF
                  Exit For}//END FOR
     }END// FOR -Case

```

Fifth case: we created a Service element with one Interface and Binding element. Every Interaction element in WS-CDL is defined as an Operation element within the Interface element. Its required definitions for binding and exchanging messages are defined in Binding and Types element respectively, by mapping its exchange messages.

```
<<Case Five>>
    case 5: Scenario = "5"
              FOR each Chorography_Elelement_WSCDL
       FOR each Interaction_Elelement_WSCDL
                  CREAT Service_WSDL
                  CREAT Interface_WSDL
                  DEFINE Operation_WSDL INTO Interface_WSDL
                  DEFINE Operation_WSDL INTO Binding_WSDL
                  DEFINE MESSAGE_WSDL INTO   Types_WSDL
                  }//END FOR
   }END// FOR -Case
```

6.4 Transformation chain

In this section, we briefly describe how we automatically developed the transformation from business process choreography in BPMN to WS-CDL and then from WS-CDL to WSDL. In Figure 6-2, we describe the transformation chain architecture that is based on the Model Driven Engineering introduced in section 2.4.3, where we described the underlying four layers of abstraction: M0, M1, M2, and M3. After discussing the M3 level, which represents the common meta-meta-model (MOF), in section 2.4.1, and presenting the M2 level as the specific meta-models for BPMN 2.0, WS-CDL and WSDL in Chapter 4, we explained the transformation rules developed in ATL at level M1.

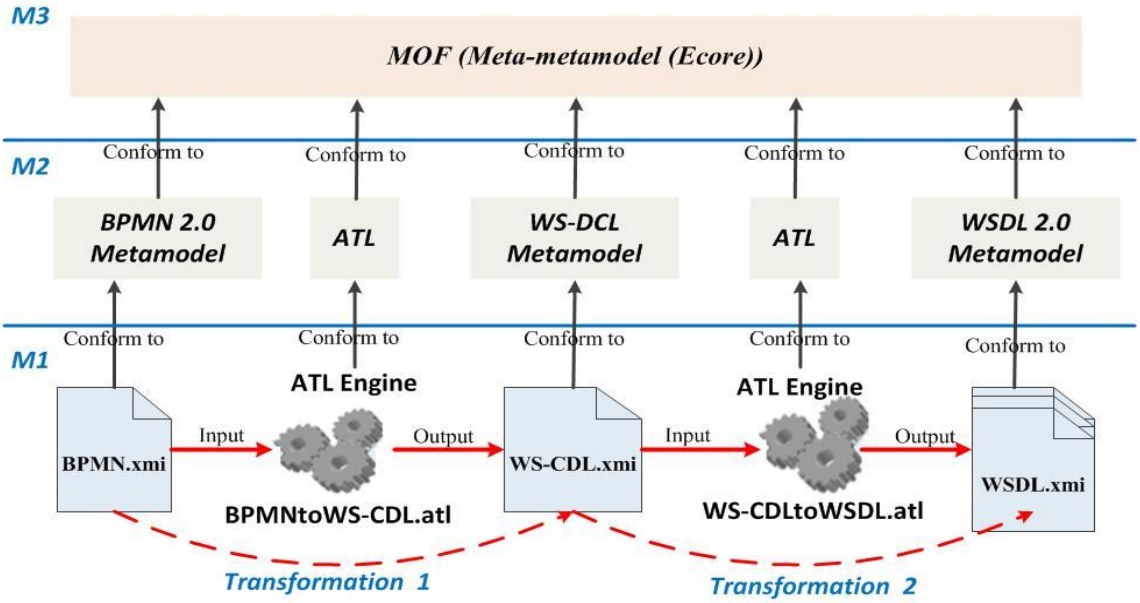


Figure 6-2 Implementation of the transformation chain

Prior to developing transformation rules in ATL, the input and target models in every transformation stage model must be confirmed to the relevant meta-models of BPMN 2.0, WS-CDL and WSDL. These defined meta-models must be confirmed to the meta-meta-model of MOF and they are developed as Ecore models based on the Eclipse Modelling Framework (EMF) (EMF was selected because the ATL integrated development environment is also built on top of the Eclipse platform). With the EMF, source and target models have to be in XMI 2.0 format. We used the Enterprise Architect (EA)[2] tool to model

[2] Enterprise Architect Case Tool by Sparx Systems Ltd available at http://www.sparxsystems.com/products/ea/index.html

the business process choreography diagrams using the BPMN 2.0 specification. The EA tool provides the capability to generate the diagrams in different formats, such as XMI and XML. All input and output models were in XMI format, as deploying the WSDL would require the conversion of WSDL (XMI) to WSDL (XML), which can be achieved using a number of available plug-ins such as ATLAS MegaModel Management (AM3). The ATL implementation consists of two transformations:

- **Transformation 1: BPMN-to-WS-CDL**

We developed ATL rules based on the semantic transformation of the first transformation BPMN-to-WS-CDL defined in (section 6.3.1) and transformed the business process choreography model in BPMN 2.0 (XMI) to service the choreographies specification in WS-CDL (XMI). The transformation rules were developed in a vertical model transformation, which means the business process choreography model (source model) and the WS-CDL (target model) exist at two different levels of abstraction. The transformation implementation developed accordingly to the choreography requirements introduced in section 4.5 (see Appendix A for an example of developed code).

- **Transformation 2: WS-CDL-to-WSDL**

We developed ATL rules based on the semantic transformation of the second transformation WS-CDL-to-WSDL defined in (section 6.3.2) and transformed the service choreographies specification in WS-CDL (XMI) to service interface design WSDL in (XMI). The transformation rules were developed in a horizontal model transformation, which means the WS-CDL specifications (source model) and the service interface design WSDL (target model) reside at the same level of abstraction. This transformation also included the implementation of the algorithm that re-factors the WS-CDL code to several service interface designs, as addressed in section 6.3.3 (see Appendix A for an example of developed code).

A number of patterns were used in the semantic transformation between the different models. The purpose of these patterns is to define the relationship between these elements in order to develop the transformation rules. These patterns are as follows:

- **One-to-One pattern**: mapping a source element to another element of another model, the source and target elements must have similar direct correspondent semantics and behaviour. An example would be the mapping of the Participant element in BPMN 2.0 to the RoleType element in WS-CDL.
- **One-to-many pattern**: an element is mapped to several elements of another model. In this pattern, the source element might have similar direct correspondent semantics and behaviour to one or more target elements collaboratively. An example would be the mapping of the Message element in BPMN 2.0 to the InformationType, Variable and Exchange elements in WS-CDL.
- **Many-to-one pattern**: several elements are mapped to one element of another model, where the source elements must collaboratively have similar behaviour to the target element. For example, the definitions of the Exchange element, which specifies exchanged data within an interaction, are transformed from Message and MessageFlow elements in BPMN. In particular, we mapped the values of the attributes name from Message and actionID from MessageFlow elements.
- **One-to-null pattern**: mapping a source element which does not have a corresponding target element on another model. Mapping such a source element is essential when constructing a new behaviour that does not exist within the new model, which entails an extension of the target meta-model. For example, adding the attribute attributekind to the InformationType and Variable elements in WS-CDL shows the data types later used for calculating the service quality attributes.
- **Null-to-one pattern**: creating a target element in a model which does not have a corresponding source element. The aim of creating this new element is to accomplish certain behaviour, for example defining a Sequence element within the Choreography element's definitions in order to direct the flow interaction execution.

6.5 Service quality model

The service quality model is implemented based on analysing syntax structures and the metrics of service quality attributes discussed in Chapter 5. The core component of our parser consists of two packages: the syntax analyser and the

metrics calculator, the two Java packages developed as a parser in Java Standard Edition (SE). Figure 6-3 shows the architecture of the parser which can be described as follows:

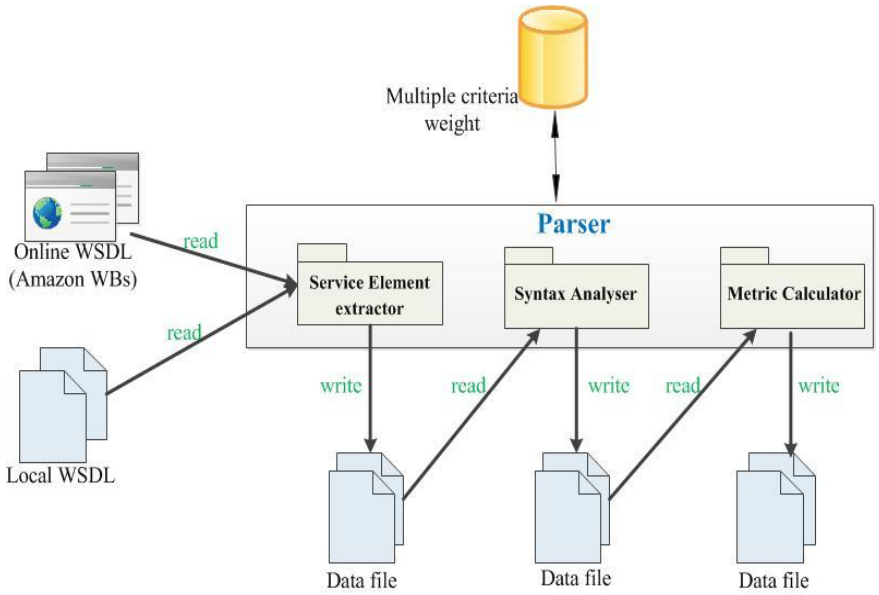


Figure 6-3 Implementation of the architecture of service quality model

- **Service Element Extractor**: the service element parser is developed to capture service elements such as messages types, operations and services; it can be used to process online web services such as Amazon WSs and internal web services. It counts the number of service elements which are then assigned their proper weight. The parser is developed on top of an open source SOA tool provided by a company called "Predic8[3]" (see Appendix A for an example code).
- **Syntax Analyser**: this package analyses and quantifies the elements of service interfaces in WSDL and quantifies syntax elements based on multiple criteria decision weights, defined as a library (see section 5.2). The output of this package is a data file for every service, which consists of four numerical lines defining properties of input messages, output messages, operation types, and number of invocations, respectively. The values of the properties defined for every data field are 1 or 5 or 10 based on the property types of the weight factor defined previously (again, see Appendix A for an example code).

[3] Predic8, available at http://predic8.com/

- **Metric Calculator:** this package calculates the metrics of ASOG ASOM, ASOC and ASOU, as defined in section 5.3. We calculated the underlying SOG metric using the service granularity quality described in section 5.2. A text file is generated for every WSDL file, which consists of the results of the four metrics mentioned previously (see Appendix A for an example code).

6.6 Summary

In this chapter, we presented the automatic generation of the service interface designs from a business process model. As Figure 6-1 shows, the framework architecture is composed of two architectural parts: the choreography and model transformation and the service quality model.

The model transformation implementation allows the service identification process to be efficiently automated, generating service interface designs. The effectiveness of this stage can be compared and evaluated against the human driven manual process, which can be expected to contain inconsistencies. We developed two transformation programs in ATL to transform business process elements in BPMN 2.0 into a service interface design via an intermediate choreography-based design; this concept is the cornerstone of model transformation. The completeness of the semantic definitions of meta-models between different models was essential to achieve the seamless transformation between source and target models. The transformations between these different abstractions required extending the semantics of BPMN 2.0 and WS-CDL to bridge any semantic gaps in the abstractions (such as message types). Although we focused on the choreography concept, our implementation also covered the collaboration model in BPMN 2.0 because of the overlap between these two concepts (a number of researchers have described collaboration as a form of interconnected choreography). The model transformation implementation is based on emerging technologies, such as the EMF and the ATL.

The current model transformation showed that it is possible to deliver service interface design automatically from a business process model. However, one fundamental drawback of this transformation must be noted, which is the dependency on the semantic completeness of the business process modelling. Current business process modelling languages such as BPMN 2.0 separate entirely the definitions of business process modelling and any potential

implementations of the business process. As a result, a complete and deployable service interface in WSDL cannot be automatically generated because there are missing semantics. For example, complete definitions for messages exchanged (attributes) in the business process choreography models (necessary for an automatic transformation to complete the definitions of the "Types" elements in WSDL).

The service quality model is developed in the Java SE environment using the service quality metrics defined previously. The implementation consists of three packages: the service element extractor, the syntax analyser and the metric calculator. We were able to quantify the service elements and to provide measurements for service granularity which potentially impact the internal service quality attributes of complexity, cohesion and coupling. This implementation allows us to evaluate different service interface designs and then decide on the most optimal service design in such cases. The level of integration with the ATL model transformation is the most severe limitation of the existing service quality model.

In Chapter 7, we will discuss the pragmatic evaluation of the model transformation implementation using three application scenarios.

Chapter 7 Pragmatic Evaluation

In Chapter 6, we discussed the implementations of the transformation models to generate a number of re-factored designs automatically of a service interface at different levels of granularity. The implementations are concerned with the solutions to the research question: it is possible to generate automatically service interface designs? Chapter 6 also described the implementation of the service quality model used to compute service quality attributes.

This Chapter focuses on evaluating the transformation models which the first part of the architectural part of the implementations proposed in Chapter 6; the evaluation is based on a pragmatic approach. For demonstrating the validation, three application scenarios are discussed. In section 7.1, we introduce briefly the pragmatic approach. In section 7.2, we discuss the hypothesis that is related to the scope of the model transformation implementation. Following, in section 7.3, we explain how the service choreographies in WS-CDL and the WSDL document are validated. In section 7.4, we use three application examples to demonstrate the use of the pragmatic approach to evaluate the framework. There are two application examples from the BPMN 2.0 OMG specification and one example from an industrial technical review of the BPMN 2.0 standard. In section 7.5, we show the limitations faced during the evaluation of research hypothesis studies in this Chapter. Finally, in section 7.6 and 7.7, we discuss reflection of research hypothesis and summarise the Chapter.

7.1 Introduction

In chapter 6, we described how model transformations can be implemented using the MDE approach. The implementation showed how we can use the choreography concept to transform a business process choreography model automatically to a service interface design. Generally speaking, there are several methods that can be used to evaluate the generation of software code from models compared to the human-manual way of doing this, these include measuring the time taken for automated transformation compared to the manual process, checking the readability of the generated code and defining benchmarks based on software quality attributes. However, it is important to track the behaviour consistency between transformed models in order to ensure the validity of the transformation.

This Chapter begins with a brief introduction of the service choreographies in WS-CDL and the design of service interfaces in WSDL. Before generating the service interface designs, the transformation process generates service choreographies in WS-CDL, which indicates how the service choreography concepts can facilitate the generation of service interfaces. The aim is to check the behavioural elements of the transformation from the business process choreography in BPMN 2.0 to the service choreographies in WS-CDL. Subsequently, the behavioural elements are traced into the WSDL documents generated to ensure that the right service behaviour is implemented. Thus, we need a pragmatic evaluation to ensure consistency between the semantics of the models generated of WS-CDL and WSDL.

7.2 Hypotheses

In this chapter, we are interested in the first hypothesis which considers consistency between the business process choreography model and the WS-CDL code and then between the WS-CDL code and service interface design in WSDL. We need to evaluate consistency to ensures that any change in the source model results in a corresponding consistent change in the target model (Mohagheghi and Dehlen 2008), the research hypothesis as follows:

H1: is it possible to use service choreographies (WS-CDL) to derive the automatic transformation of business process choreography model (BPMN 2.0) into a service interface design (WSDL)?

In the following sections, we explain how a pragmatic approach is used to evaluate consistency in the generated models of service choreography (WS-CDL code) and service interface (WSDL). We use three application scenarios to demonstrate that consistency is satisfied in each scenario. Ensuring the consistency between transformed models provides evidence that the choreography concept adapted in this thesis successfully bridge the abstraction gap between the business process modelling level and service interface design.

7.3 Pragmatic Validation

In this section, we use a pragmatic approach to validate the consistency of modelling behaviour which transforms the business process choreography models into service choreographies models. It shows how the definitions and properties of business processes choreographies are mapped to elements of service choreographies in WS-CDL. We will use examples to demonstrate the mapping and then ensure that the service choreographies generated provide complete service design interfaces in WSDL.

7.3.1 Service Choreographies (WS-CDL)

The evaluation of WS-CDL is based on two steps: validating the semantics of the XML schema and then checking the consistency of the mapping between the business process choreography model and the WS-CDL code. We first validate the WS-CDL document as XML-based language against the XML schema using a tool called "Altova XMLSpy[4]". Secondly, we ensure the transformation process has mapped all distinct elements between the BPMN 2.0 choreography process and the WS-CDL document models, while retaining the required behaviour.

The focus on the BPMN choreography process is to formalise interactions between business participants based on exchanged messages. In a pragmatic way, we validate the transferred behaviour between the input choreography process models in BPMN and the output of the service choreographies (WS-CDL). Given the specification of the choreography model from BPMN 2.0, we can construct corresponding WS-CDL elements. First, we first construct the

[4] Altova XMLSpy is an industry XML editor available at http://www.altova.com/xml-editor/

WS-CDL package corresponding to the BPMN definition, and then transform the process choreography elements.

Based on BPMN 2.0 the choreography model encapsulates the definitions of the BPMN:Message as a reference in the BPMN:ChoregraphyTask element, where each BPMN:ChoregraphyTask element might process messages as input and output. Hence, the complete definition of BPMN:Message is added to the semantics of our choreography model. This message element is essential for the extension proposed in section 4.3.1. The name and attributedkind attributes of every BPMN:Message element are translated to similar attributes of a corresponding WS-CDL:Informationtype element based one-to-one mapping pattern, where a new element attribute is created in WS-CDL:Informationtype corresponding to the messageTypesdefintions attribute which refers to the type of schema required. The element BPMN:Message is also translated to WS-CDL:Variable element-based one-one mapping pattern as definitions of variables are derived using WS-CDL:InformationType. The BPMN:Participant is translated to the WS-CDL:RoleType to exhibit the definitions of the behaviour and interface, and the name attribute of every BPMN:Participant is translated to name, behaviour and interface attributes of WS-CDL:RoleType based on the many-to-one pattern. From the BPMN:MessageFlow, we can derive definitions for WS-CDL:RelationshipType by joining the attributes of sourceRef and targetRef. We also use the attribute actionID of BPMN:MessageFlow to refer to the type of information exchanged when the message is exchanged as part of an interaction.

The WS-CDL:Choreography element encapsulates the definitions of choreography activities and collaborative behaviour where the BPMN:Choreography element refers merely to the start and end of the choreography semantics. The BPMN:ChoreographyTask element is the core element of the BPMN choreography model; it links interconnected participants through the BPMN:Participant and the Messageflow elements besides including the extension of operation type (e.g., the actionType attribute). The attributes name and actionType of the BPMN:ChoreographyTask element are translated to similar attributes of a WS-CDL:Interaction based one-to-one mapping pattern which represents a basic activity. The WS-CDL:Interaction element defines the details of interaction, i.e., the participants involved in the interaction using the reference of the attribute participate and the exchanged messages and their types using an exchange reference. Different BPMN getaway

elements such as Exclusive, Inclusive and Event-based elements are translated to WS-CDL:WorkUnite or to one of the element of the ordering structure such as the WS-CDL:Choice, WS-CDL:Parallel and WS-CDL:Sequence elements. The BPMN:Endevent states the completeness of the choreography process which translates to the WS-CDL:FinalizerBlock element. In order to specify the effect that needs to be applied by WS-CDL:FinalizerBlock element, we need to show different effects of BPMN:Endevent into semantic, .e.g., the BPMN:EndEvent of type cancel needs to be semantically translated to a specific numerical value.

First, in order to check the validation of WS-CDL codes as valid XML schema, we used the (Altova XMLspy tool). The (Altova XMLspy tool) provides support to XML-based languages' validation against XML schema. We imported the WS-CDL code for every scenario and ran the XML validation. Secondly, we ensured the consistency of behaviour across the business process choreography and the WS-CDL code. We used generated WS-CDL and WSDL documents and the hierarchical structure document to show graphically the results of mapping between BPMN 2.0 and WS-CDL and then the WSDL. The hierarchical structures demonstrate whether or not the behaviour is correctly transformed among models (An example of the hierarchical structure for the WS-CDL code is shown in Appendix B).

7.3.2 Design of Service Interfaces (WSDL)

The evaluation of the design of the service interfaces was completed in two steps: validating the semantics of the XML schema and checking the consistency of the mapping between the WS-CDL code and design of the service interface in WSDL. We first validated the WSDL document against the XML schema, as XML-based language using the (Altova XMLspy tool). Second, we ensured the WS-CDL code has been transformed into a different service interface design in the WSDL 2.0 standards and the behaviour of the WSDL document is consistent with that in the WS-CDL.

This thesis initially supports the WSDL 2.0 standard, the document structure of the WSDL 2.0 is obviously different from that in the former versions of the WSDL standards such as 1.2 and 1.1. WSDL 2.0 consists mainly of four elements: the description, interface, binding and service. Our arbitrary design of service interfaces is concerned specifically with definitions and models of data types, interface (operations) and service. As a result, our transformation

does not support the generation of the client stub and the HTTP bindings. Given WS-CDL code, we can generate a number of service interface designs in WSDL 2.0. First, we constructed the WSDL description corresponding to the package element definitions in WS-CDL, e.g., targetNamespace and the location of XSD (XML Schema). The name and attributekind attributes of WSCDL:InformationType are transformed into the WSDL:Types for XML data types, and the corresponding data type definitions computed based on the numerical value of the attribute "attributekind" in WS-CDL. The attribute "attributekind" in WS-CDL supports the proposed extension of the message types (see section 4.3.1) according to the W3C XML schema data types. The transformed definitions of WSDL:types are limited to the name and types of the data which are implicit for deriving operation definitions. It is worth noting that the level of semantic detailed in the Types element is limited because the original source of the semantic transforms from the business process choreography diagram lacks such details.

The WS-CDL:Choreography element provides the collaborative behaviour which governs the interactions via WSDL:Interaction element and order structure elements such as WSDL:WorkUnit and WSDL:Choice. The WSDL:Interface describes the operations defined by the service corresponding to the behaviour of the WS-CDL:Choreography. Although WSDL permits more than one interface element, we decided to generate one Interface element for every WSDL corresponding to the Choreography element in WS-CDL. While the WSDL:Operation is the method, the WS-CDL:Interaction element is the basic block of choreography. Both define similar behaviours by describing and processing exchanged messages (data). Thus, the WS-CDL:Interaction element is transformed to WS-CDL:Interaction with details of the attributes name and actionType to show the operation behaviour, (i.e., what operation implements CRUD function or business logic).

The data exchanged through interactions are defined in WS-CDL:Exchange which is transformed to Input/Output attributes of the WS-DL:Operation according to the value of the action attribute in the WS-CDL:Exchange. If the action value is "response", that means that the operation has an output value. The binding describes the accessibility of the web service over the protocol (currently not considered because the binding style is irrelevant to the modelling of appropriate service design). Finally, the service element is defined via the name and interface attributes as well as the

endpoint which represents eventual the service domain. In order to demonstrate our evaluation method, we will use two BPMN 2.0 scenarios for business process choreography published by OMG (OMG 2010) and one scenario published by experts from industry (Benedicto, Rosenberg et al. 2010),(the hierarchical structure for WSDL is shown in Appendix B).

7.4 Application Examples

In this section, in order to validate the transformation implementations, we apply the pragmatic evaluation on three different application scenarios. We validated the WS-CDL and WSDL documents for every example.

7.4.1 Incident Management Example

We assume that this scenario is comprehensive and representative for the choreography business process because it is published by OMG in the BPMN 2.0 specification. Figure 7-1 shows the choreography process of the Incident Management scenario which consists of nine choreography-tasks and depicts the behaviour of five participants who interact to perform business functions using seven exchanged messages. Below we evaluated the WS-CDL code and then the WSDL for this scenario.

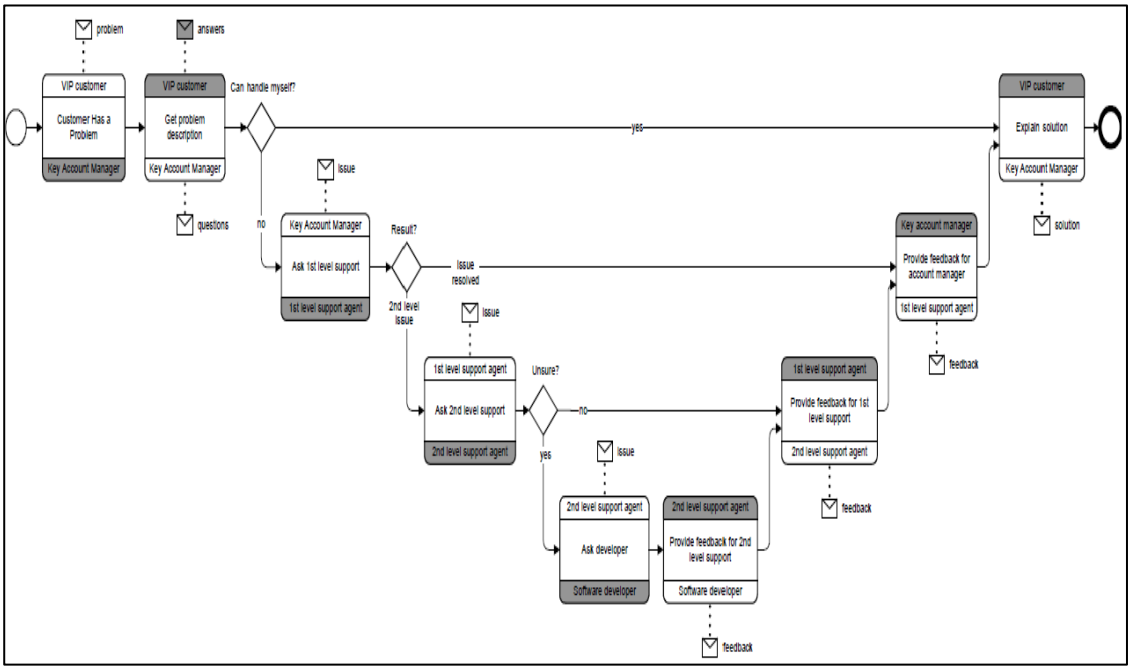


Figure 7-1 Incident Management Process Choreography

7.4.1..1 WS-CDL Validation

Listing 7-1 shows the definitions of seven WS-CDL:InformationType elements for five WS-CDL:RoleType elements that interacted through ten WS-CDL:relationshipType elements. This behaviour is the same as that of the behaviour present in the business process choreography diagram which has six BPMN:Message elements in addition to a hidden message that triggered the start event of the process choreography defined for five BPMN:Participant elements communicated through ten WS-CDL:MessageFlow elements. After defining seven possible WS-CDL:Variable elements based on existing WS-CDL:InformationType elements to capture information about objects, the choreography definition starts with a WS-CDL:Sequence element to be enabled sequentially for the defined internal activities. Three WS-CDL:WorkUnit elements are defined; each WS-CDL:WorkUnit defines internally two WS-CDL:Interaction elements, where each WS-CDL:WorkUnit element behaves similarly to the definition of three BPMN:ExclusiveGateway elements that have two outgoing paths for two BPMN:ChoreographyTask elements. The guard attribute of the WS-CDL:WorkUnit describes the constraints in similar way to the condition expressions in the BPMN:ExclusiveGateway element. Hence, the execution of the interactions element depends on the evaluation of the guard condition.

Because the WS-CDL specification does not allow internal looping WS-CDL:WorkUnit elements in a defined WS-CDL:WorkUnit element similar to that in the BPMN:ExclusiveGateway elemement, some WS-CDL:Interaction elements are defined more than once; e.g., the "Provide feedback for 1st level support" interaction is defined twice in the WS-CDL code: as part of the "Unsure" WS-CDL:WorkUnit element when the guard condition equals to "No" and as part of the standalone interactions within the WS-CDL:Sequence element. The WS-CDL:Interaction element also consists of the Exchange and Participate sub-elements which hold the semantics of the exchanged messages and participant for that particular interaction, e.g., the action required for a message such as "response". Finally, the WS-CDL:finalizerBlock confirms the completion of the choreography definition in the WS-CDL code. We can state that these behaviours described in the "IncidentMangment.cdl" are in accordance with those behaviours defined in the OMG Incident Management choreography diagram in BPMN 2.0. Table 7-1 summarises the mapping described above between BPMN elements and WS-CDL elements for this

particular scenario. The WS-CDL behaviour of the Incident Management scenario is shown in the hierarchical structure appendix B.

Table 7-1 Summary of mapping between BPMN elements and WS-CDL code for Incident Management scenario

BPMN Elements	Mapping Pattern	WS-CDL	Line of Code
Message	One-to-many	Information Type	7-13
		Variable	43-49
Participant	One-to-many	Role Type	15-29
		Participate	53,57,62,66,72,76, 81,86,90,95,99,103
Message Flow	One-to-one	Relationship Type	31-40
Choreography	One-to-one	Choreography	40
Choreography Task	One-to-one	Interaction	51-54, 55-58, 79-82, 93-96, 97-100, 101-104.
Exclusive Gateway	One-to-many	Work Unit	59-68,69-78,83-92
End Event	One-to-one	FinalizerBlock	105
Message and MessageFlow	Many-to-one	Exchange	52, 56, 61, 65 71, 75, 80, 85, 89, 94, 98, 102.
---	null-to-one	Sequence	50-106

Listing 7-1 IncidentManagment.cdl Shown with XML Schema Validation Result

```
<?xml version="1.0" encoding="UTF-8"?>
<Package version="1.0" xmlns="http://www.w3.org/2005/10/cdl"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.w3.org/2005/10/cdl IncidentManagmentExample.xsd"
xmlns:tns="http://www.example.com/IncidentManagmentExample">
    <!--Information Type Definitions-->
    <informationType name="Message_1" element="tns:Message_1" attributeKind="2"/>
    <informationType name="Message_2" element="tns:Message_2" attributeKind="1"/>
    <informationType name="Message_3" element="tns:Message_3" attributeKind="2"/>
    <informationType name="Message_4" element="tns:Message_4" attributeKind="3"/>
    <informationType name="Message_5" element="tns:Message_5" attributeKind="1"/>
    <informationType name="Message_6" element="tns:Message_6" attributeKind="3"/>
    <informationType name="Message_7" element="tns:Message_7" attributeKind="3"/>
    <!--Role definitions-->
    <roletypes name="VIPcustomer_Role">
      <behavior name="VIPcustomer_Behavior" interface="VIPcustomer_Behavior_Interface"/>
    </roletypes>
    <roletypes name="KeyAccountManager_Role">
      <behavior name="KeyAccountManager_Behavior" interface="KeyAccountManager_Behavior_Interface"/>
    </roletypes>
    <roletypes name="1stlevelsupportagent_Role">
      <behavior name="1stlevelsupportagent_Behavior" interface="1stlevelsupportagent_Behavior_Interface"/>
    </roletypes>
    <roletypes name="2ndlevelsupportagent_Role">
      <behavior name="2ndlevelsupportagent_Behavior" interface="2ndlevelsupportagent_Behavior_Interface"/>
    </roletypes>
    <roletypes name="Softwaredeveloper_Role">
      <behavior name="Softwaredeveloper_Behavior" interface="Softwaredeveloper_Behavior_Interface"/>
    </roletypes>
    <!-- Relationship definitions -->
    <relationshiptype name="Key Account Manager-VIP customer"/>
    <relationshiptype name="VIP customer-Key Account Manager"/>
    <relationshiptype name="Key Account Manager-VIP customer"/>
    <relationshiptype name="Key Account Manager-1st level support agent"/>
    <relationshiptype name="1st level support agent-2nd level support agent"/>
    <relationshiptype name="2nd level support agent-Software developer"/>
    <relationshiptype name="2nd level support agent-1st level support agent"/>
    <relationshiptype name="1st level support agent-Key Account Manager"/>
    <relationshiptype name="Software developer-2nd level support agent"/>
    <relationshiptype name="VIP customer-Key Account Manager"/>
    <!-- Choreography-->
    <Choreography name="IncidentManagement" root="false" coordination="true">
      <variable name="Message_1" element="tns:Message_1" attributeKind="2"/>
      <variable name="Message_2" element="tns:Message_2" attributeKind="1"/>
      <variable name="Message_3" element="tns:Message_3" attributeKind="2"/>
      <variable name="Message_4" element="tns:Message_4" attributeKind="3"/>
      <variable name="Message_5" element="tns:Message_5" attributeKind="1"/>
      <variable name="Message_6" element="tns:Message_6" attributeKind="3"/>
      <variable name="Message_7" element="tns:Message_7" attributeKind="3"/>
      <sequence>
        <interactions name="CustomerHasaProblem" operation="CustomerHasaProblem" actionType="1">
          <exchanges name="CustomerHasaProblemRequest" action="request" informationType="tns:Message_5"/>
          <participate relationshipType="tns:VIPcustomer2KeyAccountManager" fromRole="tns:VIPcustomer" toRole="
tns:KeyAccountManager"/>
        </interactions>
        <interactions name="Getproblemdescription" operation="Getproblemdescription" actionType="3">
          <exchanges name="GetproblemdescriptionRequest" action="request-respond" informationType="tns:Message_4
tns:Message_5"/>
          <participate relationshipType="tns:VIPcustomer2KeyAccountManager" fromRole="tns:VIPcustomer" toRole="
tns:KeyAccountManager"/>
        </interactions>
        <workunit name="Canhandlemyself?" guard="false" repeat="false">
          <interactions name="Explainsolution" operation="Explainsolution" actionType="3">
            <exchanges name="ExplainsolutionRequest" action="response" informationType="tns:Message_3"/>
            <participate relationshipType="tns:VIPcustomer2KeyAccountManager" fromRole="tns:VIPcustomer" toRole="
tns:KeyAccountManager"/>
          </interactions>
          <interactions name="Ask1stlevelsupport" operation="Ask1stlevelsupport" actionType="2">
            <exchanges name="Ask1stlevelsupportRequest" action="request" informationType="tns:Message_2"/>
            <participate relationshipType="tns:KeyAccountManager21stlevelsupportagent" fromRole="
tns:KeyAccountManager" toRole="tns:1stlevelsupportagent"/>
          </interactions>
        </workunit>
        <workunit name="Result?" guard="false" repeat="false">
          <interactions name="Ask2ndlevelsupport" operation="Ask2ndlevelsupport" actionType="2">
            <exchanges name="Ask2ndlevelsupportRequest" action="request" informationType="tns:Message_6"/>
            <participate relationshipType="tns:1stlevelsupportagent22ndlevelsupportagent" fromRole="
tns:1stlevelsupportagent" toRole="tns:2ndlevelsupportagent"/>
          </interactions>
          <interactions name="Providefeedbackforaccountmanager" operation="Providefeedbackforaccountmanager"
actionType="1">
            <exchanges name="ProvidefeedbackforaccountmanagerRequest" action="response" informationType="tns:Message_7"
/>
            <participate relationshipType="tns:KeyAccountManager21stlevelsupportagent" fromRole="tns:KeyAccountManager"
toRole="tns:1stlevelsupportagent"/>
          </interactions>
        </workunit>
        <interactions name="Explainsolution" operation="Explainsolution" actionType="3">
          <exchanges name="ExplainsolutionRequest" action="response" informationType="tns:Message_3"/>
          <participate relationshipType="tns:VIPcustomer2KeyAccountManager" fromRole="tns:VIPcustomer" toRole="
tns:KeyAccountManager"/>
        </interactions>
        <workunit name="Unsure?" guard="false" repeat="false">
          <interactions name="Providefeedbackfor1stlevelsupport" operation="Providefeedbackfor1stlevelsupport" actionType="1">
            <exchanges name="Providefeedbackfor1stlevelsupportRequest" action="response" informationType="tns:Message_7"/>
            <participate relationshipType="tns:1stlevelsupportagent22ndlevelsupportagent" fromRole="tns:1stlevelsupportagent"
toRole="tns:2ndlevelsupportagent"/>
          </interactions>
          <interactions name="Askdeveloper" operation="Askdeveloper" actionType="2">
            <exchanges name="AskdeveloperRequest" action="request" informationType="tns:Message_6"/>
            <participate relationshipType="tns:2ndlevelsupportagent2Softwaredeveloper" fromRole="tns:2ndlevelsupportagent"
toRole="tns:Softwaredeveloper"/>
          </interactions>
        </workunit>
        <interactions name="Providefeedbackfor2ndlevelsupport" operation="Providefeedbackfor2ndlevelsupport" actionType="1">
          <exchanges name="Providefeedbackfor2ndlevelsupportRequest" action="response" informationType="tns:Message_1"/>
          <participate relationshipType="tns:2ndlevelsupportagent2Softwaredeveloper" fromRole="tns:2ndlevelsupportagent" toRole
="tns:Softwaredeveloper"/>
        </interactions>
        <interactions name="Providefeedbackfor1stlevelsupport" operation="Providefeedbackfor1stlevelsupport" actionType="1">
          <exchanges name="Providefeedbackfor1stlevelsupportRequest" action="response" informationType="tns:Message_7"/>
          <participate relationshipType="tns:1stlevelsupportagent22ndlevelsupportagent" fromRole="tns:1stlevelsupportagent"
toRole="tns:2ndlevelsupportagent"/>
        </interactions>
        <interactions name="Providefeedbackforaccountmanager" operation="Providefeedbackforaccountmanager" actionType="1">
          <exchanges name="ProvidefeedbackforaccountmanagerRequest" action="response" informationType="tns:Message_7"/>
          <participate relationshipType="tns:KeyAccountManager21stlevelsupportagent" fromRole="tns:KeyAccountManager"
toRole="tns:1stlevelsupportagent"/>
        </interactions>
        <finalizeChore name="EndChore"/>
      </sequence>
    </Choreography>
</Package>
```

Text | Grid | Schema | WSDL | XBRL | Authentic | Browser

CusomterOrderProcess | CusomterOrderProcess | IncidentManagmentExample * | IncidentManagmentExample

Messages

File C:\Users\Saad\Desktop\BPMN\WSCDL files\IncidentManagmentExample.xsd is valid.

7.4.1..2 WSDL Validation

From the IncidentManagment.cdl (WS-CDL code) we generated five different designs of service interfaces in WSDL format which consist of one or more services. We validated every WSDL-document using the "Altova XMLspy tool". Five WSDL files for the Incident Management scenarios are imported and run for the WSDL validation. For brevity, we will show and discuss the results of the first re-factored scenario. Listing 7-2 represents the design of one coarse-grained service and shows that this WSDL is valid.

Here we want to check consistency of the behaviour of three main elements of any service interface design in WSDL 2.0 which are WSDL:Types, WSDL:Interface (operations) and WSDL:Service. In listing 7-2, the description of the WSDL:Types element holds the name and message-type definition of the exchanged messages (parameters), e.g., UserdefinedDefinition, SimpleTypeDefinition and ComplexTypeDefinition. The WSDL:Types element transforms the behaviour of messages resulting from interactions between participants into messages that can be used to define parameters of the Operation element; in this scenario all three data types are defined. These message types are part of the BPMN extension for XML schema-type introduced in section 4.3.1.

Although there were duplicate definitions of WS-CDL:Interaction elements in the IncidentManagement.cdl, the definition of WS-CDL:Operation elements describes concisely the behaviour based on the re-factoring algorithm. The WSDL:Interface element defines a number of operations (nine) in this particular case, which conforms to the same number of interactions in the example IncidentManagement.cdl. It concentrates on linking every Operation element with its input/output parameters through using WS-CDL:Exchange element that was previously defined in the WS-CDL:Interaction element, where its actionType is immediately transformed with its numerical value. The input/output values correspond to pre-defined schema for messages in the WSDL:Types element. The service element shows the given name of the service similar to the choreography name "IncidentManagement" and refers to the defined interface element and the address URI of the service. Table 7-2 summarises the mapping described above between WS-CDL code and WSDL document for this particular scenario. The WSDL behaviour of the Incident Management scenario is shown in the hierarchical structure appendix B.

Table 7-2 Summary of mapping between WS-CDL code and WSDL for Incident Management scenario

BPMN Elements	Mapping Pattern	WS-CDL	Line of Code
Information Type	One-to-one	Types (schema XSD)	4-19
Choreography	One-to-one	Interface	20-67
Interaction	One-to-many	Operation	22-26, 27-31, 32-36, 37-41, 42-46, 47-51, 52-56, 57-61, 62-66
Exchange	One-to-many	Input/output	23-24
Package	One-to-one	Service	81-83

Listing 7-2 IncidentManagment.wsdl Shown with XML Schema Validation Result

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<description targetNamespace="http://www.tmsws.com/wsdl20sample" xmlns="http://www.w3.org/ns/wsdl" xmlns:tns="
http://www.tmsws.com/wsdl20sample" xmlns:whttp="http://schemas.xmlsoap.org/wsdl/http/" xmlns:wsoap="
http://schemas.xmlsoap.org/wsdl/soap/">
    <!-- Types definitions -->
    <types>
      <xs:elementDeclarations name="Message_1"/>
      <xs:typeDefinition name="_1:XsUserDefinedDefinition"/>
      <xs:elementDeclarations name="Message_2"/>
      <xs:typeDefinition name="_1:XsSimpleTypeDefinition"/>
      <xs:elementDeclarations name="Message_3"/>
      <xs:typeDefinition name="_1:XsUserDefinedDefinition"/>
      <xs:elementDeclarations name="Message_4"/>
      <xs:typeDefinition name="_1:XsComplexTypeDefinition"/>
      <xs:elementDeclarations name="Message_5"/>
      <xs:typeDefinition name="_1:XsSimpleTypeDefinition"/>
      <xs:elementDeclarations name="Message_6"/>
      <xs:typeDefinition name="_1:XsComplexTypeDefinition"/>
      <xs:elementDeclarations name="Message_7"/>
      <xs:typeDefinition name="_1:XsComplexTypeDefinition"/>
    </types>
    <!-- Interface definitions -->
    <interface name="IncidentManagementInterface">
      <operation name="CustomerHasaProblem">
        <output messageLabel="Out"/>
        <input messageLabel="In" elements="tns:Message_5"/>
        <actionType>1</actionType>
      </operation>
      <operation name="Getproblemdescription">
        <output messageLabel="Out" elements="tns:Message_4 tns:Message_5"/>
        <input messageLabel="In" elements="tns:Message_4 tns:Message_5"/>
        <actionType>3</actionType>
      </operation>
      <Operation name="Providefeedbackfor2ndlevelsupport">
        <output messageLabel="Out" elements="tns:Message_1"/>
        <input messageLabel="In"/>
        <actionType>1</actionType>
      </Operation>
      <Operation name="Providefeedbackfor1stlevelsupport">
        <output messageLabel="Out" elements="tns:Message_7"/>
        <input messageLabel="In"/>
        <actionType>1</actionType>
      </Operation>
      <Operation name="Providefeedbackforaccountmanager">
        <output messageLabel="Out" elements="tns:Message_7"/>
        <input messageLabel="In"/>
        <actionType>1</actionType>
      </Operation>
      <Operation name="Explainsolution">
        <output messageLabel="Out" elements="tns:Message_3"/>
        <input messageLabel="In"/>
        <actionType>3</actionType>
      </Operation>
      <Operation name="Ask1stlevelsupport">
        <output messageLabel="Out"/>
        <input messageLabel="In" elements="tns:Message_2"/>
        <actionType>2</actionType>
      </Operation>
      <Operation name="Ask2ndlevelsupport">
        <output messageLabel="Out"/>
        <input messageLabel="In" elements="tns:Message_6"/>
        <actionType>2</actionType>
      </Operation>
      <Operation name="Askdeveloper">
        <output messageLabel="Out"/>
        <input messageLabel="In" elements="tns:Message_6"/>
        <actionType>2</actionType>
      </Operation>
    </interface>
    <!-- Binding definitions -->
    <Binding name="IncidentManagementBinding" wsoap_protocol="http://www.w3.org/2003/05/soap/bindings/HTTP/"
whttp_methodDefault="http://www.w3.org/2003/05/soap/mep/request-response" interface="tns:IncidentManagementInterface">
      <bindingOperation operation="Customer Has a Problem"/>
      <bindingOperation operation="Get problem description"/>
      <bindingOperation operation="Explain solution"/>
      <bindingOperation operation="Provide feedback for 2nd level support"/>
      <bindingOperation operation="Provide feedback for 1st level support"/>
      <bindingOperation operation="Provide feedback for account manager"/>
      <bindingOperation operation="Ask developer"/>
      <bindingOperation operation="Ask 2nd level support"/>
      <bindingOperation operation="Ask 1st level support"/>
    </Binding>
    <!-- Service definitions -->
    <Service name="IncidentManagementService" interface="tns:IncidentManagementInterface">
      <endpoint name="IncidentManagementServiceHttpEndpoint" address="http://www.IncidentManagement.com/rest/" binding
="tns:IncidentManagementInterfaceHttpBinding"/>
    </Service>
</description>
```

Text | Grid | Schema | WSDL | XBRL | Authentic | Browser

TestforWSDLCustomer_Sci1_Oper_6 | NobelPrize | IncidentManagement

Messages

File C:\Users\Saad\Desktop\BPMN\WSDL FILES\IncidentManagement.wsdl is valid.

7.4.2 Nobel Prize Example

This is the second example published by OMG in the BPMN 2.0 specification. Fig. 7-2 shows the choreography process of the Nobel Prize which consists of five choreography tasks and shows the behaviour of five participants who interact to solve business issues using seven exchanged messages. Below we evaluated the WS-CDL code and then the WSDL for this scenario.

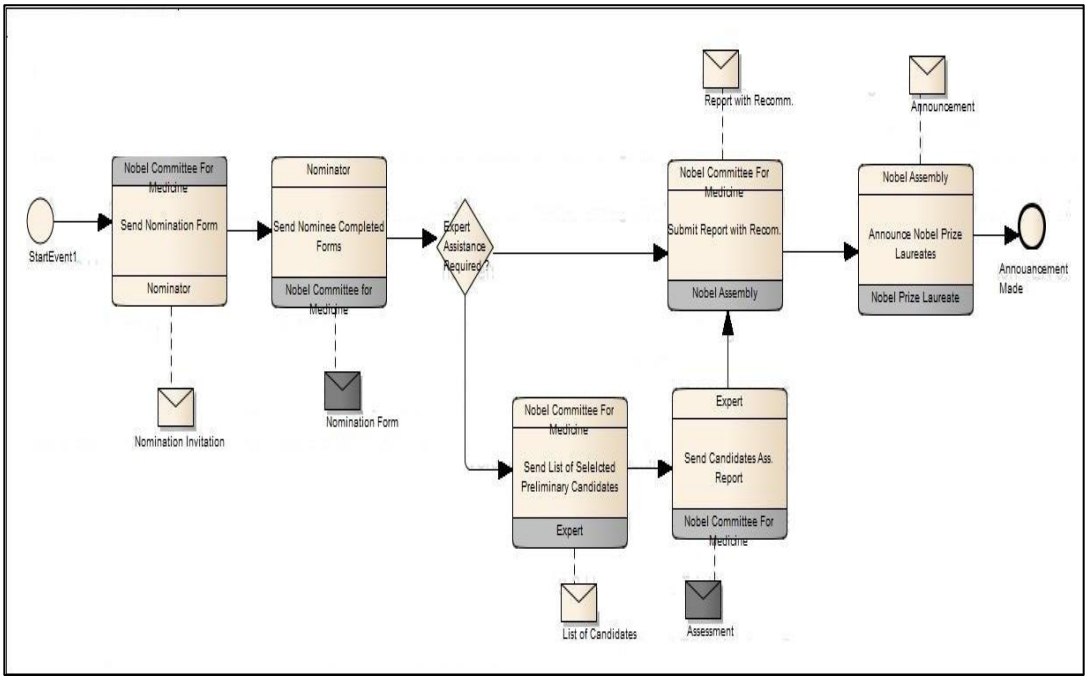


Figure 7-2 Nobel Prize Process Choreography

7.4.2..1 WS-CDL Validation

Listing 7-3 shows the definition of six WS-CDL:InformationType elements for five WS-CDL:RoleType elements that interacted through ten WS-CDL:relationshipType elements. This behaviour is same as that of the behaviour present in the business process choreography diagram which has six BPMN:Message elements for five BPMN:Participant elements communicated through six BPMN:MessageFlow elements. Six WS-CDL:Variable defined elements based on existing WS-CDL:InformationType elements are followed by WS-CDL:Sequence element. One WS-CDL:WorkUnit element is defined; each includes definitions of two WS-CDL:Interaction elements, corresponding to one BPMN:ExclusiveGateway element in BPMN that has two alternative outgoing paths. The guard condition of the WS-CDL:WorkUnit "Expert Assistance Required?" element constrains the order sequence of selecting interactions, e.g.,

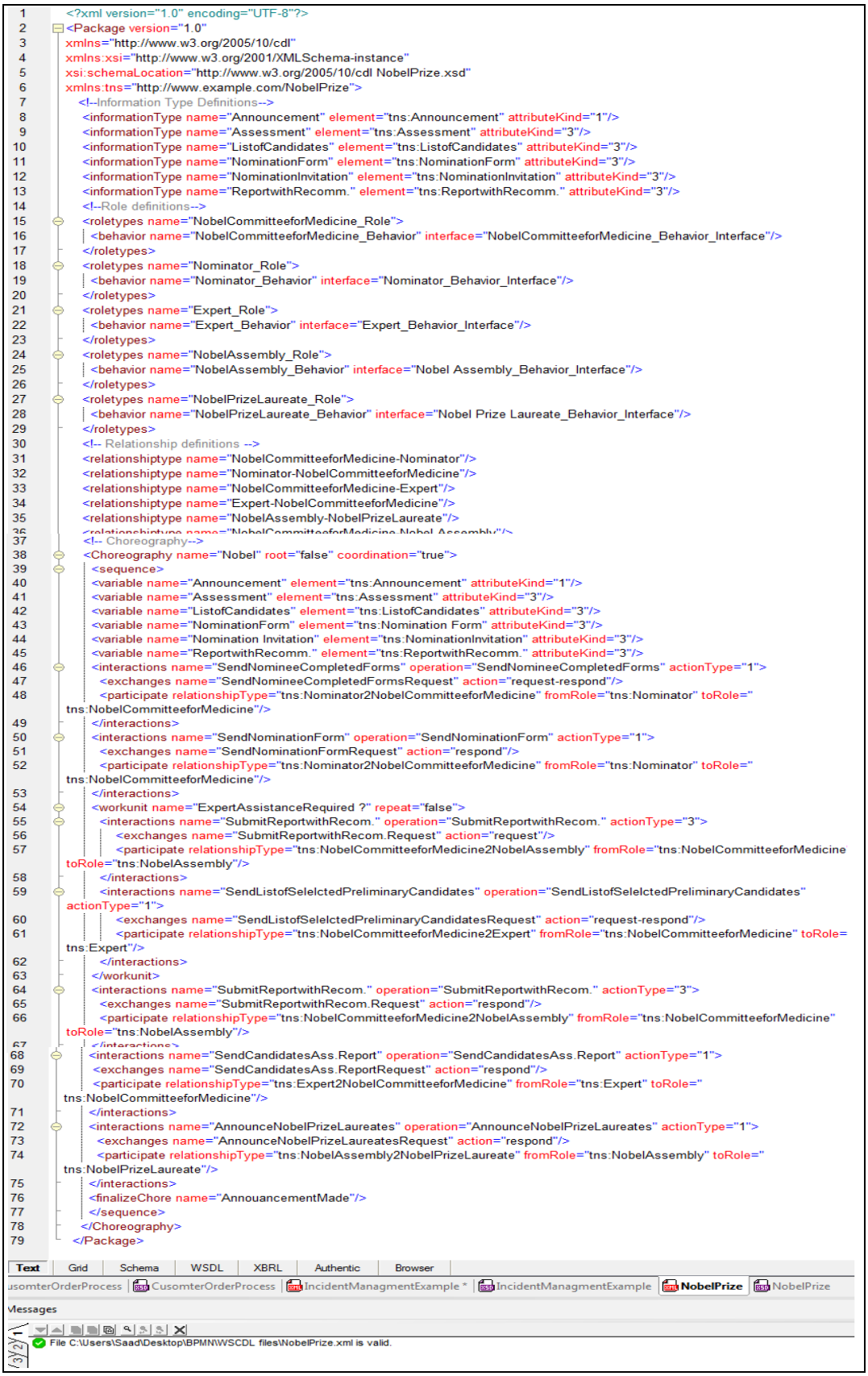
if the evaluation of the guard condition equals "false", the "Submit Report with Recom" WS-CDL:Interaction will be executed and followed by the last WS-CDL:Interaction, "Announce Nobel Prize Laureates". The use of the guard condition in the WS-CDL:WorkUnit element guarantees similar behaviour, particularly with controlling order structure by Gateway elements in the business process design. Seven WS-CDL:Interaction elements were defined for this case; the Interaction element "Submit Report with Recom" was defined twice because it appears as an alternative path of the WS-CDL:WorkUnit element, and it follows the WS-CDL:Interaction element "SendCandidatesAss.Report" in structure order.

Finally, the WS-CDL:finalizerBlock of "AnnouncementMade" ends the choreography definition in the WS-CDL code. We note that these behaviours described in the NobelPrze.cdl are in accordance with those behaviours defined in the OMG Nobel Prize choreography diagram in BPMN 2.0. Table 7-3 summarises the above described mapping between BPMN elements and WS-CDL element for this particular scenario.

Table 7-3 Summary of mapping between BPMN elements and WS-CDL code for the Nobel Prize scenario

BPMN Elements	Mapping Pattern	WS-CDL	Line of Code
Message	One-to-many	Information Type	8-13
		Variable	40-45
		Exchange	47, 51, 56, 60,56, 69,73
Participant	One-to-many	Role Type	15-29
		Participate	48, 52, 57, 61,57, 70,74
Message Flow	One-to-one	Relationship Type	31-36
Choreography	One-to-one	Choreography	37
Choreography Task	One-to-one	Interaction	51-54, 55-58, 79-82, 93-96, 97-100, 101-104.
Exclusive Gateway	One-to-many	Work Unit	54-63
End Event	One-to-one	FinalizerBlock	76
---	Null-to-one	Sequence	39-77

Listing 7-3 NobelPrize.cdl Shown with XML Schema Validation Result



```
<?xml version="1.0" encoding="UTF-8"?>
<Package version="1.0"
xmlns="http://www.w3.org/2005/10/cdl"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.w3.org/2005/10/cdl NobelPrize.xsd"
xmlns:tns="http://www.example.com/NobelPrize">
  <!--Information Type Definitions-->
   <informationType name="Announcement" element="tns:Announcement" attributeKind="1"/>
   <informationType name="Assessment" element="tns:Assessment" attributeKind="3"/>
   <informationType name="ListofCandidates" element="tns:ListofCandidates" attributeKind="3"/>
   <informationType name="NominationForm" element="tns:NominationForm" attributeKind="3"/>
   <informationType name="NominationInvitation" element="tns:NominationInvitation" attributeKind="3"/>
   <informationType name="ReportwithRecomm." element="tns:ReportwithRecomm." attributeKind="3"/>
   <!--Role definitions-->
   <roletypes name="NobelCommitteeforMedicine_Role">
    <behavior name="NobelCommitteeforMedicine_Behavior" interface="NobelCommitteeforMedicine_Behavior_Interface"/>
   </roletypes>
   <roletypes name="Nominator_Role">
    <behavior name="Nominator_Behavior" interface="Nominator_Behavior_Interface"/>
   </roletypes>
   <roletypes name="Expert_Role">
    <behavior name="Expert_Behavior" interface="Expert_Behavior_Interface"/>
   </roletypes>
   <roletypes name="NobelAssembly_Role">
    <behavior name="NobelAssembly_Behavior" interface="Nobel Assembly_Behavior_Interface"/>
   </roletypes>
   <roletypes name="NobelPrizeLaureate_Role">
    <behavior name="NobelPrizeLaureate_Behavior" interface="Nobel Prize Laureate_Behavior_Interface"/>
   </roletypes>
   <!-- Relationship definitions -->
   <relationshiptype name="NobelCommitteeforMedicine-Nominator"/>
   <relationshiptype name="Nominator-NobelCommitteeforMedicine"/>
   <relationshiptype name="NobelCommitteeforMedicine-Expert"/>
   <relationshiptype name="Expert-NobelCommitteeforMedicine"/>
   <relationshiptype name="NobelAssembly-NobelPrizeLaureate"/>
   <relationshiptype name="NobelCommitteeforMedicine-NobelAssembly"/>
   <!-- Choreography-->
   <Choreography name="Nobel" root="false" coordination="true">
    <sequence>
     <variable name="Announcement" element="tns:Announcement" attributeKind="1"/>
     <variable name="Assessment" element="tns:Assessment" attributeKind="3"/>
     <variable name="ListofCandidates" element="tns:ListofCandidates" attributeKind="3"/>
     <variable name="NominationForm" element="tns:Nomination Form" attributeKind="3"/>
     <variable name="Nomination Invitation" element="tns:NominationInvitation" attributeKind="3"/>
     <variable name="ReportwithRecomm." element="tns:ReportwithRecomm." attributeKind="3"/>
     <interactions name="SendNomineeCompletedForms" operation="SendNomineeCompletedForms" actionType="1">
      <exchanges name="SendNomineeCompletedFormsRequest" action="request-respond"/>
      <participate relationshipType="tns:Nominator2NobelCommitteeforMedicine" fromRole="tns:Nominator" toRole="
tns:NobelCommitteeforMedicine"/>
     </interactions>
     <interactions name="SendNominationForm" operation="SendNominationForm" actionType="1">
      <exchanges name="SendNominationFormRequest" action="respond"/>
      <participate relationshipType="tns:Nominator2NobelCommitteeforMedicine" fromRole="tns:Nominator" toRole="
tns:NobelCommitteeforMedicine"/>
     </interactions>
     <workunit name="ExpertAssistanceRequired ?" repeat="false">
      <interactions name="SubmitReportwithRecom." operation="SubmitReportwithRecom." actionType="3">
       <exchanges name="SubmitReportwithRecom.Request" action="request"/>
       <participate relationshipType="tns:NobelCommitteeforMedicine2NobelAssembly" fromRole="tns:NobelCommitteeforMedicine
toRole="tns:NobelAssembly"/>
      </interactions>
      <interactions name="SendListofSeleIctedPreliminaryCandidates" operation="SendListofSeleIctedPreliminaryCandidates"
actionType="1">
       <exchanges name="SendListofSeleIctedPreliminaryCandidatesRequest" action="request-respond"/>
       <participate relationshipType="tns:NobelCommitteeforMedicine2Expert" fromRole="tns:NobelCommitteeforMedicine" toRole=
tns:Expert"/>
      </interactions>
     </workunit>
     <interactions name="SubmitReportwithRecom." operation="SubmitReportwithRecom." actionType="3">
      <exchanges name="SubmitReportwithRecom.Request" action="respond"/>
      <participate relationshipType="tns:NobelCommitteeforMedicine2NobelAssembly" fromRole="tns:NobelCommitteeforMedicine"
toRole="tns:NobelAssembly"/>
     </interactions>
     <interactions name="SendCandidatesAss.Report" operation="SendCandidatesAss.Report" actionType="1">
      <exchanges name="SendCandidatesAss.ReportRequest" action="respond"/>
      <participate relationshipType="tns:Expert2NobelCommitteeforMedicine" fromRole="tns:Expert" toRole="
tns:NobelCommitteeforMedicine"/>
     </interactions>
     <interactions name="AnnounceNobelPrizeLaureates" operation="AnnounceNobelPrizeLaureates" actionType="1">
      <exchanges name="AnnounceNobelPrizeLaureatesRequest" action="respond"/>
      <participate relationshipType="tns:NobelAssembly2NobelPrizeLaureate" fromRole="tns:NobelAssembly" toRole="
tns:NobelPrizeLaureate"/>
     </interactions>
     <finalizeChore name="AnnouancementMade"/>
    </sequence>
   </Choreography>
  </Package>
```

Text | Grid | Schema | WSDL | XBRL | Authentic | Browser

usomterOrderProcess | CusomterOrderProcess | IncidentManagmentExample * | IncidentManagmentExample | NobelPrize | NobelPrize

Messages

File C:\Users\Saad\Desktop\BPMN\WSCDL files\NobelPrize.xml is valid.

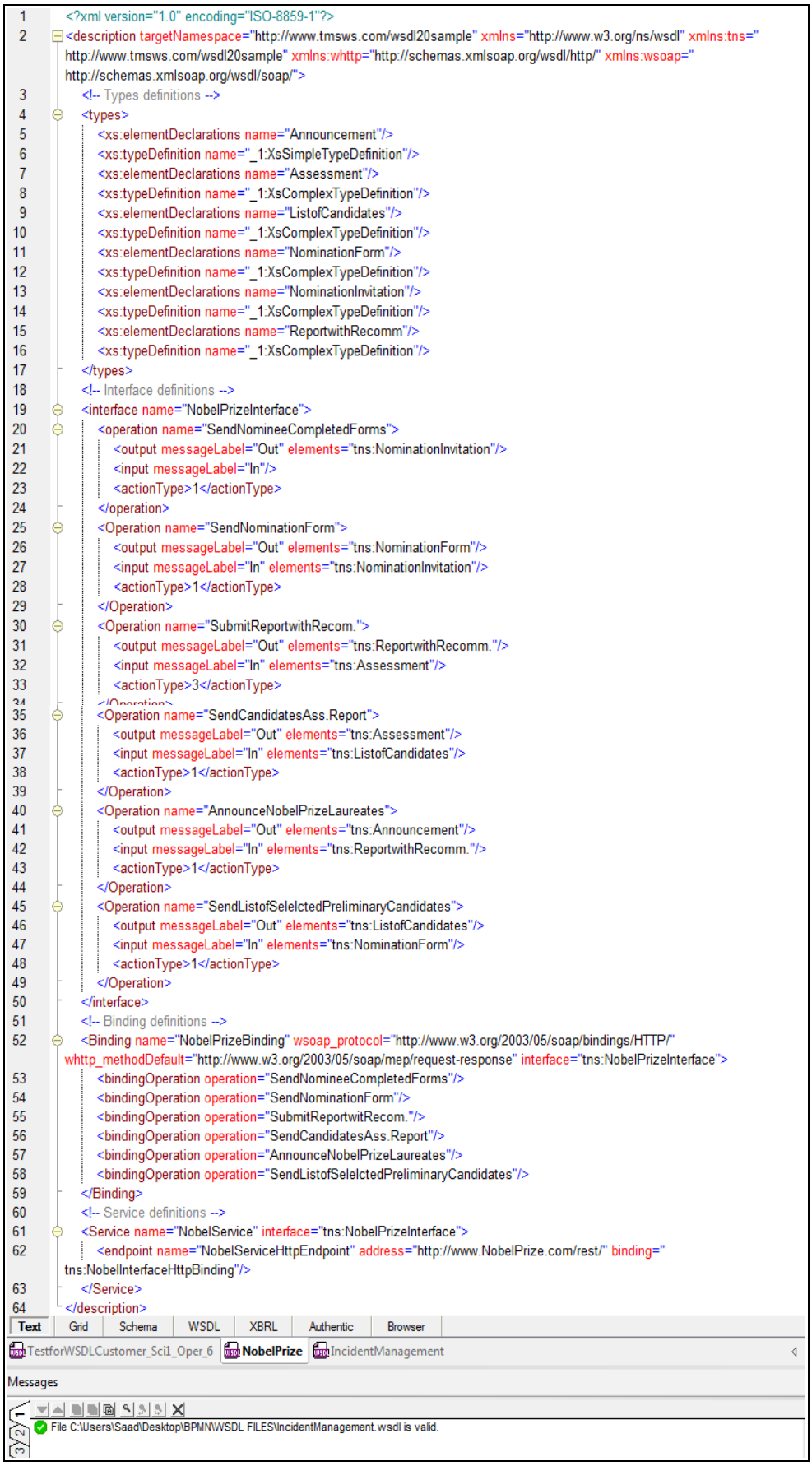
7.4.2..2 WSDL Validation

Listing 7-4 represents one of the validated cases for the OMG Nobel Prize scenarios, a design of a service interfaces as one coarse-grained service in WSDL document. All six messages defined previously in the WS-CDL code were transformed to six data types with their appropriate message-type classification as introduced within the message-type extension. Compared with the previous scenario, all data types of this scenario are defined as a complex type, which means they have similar data granularity weight. In the WSDL:*Interface* element, six WSDL:*Operation* elements are defined which conform to the same number as the interactions in NobelPrize.cdl with similar data exchanged and similar value for the *actionType* element. Finally, the WSDL:*Service* element definitions were completed as coarse-grained service with reference to the interface and the URI address. Table 7-4 summarises the above described mapping between WS-CDL code and WSDL document for this particular scenario.

Table 7-4 Summary of mapping between WS-CDL code and WSDL for the Nobel Prize scenario

BPMN Elements	Mapping Pattern	WS-CDL	Line of Code
Information Type	One-to-one	Types (schema XSD)	4-17
Choreography	One-to-one	Interface	19-50
Interaction	One-to-many	Operation	20-24, 25-29, 30-34, 35-39, 40-44, 45-49
Exchange	One-to-many	Input/output	21-22, 26-27, 31-32, 36-37, 41-42, 46-47
Package	One-to-one	Service	61-63

Listing 7-4 NobelPrize.wsdl Shown with XML Schema Validation Result



```
<?xml version="1.0" encoding="ISO-8859-1"?>
<description targetNamespace="http://www.tmsws.com/wsdl20sample" xmlns="http://www.w3.org/ns/wsdl" xmlns:tns="
http://www.tmsws.com/wsdl20sample" xmlns:whttp="http://schemas.xmlsoap.org/wsdl/http/" xmlns:wsoap="
http://schemas.xmlsoap.org/wsdl/soap/">
  <!-- Types definitions -->
  <types>
    <xs:elementDeclarations name="Announcement"/>
    <xs:typeDefinition name="_1:XsSimpleTypeDefinition"/>
    <xs:elementDeclarations name="Assessment"/>
    <xs:typeDefinition name="_1:XsComplexTypeDefinition"/>
    <xs:elementDeclarations name="ListofCandidates"/>
    <xs:typeDefinition name="_1:XsComplexTypeDefinition"/>
    <xs:elementDeclarations name="NominationForm"/>
    <xs:typeDefinition name="_1:XsComplexTypeDefinition"/>
    <xs:elementDeclarations name="NominationInvitation"/>
    <xs:typeDefinition name="_1:XsComplexTypeDefinition"/>
    <xs:elementDeclarations name="ReportwithRecomm"/>
    <xs:typeDefinition name="_1:XsComplexTypeDefinition"/>
  </types>
  <!-- Interface definitions -->
  <interface name="NobelPrizeInterface">
    <operation name="SendNomineeCompletedForms">
      <output messageLabel="Out" elements="tns:NominationInvitation"/>
      <input messageLabel="In"/>
      <actionType>1</actionType>
    </operation>
    <Operation name="SendNominationForm">
      <output messageLabel="Out" elements="tns:NominationForm"/>
      <input messageLabel="In" elements="tns:NominationInvitation"/>
      <actionType>1</actionType>
    </Operation>
    <Operation name="SubmitReportwithRecom.">
      <output messageLabel="Out" elements="tns:ReportwithRecomm."/>
      <input messageLabel="In" elements="tns:Assessment"/>
      <actionType>3</actionType>
    </Operation>
    <Operation name="SendCandidatesAss.Report">
      <output messageLabel="Out" elements="tns:Assessment"/>
      <input messageLabel="In" elements="tns:ListofCandidates"/>
      <actionType>1</actionType>
    </Operation>
    <Operation name="AnnounceNobelPrizeLaureates">
      <output messageLabel="Out" elements="tns:Announcement"/>
      <input messageLabel="In" elements="tns:ReportwithRecomm."/>
      <actionType>1</actionType>
    </Operation>
    <Operation name="SendListofSeleIctedPreliminaryCandidates">
      <output messageLabel="Out" elements="tns:ListofCandidates"/>
      <input messageLabel="In" elements="tns:NominationForm"/>
      <actionType>1</actionType>
    </Operation>
  </interface>
  <!-- Binding definitions -->
  <Binding name="NobelPrizeBinding" wsoap_protocol="http://www.w3.org/2003/05/soap/bindings/HTTP/"
whttp_methodDefault="http://www.w3.org/2003/05/soap/mep/request-response" interface="tns:NobelPrizeInterface">
    <bindingOperation operation="SendNomineeCompletedForms"/>
    <bindingOperation operation="SendNominationForm"/>
    <bindingOperation operation="SubmitReportwitRecom."/>
    <bindingOperation operation="SendCandidatesAss.Report"/>
    <bindingOperation operation="AnnounceNobelPrizeLaureates"/>
    <bindingOperation operation="SendListofSeleIctedPreliminaryCandidates"/>
  </Binding>
  <!-- Service definitions -->
  <Service name="NobelService" interface="tns:NobelPrizeInterface">
    <endpoint name="NobelServiceHttpEndpoint" address="http://www.NobelPrize.com/rest/" binding="
tns:NobelInterfaceHttpBinding"/>
  </Service>
</description>
```

7.4.3 Customer Order Example

This is third example was selected from a technical report because it represents a new BPMN 2.0 semantic that is not covered in the other two OMG scenarios, i.e., BPMN:IntermediateThrowEvent and BPMN:EventBasedGateway elements. Fig. 7-3 shows the choreography process of the Customer Order which consists of five choreography tasks and shows the behaviour of four participants who interact to solve business issues using four exchanged messages. Below we evaluated the WS-CDL code and then the WSDL for this scenario.

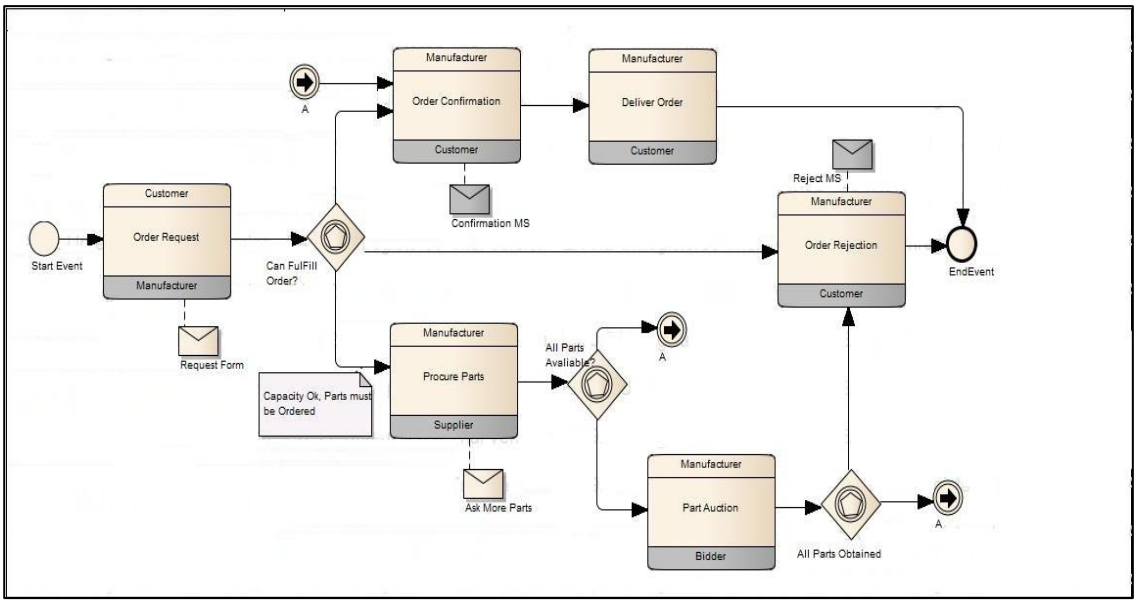


Figure 7-3 Customer Order Process Choreography

7.4.3..1 WS-CDL Validation

Listing 7-5 shows that the definitions of four WS-CDL:InformationType elements for four WS-CDL:RoleType elements who interacted through six WS-CDL:relationshipType elements. This behaviour is same as the behaviour represented in the business process choreography diagram which has four BPMN:Message elements for four BPMN:Participant elements communicated through six BPMN:MessageFlow elements. Four WS-CDL:Variable elements were defined based on existing WS-CDL:InformationType elements. The EventBasedGateway element is mapped to the WS-CDL:Choice elements in WS-CDL because the semantic of EventBasedGateway is exclusive. Three WS-CDL:Choice elements were defined including two or more Interaction elements. One of the Interaction elements will be performed, e.g., the

WS-CDL:Choice element "All Part Available" has two WS-CDL:Interaction elements, "Part Auction" and "Order Confirmation".

Since there is no element in WS-CDL that behaves similarly to the BPMN:IntermediateThrowEvent element, we consider the BPMN element that the BPMN:IntermediateThrowEvent links to. In this scenario, the BPMN:IntermediateThrowEvent attempts to link to the BPMN:ChoreographyTask element which transforms eventually to WS-CDL:Interaction elements, e.g., an BPMN:IntermediateThrowEvent element "A" throws an event in the process which links to the Interaction element "Order Confirmation". As a result, the WS-CDL:Interaction element "Order Confirmation" was redefined three times and the total number of WS-CDL:Interaction elements for this case then becomes nine compared to six BPMN:ChoreographyTask elements. Finally, the WS-CDL:finalizerBlock similar to that in "BPMN:Endevent" shows the end of the choreography definition in the WS-CDL code. We can note that these behaviours described in the CustomerOrder.cdl are in accordance with those behaviours defined in the OMG Customer Order choreography diagram in BPMN 2.0. Table 7-5 summarises the above described mapping between the BPMN elements and the WS-CDL element for this particular scenario.

Table 7-5 Summary of mapping between BPMN elements and WS-CDL code for the Customer Order scenario

BPMN Elements	Mapping Pattern	WS-CDL	Line of Code
Message	One-to-many	InformationType	4-7
		Variable	30-33
		Exchange	
Participant	One-to-many	RoleType	9-20
		Participate	
MessageFlow	One-to-one	RelationshipType	22-27
Choreography	One-to-one	Choreography	29-77
ChoreographyTask	One-to-one	Interaction	34-37, 39-42, 43-46-47-50, 53-56, 57-60,63-66,67-70, 72-75
EventBasedGateway	One-to-many	Choice	38-51, 52-61, 62-71.
IntermediateThrowEvent	One-to-one	Interaction	57-60, 67-70
End Event	One-to-one	FinalizerBlock	76

Listing 7-5 CustomerOrder.cdl Shown with XML Schema Validation Result

```
<?xml version="1.0" encoding="UTF-8"?>
<Package version="1.0" xmlns="http://www.w3.org/2005/10/cdl" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.w3.org/2005/10/cdl CusomterOrderProcess.xsd" xmlns:tns="
http://www.example.com/CusomterOrderProcess">
    <!--Information Type Definitions-->
    <informationType name="AskMoreParts" element="tns:AskMoreParts" attributeKind="3"/>
    <informationType name="ConfirmationMS" element="tns:ConfirmationMS" attributeKind="3"/>
    <informationType name="RejectMS" element="tns:RejectMS" attributeKind="3"/>
    <informationType name="RequestForm" element="tns:RequestForm" attributeKind="3"/>
    <!--Role definitions-->
    <roletypes name="Manufacture_Role">
      <behavior name="Manufacture_Behavior" interface="Manufacture_Behavior_Interface"/>
    </roletypes>
    <roletypes name="Customer_Role">
      <behavior name="Customer_Behavior" interface="Customer_Behavior_Interface"/>
    </roletypes>
    <roletypes name="Supplier_Role">
      <behavior name="Supplier_Behavior" interface="Supplier_Behavior_Interface"/>
    </roletypes>
    <roletypes name="Bidder_Role">
      <behavior name="Bidder_Behavior" interface="Bidder_Behavior_Interface"/>
    </roletypes>
    <!-- Relationship definitions -->
    <relationshiptype name="Customer2Manufacture"/>
    <relationshiptype name="Manufacture2Customer"/>
    <relationshiptype name="Manufacture2Supplier"/>
    <relationshiptype name="Manufacture2Customer"/>
    <relationshiptype name="Manufacture2Bidder"/>
    <relationshiptype name="Manufacture2Customer"/>
    <!-- Choreography-->
    <Choreography name="CustomerOrder" root="false" coordination="true">
      <variable name="AskMoreParts" element="tns:AskMoreParts" attributeKind="3"/>
      <variable name="ConfirmationMS" element="tns:ConfirmationMS" attributeKind="3"/>
      <variable name="RejectMS" element="tns:RejectMS" attributeKind="3"/>
      <variable name="RequestForm" element="tns:RequestForm" attributeKind="3"/>
      <interactions name="OrderRequest" operation="OrderRequest" actionType="1">
        <exchanges name="OrderRequestRequest" action="request-respond"/>
        <participate relationshipType="tns:Customer2Manufacture" fromRole="tns:Customer" toRole="tns:Manufacture"/>
      </interactions>
      <choice name="CanFulFillOrder?">
        <interactions name="OrderConfirmation" operation="Order Confirmation" actionType="1">
          <exchanges name="OrderConfirmationRequest" action="respond"/>
          <participate relationshipType="tns:Manufacture2Customer" fromRole="tns:Manufacture" toRole="tns:Customer"/>
        </interactions>
        <interactions name="OrderRejection" operation="OrderRejection" actionType="3">
          <exchanges name="OrderRejectionRequest" action="respond"/>
          <participate relationshipType="tns:Manufacture2Customer" fromRole="tns:Manufacture" toRole="tns:Customer"/>
        </interactions>
        <interactions name="ProcureParts" operation="ProcureParts" actionType="2">
          <exchanges name="ProcurePartsRequest" action="request-respond"/>
          <participate relationshipType="tns:Manufacture2Supplier" fromRole="tns:Manufacture" toRole="tns:Supplier"/>
        </interactions>
      </choice>
      <choice name="AllPartsObtained">
        <interactions name="PartAuction" operation="PartAuction" actionType="2">
          <exchanges name="PartAuctionRequest" action="request-respond"/>
          <participate relationshipType="tns:Manufacture2Bidder" fromRole="tns:Manufacture" toRole="tns:Bidder"/>
        </interactions>
        <interactions name="OrderConfirmation" operation="OrderConfirmation" actionType="1">
          <exchanges name="OrderConfirmationRequest" action="respond"/>
          <participate relationshipType="tns:Manufacture2Customer" fromRole="tns:Manufacture" toRole="tns:Customer"/>
        </interactions>
      </choice>
      <choice name="AllPartsAvaliable?">
        <interactions name="OrderRejection" operation="OrderRejection" actionType="3">
          <exchanges name="OrderRejectionRequest" action="respond"/>
          <participate relationshipType="tns:Manufacture2Customer" fromRole="tns:Manufacture" toRole="tns:Customer"/>
        </interactions>
        <interactions name="OrderConfirmation" operation="OrderConfirmation" actionType="1">
          <exchanges name="OrderConfirmationRequest" action="respond"/>
          <participate relationshipType="tns:Manufacture2Customer" fromRole="tns:Manufacture" toRole="tns:Customer"/>
        </interactions>
      </choice>
      <interactions name="DeliverOrder" operation="DeliverOrder" actionType="1">
        <exchanges name="DeliverOrderRequest" action="respond"/>
        <participate relationshipType="tns:Manufacture2Customer" fromRole="tns:Manufacture" toRole="tns:Customer"/>
      </interactions>
      <finalizeChore name="EndEvent"/>
    </Choreography>
</Package>
```

Text | Grid | Schema | WSDL | XBRL | Authentic | Browser

CusomterOrderProcess | IncidentManagment | IncidentManagmentExample | IncidentManagmentExample | CusomterOrderProc

Messages

File C:\Users\Saad\Desktop\BPMNWSCDL files\CusomterOrderProcess.xml is valid.

7.4.3..2 WSDL Validation

Listing 7-6 represents one of the validated cases, for the OMG Customer Order scenarios, which is the design of one coarse-grained service in the WSDL document. All four messages defined previously in the WS-CDL code were transformed to four data types with their right message-type classification as introduced on the message-type extension. These data types of this scenario are defined as a complex type. In the WSDL:*Interface* element, WSDL:*Operation* elements are defined which conform to the same number as the number of interactions in the CustomerOrder.cdl, regardless of the reparation of definitions of the WS-CDL:Interaction element "Order Confirmation", and the numerical value of WS-CDL:*actionType* element for every Operation matches correctly the same element in the CustomerOrder.cdl. Finally, the WSDL:*Service* element definitions were completed as coarse-grained service with all operations. Table 7-6 summarises the above described mapping between WS-CDL code and WSDL document for this particular scenario.

Table 7-6 Summary of mapping between WS-CDL code and WSDL for the Customer Order scenario

BPMN Elements	Mapping Pattern	WS-CDL	Line of Code
Information Type	One-to-one	Types (schema XSD)	5-14
Choreography	One-to-one	Interface	16-47
Interaction	One-to-many	Operation	17-21, 22,26, 27-31, 32-36, 37-41, 42-46
Exchange	One-to-many	Input/output	18-19, 23-24, 28-29, 33-34, 38-39, 43-44
Package	One-to-one	Service	58-60

Listing 7-6 CustomerOrder.wsdl Shown with XML Schema Validation Result

```
1   <?xml version="1.0" encoding="ISO-8859-1"?>
2   <!-- edited with XMLSpy v2012 rel. 2 (x64) (http://www.altova.com) by S.Alah (Univeristy of Southampton) -->
3   <description targetNamespace="http://www.tmsws.com/wsdl20sample" xmlns="http://www.w3.org/ns/wsdl" xmlns:tns="
    http://www.tmsws.com/wsdl20sample" xmlns:whttp="http://schemas.xmlsoap.org/wsdl/http/" xmlns:wsoap="
    http://schemas.xmlsoap.org/wsdl/soap/">
4     <!-- Types definitions -->
5     <types>
6       <xs:elementDeclarations name="Ask More Parts"/>
7       <xs:typeDefinition name="_1:XsComplexTypeDefinition"/>
8       <xs:elementDeclarations name="Confirmation MS"/>
9       <xs:typeDefinition name="_1:XsComplexTypeDefinition"/>
10      <xs:elementDeclarations name="Reject MS"/>
11      <xs:typeDefinition name="_1:XsComplexTypeDefinition"/>
12      <xs:elementDeclarations name="Request Form"/>
13      <xs:typeDefinition name="_1:XsComplexTypeDefinition"/>
14    </types>
15    <!-- Interface definitions -->
16    <interface name="CustomerOrderInterface">
17      <operation name="OrderRequest">
18        <output messageLabel="Out" elements="tns:RequestForm"/>
19        <input messageLabel="In"/>
20        <actionType>1</actionType>
21      </operation>
22      <operation name="DeliverOrder">
23        <output messageLabel="Out" elements="tns:ConfirmationMS"/>
24        <input messageLabel="In" elements="tns:RequestForm"/>
25        <actionType>1</actionType>
26      </operation>
27      <Operation name="OrderConfirmation">
28        <output messageLabel="Out" elements="tns:ConfirmationMS"/>
29        <input messageLabel="In" elements="tns:RequestForm"/>
30        <actionType>1</actionType>
31      </Operation>
32      <Operation name="OrderRejection">
33        <output messageLabel="Out" elements="tns:RejectMS"/>
34        <input messageLabel="In" elements="tns:RequestForm"/>
35        <actionType>3</actionType>
36      </Operation>
37      <Operation name="ProcureParts">
38        <output messageLabel="Out" elements="tns:AskMoreParts"/>
39        <input messageLabel="In" elements="tns:RequestForm"/>
40        <actionType>2</actionType>
41      </Operation>
42      <Operation name="Part Auction">
43        <output messageLabel="Out"/>
44        <input messageLabel="In" elements="tns:AskMoreParts"/>
45        <actionType>2</actionType>
46      </Operation>
47    </interface>
48    <!-- Binding definitions -->
49    <Binding name="Customer OrderBinding" wsoap_protocol="http://www.w3.org/2003/05/soap/bindings/HTTP/"
    whttp_methodDefault="http://www.w3.org/2003/05/soap/mep/request-response" interface="tns:CustomerOrderInterface">
50      <bindingOperation operation="Order Request"/>
51      <bindingOperation operation="Deliver Order"/>
52      <bindingOperation operation="Order Confirmation"/>
53      <bindingOperation operation="Order Rejection"/>
54      <bindingOperation operation="Procure Parts"/>
55      <bindingOperation operation="Part Auction"/>
56    </Binding>
57    <!-- Service definitions -->
58    <Service name="Customer OrderService" interface="tns:CustomerOrderInterface">
59      <endpoint name="CustomerOrderServiceHttpEndpoint" address="http://www.Customer Order.com/rest/" binding="
    tns:Customer OrderInterfaceHttpBinding"/>
60    </Service>
61  </description>
```

Text | Grid | Schema | WSDL | XBRL | Authentic | Browser

TestforWSDLCustomer_Sci1_Oper_6 | NobelPrize | IncidentManagement

Messages

File C:\Users\Saad\Desktop\BPMN\WSDL FILES\IncidentManagement.wsdl is valid.

7.5 Limitations of Pragmatic Evaluation

7.5.1 Semantic elements

Not all elements of the BPMN 2.0 choreography modelling conformance were covered in the transformation process, in that our mapping between BPMN 2.0 and WS-CDL is presently done on the basis of core behavioural elements of a choreography process. Our meta-models initially supported definitions of elements that appear in BPMN 2.0 examples (OMG 2010). A choreography diagram with compound activities such as sub-choreographies that contain more than two participants in a choreography task is not currently supported. Such elements cannot be mapped directly to the WS-CDL; it would require a normalisation stage, i.e., define a choreography composition in WS-CDL and refer to the BPMN sub-choreography as an enclosed choreography. However, there remain cases where normalisation cannot fill the gap between the choreography definitions in BPMN 2.0 and WS-CDL.

Currently, in order to bridge the semantics gap between BPMN 2.0 choreography specification for the business process model and the descriptions of service choreographies in WS-CDL, we extend the definitions of message element in BPMN 2.0 (as discussed in section 4.3.1). The aim of the extension is to facilitate the transformation process and apply the quality metrics rather than address those shortcomings of the BPMN 2.0 choreography modelling specification. We note that no implementation for the BPMN 2.0 choreography model is currently available (even a partial implementation).

7.5.2 Abstraction gap

Business process choreography diagrams are designed by a business analyst, who is usually not aware of the implementation details and not interested in knowing how business processes will be implemented. This problem refers to the abstraction gap between the level of details defined in the business model compared to the generated code (Haeng-Kon 2008). It is essential that such a level of implementation details is defined during the early stages in order to enhance the design and implementation phases for MDA approaches; e.g., defining the types of data (parameters) by the business analyst at early stage of business process design. This lack in a parameter's (messages) definitions of the

business process model is moved to the implementation phase where it requires intervention from the developer. As a result of this problem, the service interface generated by our transformation process lacks complete definitions of XML-schema for element types in WSDL.

As future work, in order to overcome this issue, we propose to extend the BPMN 2.0 definition messages/data-object within the process choreography model. The extension should include numerical types for different types of attributes in XML-schema in such a way that a business analyst can seamlessly interact with them. However, the abstraction gap is a common issue, not only with SOA but also with other software architecture, and thus the premise of MDE is to fill this gap by narrowing the problem space (Kim and Lee 2008).

7.6 Reflection on Research Hypotheses

(H1): *it is possible to generate service interface designs (WSDL) automatically from business process choreography (BPMN 2.0) using service choreographies (WS-CDL).*

To evaluate this, we traced representative modelling elements mapped using the transformation rules between source and target models. We ensured the behaviour was consistently transformed by checking the correct mapping of elements of source elements to correspondent elements in the target elements. Three application scenarios were used to evaluate transformation rules.

For each scenario, we evaluated the first phase of the transformation (BPMN-to-WS-CDL) and the second phase of the transformation (WS-CDL-to-WSDL). For the first phase, listing 7-1, 7-3 and 7-5 showed that the generated WS-CDL codes are consistent semantically with definitions of service process choreography diagrams (fig. 7-1, 7-2 and 7-3 respectively) and valid XML-based language according to the XML schema. The generation of WS-CDL code in Incident Management and Customer Order examples (fig. 7-1 and 7-3) showed that the more the business process choreography model contains gateways (event elements); the more complex are the choreography definitions. In contrast, the Nobel Prize scenario can be directly mapped with less complexity as it has only one gateway.

In the second phase, listing 7-2, 7-4 and 7-6 showed that the WSDL documents generated are consistent semantically with the definitions of

WS-CDL codes (listing 7-1, 7-2 and 7-3 respectively) and valid XML-based language according to the XML schema. The consistency of behaviour between WS-CDL and WSDL is evaluated through the correct mapping of the essential WS-CDL elements (choreography-tasks and message types) to "Operation" and passed "Types" definitions. This transformation of WS-CDL to WSDL is more syntactically oriented than that in BPMN to WS-CDL. This is because level of abstraction is closer between WS-CDL and WSDL more than that in BPMN to WS-CDL. In the other words, the specifications WS-CDL and WSDL are based on the same term "service".

7.7 Summary

In this chapter, we evaluated our framework with a pragmatic approach. The pragmatic evaluation shows that it is possible to derive a valid service interface design in WSDL based on the choreography model in WS-CDL from a business process choreography model. We used three different examples in the demonstration: the two OMG business process choreography examples from the BPMN 2.0 specification and one example from a technical report. We successfully traced elements defined in the source models through the transformation process to correspondent semantically elements in the target models. The three scenarios showed that the choreography semantic can be a mediator as it shields the complexity of business process definitions and defines the service interface seamlessly.

Furthermore, we checked these scenarios manually to establish that a similar behaviour is consistent during the transformation process, thus ensuring behavioural elements are mapped correctly. The validation is completed using the "Altova XMLspy tool". Finally, the proposed transformation is a reasonable application for the automated transformation from the business process choreography model to service interface design comparing to manual-human process. However, a drawback of the proposed transformation process is the current lack of a capability to create client stubs for generated service interfaces.

In Chapter 8, we will conduct an empirical evaluation using the service interface designs generated from these three application examples. The datasets will be processed and computed using the implementation of software quality

model design in section 6.5. The evaluation will test statistically the second and the third research hypotheses proposed in Chapter 1.

Chapter 8 Empirical Evaluation

In Chapter 7, we presented a successful pragmatic evaluation of the generation of service interface designs automatically using transformation models (the first part of the framework architecture fig. 6-1). This chapter focuses on evaluating the implementations of the service quality model (the second part of the framework architecture fig. 6-1). It evaluates the service quality model from the perspective of the research question: what is the impact of a high level of service granularity on the quality attributes of complexity and cohesion and compared to a service interface design with a low level of service granularity? It also investigates the final research question: what are the relationships between attributes of service quality?

In section 8.1, we introduce the empirical evaluation approach. This is followed, in section 8.2, by layout of the details of the second and third research hypotheses which were based on the proposed service quality model. The study design and explanations of the research variables (dependent and independent) are described in section 8.3 and 8.4, respectively. Then, in section 8.5, the descriptions of the method of data collections that supports the answer to the research hypotheses. In section 8.6, we present the statistical tests that are applied in the research and in the following section 8.7 we present the results of investigating the relationships between defined study variables. Section 8.8 presents our analysis of the study results against the details of the research hypotheses; while in section 8.9 the limitation of the empirical evaluation is discussed. Finally, in section 8.10, we summarise the results of the statistical tests.

8.1 An Empirical Evaluation

In this section, we evaluate empirically the service quality attributes of the experimental service identification process based on the service quality model defined in Chapter 5. The implementation of the service quality model showed that the service quality attributes provide benchmarks for the quality attributes of different service interface designs. The framework implementation is enhanced using benchmarks for the quality attributes of service interfaces designs generated. The aim of integrating the service quality attributes in our framework is to guide and to evaluate the service interface designs generated. In particular, the service quality attributes evaluation is conducted after re-factoring several service interface designs of a scenario.

The scope of this part of our work is to investigate the impact of service granularity on the other architectural quality attributes of complexity, cohesion and coupling using quality metrics when there is a given set of services. We also examine the dependencies between different internal architectural quality attributes. The experimental study has been conducted to show that the factors that affect a particular scenario in practice can be different compared with other scenarios. This supports our framework theoretically by studying statistically the relationships between service granularity and the other architectural attributes of complexity, cohesion and coupling.

The size of software system has often been used for measuring the development effort and cost (e.g., (Costagliola, Ferrucci et al. 2005; Nguyen 2010; Alba and Gil 2011)). In this thesis, we proposed a measurement for selecting the service granularity (software size) in the context of service-oriented architecture. This measurement is for service granularity which we validated theoretically using mathematical properties for size measurements in section 5.4. We then employed the metric of the service granularity to guide the service identification phase of the software development cycle. Here we use the empirical evaluation to verify the predicative power of our proposed service quality metrics in chapter 5. We used a dataset that is generated from the three application scenarios used for pragmatic evaluation in chapter 7.

8.2 Hypotheses

During our empirical investigation, we are interested in the second and third research hypotheses. These two research hypotheses are concerned with the

investigation of the suitability of using service quality measurements to assist the process of identifying the optimum services. Firstly, we study the second hypothesis that investigates the effect of the service granularity on the other service quality attributes of complexity, cohesion and coupling. The aim of this study is twofold: quantifying the service quality attributes to enable reasonable measurements which can be used to select the optimum services and also evaluating the implementation of the service quality model. The quality attributes are calculated based on the service quality metrics proposed in Chapter 5. The positive and negative effects are implied the direction of the relationships between the variables and degree of effects (e.g., the increase of service granularity would result in increases in service complexity which refers to a positive effect on the same direction).

Secondly, we study the third hypothesis that investigates statistically the relationships between the architectural attributes of complexity, cohesion and coupling. This study aims to examine any significant effect of these quality attributes on each other which might provide an insight to the results of the testing of the second hypothesis:

H2: a set of services with a high value of service granularity (ASOG) would correspond with a positive effect on the quality attributes of complexity (ASOM) and cohesion (ASOC) and a negative effect on the quality attribute of coupling (ASOU) compared to services with a low value of service granularity (ASOG).

To simplify the hypothesis analysis, the second hypothesis (H2) can be written as three sub- hypotheses as follows:

H2:A: *A high value of service granularity (ASOG) corresponds with a positive effect on the complexity quality attribute (ASOM).*

H2:B*: A high value of service granularity (ASOG) corresponds with a positive effect on the cohesion quality attribute (ASOC).*

H2:C: *high value of service granularity (ASOG) corresponds with a negative effect on the coupling quality attribute (ASOU).*

H3: the following architectural quality attributes are dependent on one another; cohesion is correlated with (ASOU) coupling, coupling is correlated with complexity (ASOM) and complexity (ASOM) is correlated with cohesion (ASOC).

The third hypothesis (H3) can be written as three sub- hypotheses as follows:

H3:A: *The architectural quality attributes of complexity (ASOM) and cohesion (ASOC) are correlated.*

H3:B: *The architectural quality attributes of complexity (ASOM) and coupling (ASOU) are correlated.*

H3:C: *The architectural quality attributes of cohesion (ASOC) and coupling (ASOU) are correlated.*

8.3 Study Design

Collecting data for metrics measurements is often a difficult task (Pandian 2003). The process needs to be developed in an evolutionary development style that considers a heterogeneous change in models and metric measurements. The goal of this study is to determine whether our framework can assist in developing the appropriate service interface designs to provide the appropriate level of service granularity. After generating several service interface designs using the modelling transformation, the service quality model is used not only to evaluate the service quality attributes but also to select the optimum service interface design for a given set of services. The framework provides a methodology to guide the service modelling phase, considering the impact of service granularity on architectural quality attributes. It suggests various measurements for quality attributes for a set of services in a given service domain.

8.4 Variables and Measures

To assess the feasibility of using our framework to deriving different service designs and to quantify the impact of the service granularity concept on other SOA internal architectural attributes, we then applied metrics defined in section 5.2 and 5.3 using the framework's dataset (collected from our framework scenarios)

8.4.1 Independent Variables

There is one independent variable that might have a significant influence on the final result of the experiment. In the context of appropriate service design, the service granularity reflects the independent variable as the service granularity which has been represented here by the metric Average Service Operation Granularity (ASOG). The ASOG metric defined is based on the quality model introduced in (section 5.2.3) and it quantifies the service granularity for all services in a given service domain.

$$ASOG = \frac{\sum_{i=1}^{n}(SOG(i))}{NS}$$

To be able to quantify ASOG, we develop a new quality model that provides a measurement method for Service Operation Granularity (SOG). The service granularity was firstly calculated for the operations level of a service and then for the services level of a service domain. The ASOG was then calculated for three examples (two from OMG, and one from a published academic report); and each example was re-factored to provide several service design cases.

8.4.2 Dependent Variables

There are three dependent variables defined that could have been affected by the independent variable of service granularity (ASOG) as follows:

1. The average service operation complexity (ASOM) metric focuses on the functionality aspect of the complexity quality attribute (defined as part of service operation granularity (SOG)). The ASOM is a metric defined based on the quality model introduced in (section 5.3.1):

$$ASOM = \frac{\left(\sum_{i=1}^{n}(SOG(i))^{2}\right)}{NS}$$

2. The average service operation cohesion (ASOC) metric considers the occurrence of similar size of data and the operation types of service operations based on the service operation cohesion $\mathrm{SOC(s)}$ metric previously defined. Thus, the cohesion metric is initially calculated on a

service level then applied to all services on a service domain. The ASOC metric is defined (using the quality metrics explained in section 5.3.2) as follows:

$$\text{ASOC} = \frac{\sum_{i=1}^{n} \text{SOC}_i}{\text{NS}}$$

3. The average service operation coupling (ASOU) metric measures dependency between service operations through invocation methods (synchronous and asynchronous) to take account of the strong impact of service size. The ASOC metrics is defined (using the quality metric explained in section 5.3.3) as follows:

$$ASOU = \frac{\sum_{i=1}^{n} (S_{i,sync} + S_{i,async})}{NS}$$

8.5 Research Data

This section describes how the dataset was collected and used to describe and explore the research study. The dataset was generated from our experimental framework. The dataset extraction and processing are completed using a parser which was developed as part our framework (the parser implementation was explained in section6.5).

The dataset was collected based on the WS-CDL document (XML format) produced automatically from the three scenarios used to demonstrate the BPMN to WS-CDL transformation (these are defined in section 6.3.1). We generated five re-factored designs of service interfaces for each application examples using the algorithm which we discussed in section 6.3.3. As result, five cases (in the form of the WSDL documents generated for every example) were processed through the syntax analyser in text data to produce the four metrics of ASOG, ASOM, ASOC and ASOU. Table 8-1 shows the computation of the four metrics for the dataset.

Table 8-1 Metric Results for Framework Dataset

Example	Scenario	ASOG	ASOM	ASOC	ASOU
OMG Incident Management NO = 9	1	0.722	0.076	0	0
	2	1.087	1.204	0	2
	3	1.148	1.408	0	1.6
	4	1.106	1.291	0.142	1
	5	1.111	1.333	0	1
OMG Nobel Prize NO = 5	1	0.629	0.395	0.5	0
	2	0.75	0.625	0.25	1
	3	1	1	0	0
	4	0.850	0.767	0.166	0.333
	5	0.875	0.916	0	1
Procurement NO = 5	1	0.653	0.426	0.333	0
	2	0.833	0.722	0	0.25
	3	0.8	0.666	0	0.4
	4	1	1	0.1	0.6
	5	1	1	0	0.5

8.6 The Data Analysis

The dataset was generated from our parser as text files. SPSS[5] (a statistical analysis tool) is used to analyse our data and conduct descriptive and statistical testing. The SPSS tool was selected because it is a well-accepted and widely used.

8.6.1 Descriptive Statistics

Descriptive statistics are used in different ways to present different characteristics of dataset graphs and statistical techniques. We will use graphs to show how our framework might assist in deciding which service design models best meet the given design requirements.

In order to check the normality of the datasets and to decide appropriate statistical tests, we use the Shapiro-Wilk test. This tests a composite hypothesis and is suitable for a small number of samples (less than 50 samples); a Sig (P-value) of the Shapiro-Wilk greater than 0.05 indicates that the data are normally distributed (Shapiro and Wilk 1965) (it measures the skewness of the data distribution (referred to the asymmetry). We also use the quantile-quantile (Q-Q plot) as graphical tests that examine whether or not the

[5] http://www-01.ibm.com/software/analytics/spss/downloads/

data is normal. Data appears as a linear line suggests a normal distribution, where data appears as a nonlinear line suggests a distribution that is not normal.

8.6.2 Statistical Testing

The relationships between bivariate or multivariate data can be effectively defined and the degree and direction can be measured by using statistical tests such as correlation (Johnson and Bhattacharyya 1986). It is important to analyse the nature of the relationships between variables to find out if one manipulates the other to apply the appropriate test types. In our study, we will apply correlation tests using Pearson's (r) technique because our data are based on the ratio-type when the data are normally distributed. Otherwise, Spearman's correlation coefficient (rs) will be used. The Pearson's technique is also called the linear or product-moment correlation and is intended to be used to describe the association between continuous variables. The value of a correlation coefficient varies between -1 and +1. For further analysis, the correlation coefficient value can be interpreted within different scales (strong, moderate or weak); for instance, $r = \pm.70$ represents a very strong relationship, from ±.40 to ±.69 represents a strong relationship, from ±.30 to ±.39 represents a moderate relationship, from ±.20 to ±.29 represents a weak relationship and from ±.01 to ±.19 represents no or negligible relationship (Cohen 1988). Pearson's test assumes the variables are normally distributed and there are no outliers (Kowalski 1972). If any skewed data are very small, outliers can be removed from the data after scanning the data using the scatter-plot chart. However, the Spearman's correlation coefficient rs is also a correlation test which can be used when the data are not normally distributed, and with all types of scale measurements.

8.6.3 Regression Analysis

In order to analyse existing relationships between different dependent and independent variables, we apply regression analysis. Regression analysis is a method to discover the relationship between one or more dependent variables and independent variables (Yan and Su 2009). The casual relationship between

two quantitative variables can be measured using regression analysis (Johnson and Bhattacharyya 1986) (this is also called the "line of best fit").

There are three types of regression; simple linear, multiple linear and nonlinear regressions. Simple linear regression studies the linear relationship between two variables and assumes that one variable (independent) controls the other one (dependent). In other words, linear regression represents the equation y= a + bx, where y is the predictor (dependent) variable and x is the response (independent) variable, where the value of ***a*** is constant and b is the slope of the linear equation. The relationship can be demonstrated graphically as a straight line where the independent variable is multiplied by the slope coefficient and a constant is added. When there are more than one independent variables and one dependent, multiple linear regression is applied. Multiple regression studies the linear relationship between one dependent variable and several independent variables. It assumes that the response variable has a linear relationship in a model with several predictor variables (Yan and Su 2009). The formula of the multiple regression models is:

$$y = \beta 0 + \beta 1\, x + \cdots + \beta p\, xp + \varepsilon,$$

Where $0, 1, 2, \ldots . \mathrm{p}$ are regression coefficients, $\mathrm{x}_1, \mathrm{x}_2, \mathrm{x}_3, , \mathrm{x}_\mathrm{n}$ are independent variables, y is the dependent variable and ε is an error value. Finally, nonlinear regression studies any kind of relationship between dependent and independent variables that is not linear. It is also called "nonlinear least squares fittings", and always assumes that there is a nonlinear relationship depending on one or more undefined parameters. There are two main types of nonlinear models; polynomial models and alternate nonlinear models (Munson 2003). We tested several nonlinear regression models such as Cubic, Exponential, Quadratic and Power. Finally, we found that the Cubic model behaves better and gives usually practical results with a quality of fit corresponding to a high R^2. The Cubic model represents a nonlinear regression polynomial degree-three equation of "best fit". The formula of the equation of cubic regression is

$$y = ax^3 + bx^2 + cx + d\,, \qquad a \neq 0\,,$$

Where a, b and c are regression coefficients, d is a fixed value (the dependent variable), x is an independent variable and y is a dependent variable.

8.7 Results and Discussion

The descriptive statistics in this section present the values for computed metrics of ASOG, ASOM, ASOC and ASOU for the research datasets. In particular, table 8-3 shows the descriptive summaries of the minimum and maximum values, as well as the values of central tendency (mean and median), dispersion (standard deviation and variance) for the first dataset which was generated from our transformation experimental process. The following observations can be made from table 8-2:

For the results reported for ASOG, ASOM and ASOU, apart from the ASOC, the standard deviation is small relative to the value of the mean; therefore the mean is a good representation of the data.

- The ASOC has a relatively low mean value and standard deviation. As we have seen in table 8-1 (section 8.5), several cases in the demonstrated scenarios do not have any significant cohesion, which explains why the ASOC metric has low value and variance.
- The ASOU has the largest maximum, which could be due to large dependencies between services with very low levels of granularity. The ASOU also has the largest standard deviation, which confirms there is a wide range of values among cases.
- There are cases of the service design interfaces with no cohesion and coupling among services (ASOC and ASOU have zero minimum). This may reflect having one monolithic service in a service domain with a minimum value of zero in coupling (ASOU), it alternatively refers to lack of cohesion (ASOC) for other cases in a scenario.

Table 8-2 Descriptive statistics - ASOG, ASOM, ASOC and ASOU metrics

Descriptive Statistics						
	N	Minimum	Maximum	Mean	Std. Deviation	Variance
ASOG	15	.277	1.148	.874	.234	.055
ASOM	15	.076	1.408	.855	.380	.145
ASOC	15	.000	.500	.090	.155	.024
ASOU	15	.000	1.600	.680	.508	.258
Valid N (listwise)	15					

8.7.1 Service granularity versus individual quality attributes (H2)

In this section, we investigate the second hypothesis that is concerned with the relationships between the service granularity variable (ASOG) and other internal architectural service quality attributes such as complexity (ASOM), cohesion (ASOC) and coupling (ASOU). We used linear and nonlinear regression analysis to investigate the relationship between ASOG as the dependent variable and all quality attributes as independent variables (ASOM, ASOC and ASOU). The study uses statistical tests with the dataset from our framework. For this study, the ASOG presents the independent variable and the ASOM, ASOC and ASOU are independent variables.

8.7.1..1 Service Granularity versus Complexity (H2:A)

To test the sub-hypothesis that high value of service granularity (ASOG) would correspond with positive effect on the complexity quality attribute (ASOM) a simple linear regression was performed. Tests indicated that a linear relationship between service granularity (ASOG) and service complexity (ASOM) for our framework's dataset. The coefficient table is shown for the dataset below (related statistical tables are provided in appendix C).

Using the framework dataset: the dataset is normally distributed according to the Shapiro-Wilk's test and the Q-Q plot. Fig. 8-1 shows the best-fit line equation and table 8-3 illustrates the linear regression results as follow:

$$\text{ASOM} = (-.534) + 1.588 * ASOG$$

$$R^2 = 0.953\,,\ R^2_{adj} = 0.950\ , F_{1,13} = 266.375\,, \text{P} - \text{vaule} < 0.05$$

- The $R^2 = 0.953$ indicates that with probability of about 95%, knowing ASOG would predict ASOM.
- The constant's value a = -0.534 estimates the value of service complexity (ASOM) with a zero value of service granularity (ASOG). The interpretation of this particular constant variable with negative value (-.534) is not meaningful since the complexity attribute does not have a negative value. However, this negative value refers to the relationship between the ASOG and ASOM and can still be used for computations with the value of the ASOG variable.
- The regression coefficient's value is 1.588, which represents changes in the value of ASOM when a change taken place in the value of ASOG. In other words, the average level of service complexity increases by

1.588 corresponding to an increase of 1.00 in the value of average level of service granularity. It indicates that there is a positive effect based on intercept = 1.588, thus ASOG has a positive effect on ASOM.

Table 8-3 Simple linear regression coefficients for ASOG dependent and ASOM independent variables for the framework dataset

Coefficients[a]						
Model		Unstandardised Coefficients		Standardised Coefficients		
		B	Std. Error	Beta	t	Sig.
1	(Constant)	-.534	.088		-6.069	.000
	ASOG	1.588	.097	.976	16.321	.000
a. Dependent Variable: ASOM						

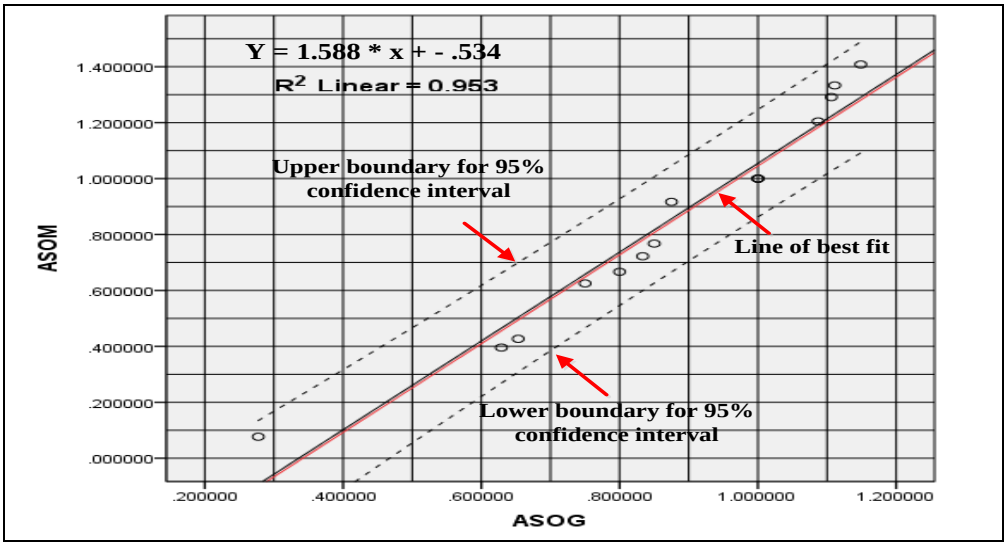


Figure 8-1 Linear regression results of ASOM and ASOG variables from the framework dataset

8.7.1..2 Service Granularity versus Cohesion (H2:B)

To test the sub-hypothesis that a high value of service granularity (ASOG) would correspond with positive effect on the cohesion quality attribute (ASOC) simple linear regression was performed. Tests indicated a nonlinear relationship between service granularity (ASOG) and service complexity (ASOC) using the cubic regression model for our framework's dataset. The coefficient table is shown for the dataset below (related statistical tables are provided in appendix C).

Using the framework dataset: the dataset is not normally distributed poor fit for the current data according to Shapiro-Wilk's test and the Q-Q plot,

ASOC's $P\text{-}value = .000 < 0.05$. The cubic test was selected because it showed a higher ($R^2 = 0.741$) value and gave closest data points to the regression equation. Fig. 8-2 shows the nonlinear equation and table 8-4 illustrates the nonlinear regression results using a cubic model as follows:

$$\text{ASOC} = -2.0573 + 11.566 * \text{ASOG} + -16.810 * \text{ASOG} * \text{ASOG} + 7.267 * \text{ASOG} * \text{ASOG} * \text{ASOG}$$

$$R^2 = 0.741\,, \;\; , F_{1,13} = 10.48\,, P - vaule < 0.05$$

- The $R^2 = 0.741$ indicate that there is probability of 74% that knowing ASOG would predict ASOC.
- The constant's value a $= -2.0573$ estimates that the value of service cohesion (ASOC) with zero value of service granularity (ASOG). This can be used for computations with the value of the ASOG variable. With two bends on the figure 8-2, we can conclude that there is a positive effect based on $\beta 1 = 11.566$ (a positive value), thus ASOG has a positive effect on ASOC.
- The F test indicates that the variability between ASOC and ASOG is statistically significant with $F_{1,13} = 10.48$, P-value < 0.05.

Table 8-4 Nonlinear regression model summary using cubic test for ASOC and ASOG variables on the framework dataset

Equation	Model Summary					Parameter Estimates			
	R Square	F	df1	df2	Sig.	Const.	b1	b2	b3
Cubic	0.741	10.48	3	11	.001	-2.057	11.566	-16.810	7.267

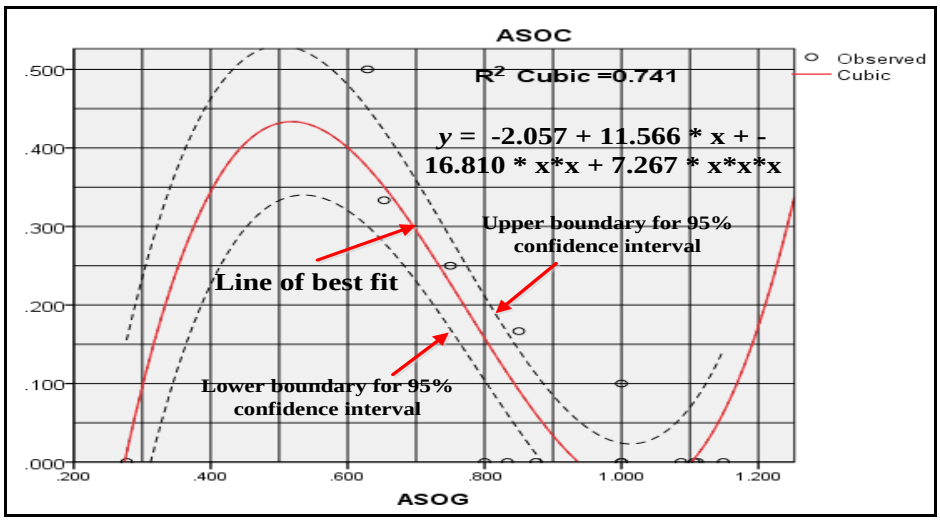


Figure 8-2 Nonlinear regression results of ASOC and ASOG variables using the Cubic regression model for the framework dataset

8.7.1..3 Service Granularity versus Coupling (H2:C)

To test the sub-hypothesis that a high value of service granularity (ASOG) would correspond with negative effect on the coupling quality attribute (ASOU) simple linear regression was performed. Tests indicated that a linear relationship between service granularity (ASOG) and coupling (ASOU) for the framework dataset. The coefficient table is shown for the dataset below (related statistical tables are provided in appendix C).

Using the framework dataset: the dataset is normally distributed on the basis of the Shapiro-Wilk's test and the Q-Q plot. Fig. 8-3 shows the best-fit line equation and table 8-5 illustrates the linear regression results as follow:

$$\text{ASOU} = (-.906) + (1.813) * ASOG$$

$$R^2 = 0.697\,,\ R^2_{adj} = 0.673\ ,F_{1,13} = 29.854\,,\text{P} - \text{vaule} < 0.05$$

- The $R^2 = 0.697$ indicates that there is probability of 70% that knowing ASOG would predict ASOU.
- The constant's value a = -0.906 estimates that the value of ASOU with a zero value of ASOG. The interpretation of this particular constant variable with negative value (-.0.906) is not meaningful since the coupling attribute does not have a negative value.
- The regression coefficient's value is 1.813, which represents changes in the value of ASOU when a change is accounted in ASOG. In other words, the average level of ASOU increases by 1.588 corresponding to

an increase of 1.00 in the value of average level of ASOG. It indicates that there is a positive effect based on intercept = 1.813, thus ASOG has a positive effect on ASOU.

Table 8-5 Linear regression model summary for ASOU and ASOG variables on the framework dataset

Coefficients[a]					
	Unstandardised Coefficients		Standardised Coefficients		
Model	B	Std. Error	Beta	t	Sig.
(Constant)	-.906	.300		-3.022	.010
ASOG	1.813	.332	.835	5.464	.000

a. Dependent Variable: ASOU

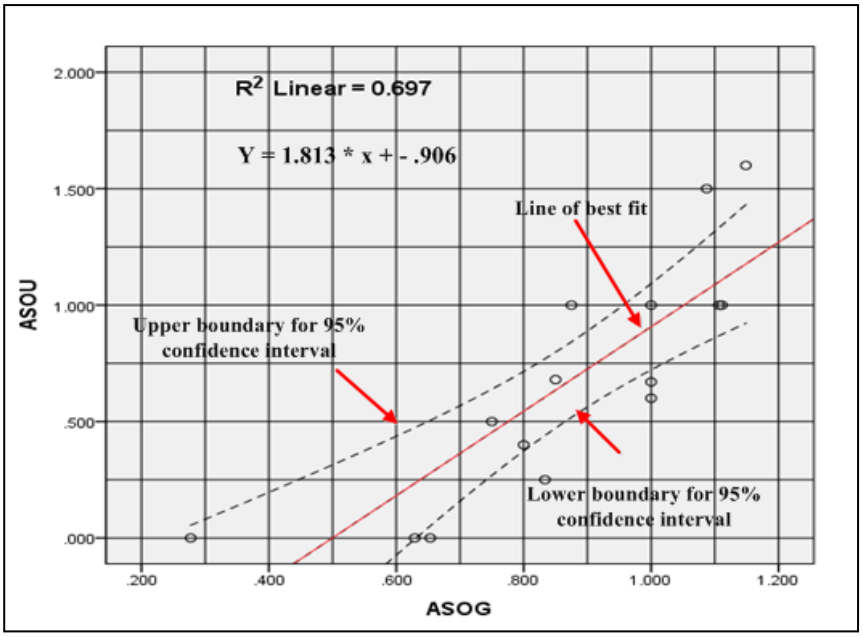


Figure 8-3 Linear regression chart of ASOU and ASOG variables on the framework dataset

8.7.2 Relationships between quality attributes (H3)

In this section, we investigated the third hypothesis that is concerned with the relationships between the service qualities of different attributes such as complexity (ASOM), cohesion (ASOC) and coupling (ASOU). We used correlation coefficient to investigate the relationship between the (ASOM), (ASOC) and (ASOU), two variables at time. The investigation is based on statistical tests using the dataset from our framework. The classifications of the

relationships (e.g., strong positive, weak positive) between variables are defined based on the definitions in section 8.6.2.

8.7.2..1 Complexity versus Cohesion (H3:A)

A correlation test was performed to test the sub-hypothesis that the architectural quality attributes of complexity (ASOM) and cohesion (ASOC) are correlated. Spearman's correlation coefficient (rs) or Pearson's correlation coefficient (r) tests are conducted depending on the normality of the dataset. The correlations coefficient table is shown below for the dataset as follows:

Using the framework dataset: the dataset is not normally distributed on the basis of the Shapiro-Wilk's test and the Q-Q plot, thus Pearson's correlation coefficient (r) test was conducted. Table 8-6 illustrates the correlation coefficient results as follow:

- The correlation coefficient (rs = -0.533) showed that the relationship between ASOM and ASOC is a strong negative relationship and is statistically significant (P-value = 0.041 < .05). The result of (rs) suggests that an increase in ASOM results in a large decrease in the value of ASOC, and vice versa.

Table 8-6 The Spearman's rho for ASOM and ASOC variables from the framework dataset

			ASOM	ASOC
Spearman's rho	ASOM	Correlation Coefficient	1.000	-.533*
		Sig. (2-tailed)	.	.041
		N	15	15
	ASOC	Correlation Coefficient	-.533*	1.000
		Sig. (2-tailed)	.041	.
		N	15	15

*. Correlation is significant at the 0.05 level (2-tailed).

8.7.2..2 Complexity versus Coupling (H3:B)

A correlation test was performed to test the sub-hypothesis that the architectural quality attributes of complexity (ASOM) and coupling (ASOU) are correlated. Spearman's correlation coefficient (rs) or Pearson's correlation coefficient (r) tests are conducted depend on the normality of the dataset. The correlations coefficient table is shown below for the dataset as follows:

Using the framework dataset: the dataset is normally distributed on the basis of the Shapiro-Wilk's test and the Q-Q plot, thus Pearson's correlation coefficient (r) test was conducted. Table 8-7 illustrates the correlation coefficient results as follow:

- The Pearson correlation coefficient (r =0.895) showed that the relationship between ASOM and ASOU is a very strong positive relationship and is statistically significant (P-value = 0.000 < .05). The result of (r) suggests that the increase in ASOM might also cause a large increase in the value of ASOU, and vice versa.

Table 8-7 The Pearson (r) test for ASOM and ASOU variables from the framework dataset

		ASOM	ASOU
ASOM	Pearson Correlation	1	.895**
	Sig. (2-tailed)		.000
	N	15	15
ASOU	Pearson Correlation	.895**	1
	Sig. (2-tailed)	.000	
	N	15	15
**. Correlation is significant at the 0.01 level (2-tailed).			

8.7.2..3 Coupling versus Cohesion (H3:C)

A correlation test was performed to test the sub-hypothesis that the architectural quality attributes of coupling (ASOU) and cohesion (ASOC) are correlated. Spearman's correlation coefficient (rs) or Pearson's correlation coefficient (r) tests are conducted depend on the normality of the dataset. The correlations coefficient table is shown below for the dataset as follows:

Using the framework dataset: the dataset is not normally distributed on the basis of the Shapiro-Wilk's test and the Q-Q plot, thus the Spearman's correlation coefficient (rs) test was conducted test was conducted. Table 8-8 illustrates the correlation coefficient results as follow:

- The correlation coefficient (rs = -0.525) showed that the relationship between ASOC and ASOU is a strong negative relationship and is statistically significant (P-value = 0.044). The result of (rs) suggests that the increase in ASOC might cause a large decrease in the value of ASOU, and vice versa.

Table 8-8 The Spearman's rho for ASOC and ASOU variables from the framework dataset

Variable	Test	ASOC	ASOU
ASOC	Spearman'srho Correlation Coefficient	1.000	-.525
	Sig. (2-tailed)		.044
	N	15	15
ASOU	Spearman'srho Correlation Coefficient	-.525	1.000
	Sig. (2-tailed)	.044	
	N	15	15

8.8 Reflection on Research Hypotheses

In this section, we will conduct an analysis of the results from the empirical evaluation to accept or reject the second (***H2)*** and third (***H3)*** research hypotheses addressed in (section 8.7). During our empirical evaluation, we are interested in the second and the third hypotheses as follows:

In order to accept or reject the hypotheses, we define a preselected significance level equal to .05 and consider the following elements in selecting the testing methods:

1. With Pearson (r) and Spearman Rho correlation relationships, we calculated the test statistic using this formula, $T = \frac{r\sqrt{n-2}}{\sqrt{1-r^2}}$, and observed the T-critical value using the t-table distribution. When T value > T-critical value, we reject the null hypothesis(H_0), otherwise, the (H_0) is accepted.
2. With linear regression models, we used the t-test to investigate whether the slope $\beta 1$ (regression coefficient) of the regression line differs significantly from zero. The t-score is used to compensate for standard error because our sample data size (n) is less than 30 samples.
3. With nonlinear regression (polynomial), we used the F-test to test the null hypothesis $\beta 1 = \beta 2 = \beta 3 = 0$ and to investigate the significance of relationships between independent and dependent variables. The F-test is used to check the significance of the regression within preselected a significance level alpha $(1 - \alpha)$.

8.8.1 Impact of granularity on quality attributes (H2)

In this section, we discuss the analysis results of the second hypothesis concerned with the impact of service granularity on the other quality attributes of complexity, cohesion and coupling. This presents the impact based on the dataset generated from our framework using regression analyses. The second hypothesis discussion is divided into three sub-hypotheses, as follows:

H2:A: *the impact of service granularity (ASOG) on complexity (ASOM).*

H2:B: *the impact of service granularity (ASOG) on cohesion (ASOC).*

H2:C: *the impact of service granularity (ASOG) on coupling (ASOU).*

8.8.1..1 Impact of Service Granularity on Complexity (H2:A)

Null Hypothesis $(\boldsymbol{H_0})$: *a high value of ASOG has* ***no*** *effect on ASOM.*
Alternative Hypothesis $(\boldsymbol{H_a})$: *a high value of ASOG has a positive effect on ASOM.*

Using the framework dataset: the null hypothesis $(\boldsymbol{H_0})$ is rejected because we noticed that there is a statistically significant difference in the distributions $F_{1,13} = 266.375$, $\mathrm{P-vaule} < 0.05$. Thus, this linear equation can assist in predicting the service complexity (ASOM). A high value of service granularity (ASOG) would correspond with effect on the complexity quality attribute (ASOM).

We calculated the t-score in order to check the credibility of our model and derived the T-score =16.37 (t= β1/SE = 1.588/0. 097) (based on the slope β1 being equal to 1.588, the standard error (SE) = .097, and the degree of freedom (df) =14 (based on our dataset df = n - 15) (table 12)). We used the t-table distribution to determine the two-tailed P-value (t > 16.37) = 0.0001. By conventional criteria, this difference is considered to be statistically significant. Thus, the P-value (0.0001) is less than the significance level (0.05); so we can reject the null hypothesis. That is, at 95% confidence interval of this difference is from 1.379 to 1.796.

The alternative hypothesis $(\boldsymbol{H_a})$ is accepted. The linear equation shows a positive value for the regression coefficient (intercept), this means a positive effect and confirms our hypothesis (H2.A) that states ASOG has a positive effect on ASOM.

Figure 8-4 shows three examples with different numbers of service operations and granularity scales; we note that the overall number of operations in all services in the domain affects the relationship between the average service granularity and complexity of domain services. For example, the increase of service granularity (ASOG) in figure 8-4.A, B and C (with low numbers of service operations 9, 5 and 5, respectively) causes a slight decrease/increase in the complexity. This result may also occur because the ASOM and ASOG metrics were driven based on the calculated value of Service Operation Granularity (SOG).

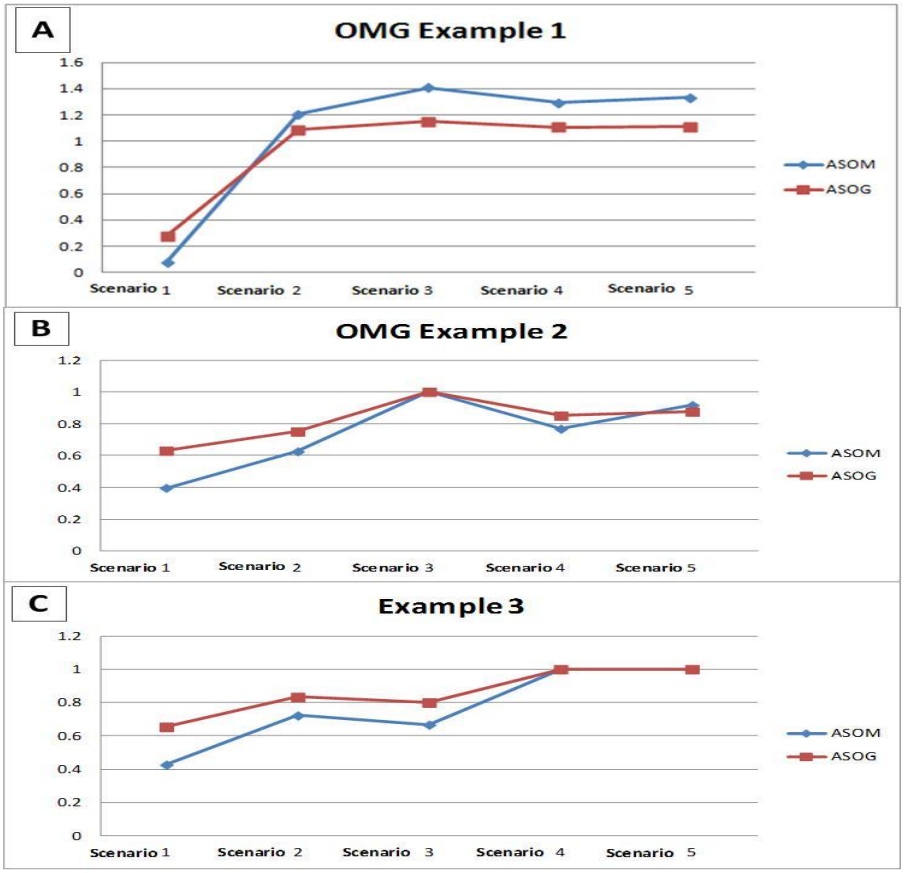


Figure 8-4 The relationship between Granularity (ASOG) and Complexity (ASOM)

8.8.1..2 Impact of Service Granularity on Cohesion (H2:B)

Null *Hypothesis* $(\boldsymbol{H_0})$: *a high value of ASOG has* ***no*** *effect on ASOC.*
Alternative *Hypothesis* $(\boldsymbol{H_a})$: *a high value of ASOG has a positive effect on ASOC.*

Using the framework dataset: The null hypothesis $(\boldsymbol{H_0})$ is that $\beta 1 = \beta 2 = \beta 3 = 0$ is rejected because we noticed that there is a statistically significant

difference in the distributions $F_{1,13} = 10.48, P - vaule < 0.05$. Thus, the nonlinear equation can assist in predicting the service cohesion (ASOC). A high value of service granularity (ASOG) would correspond with effect on the cohesion quality attributes (ASOC).

The computed F value = 10.48, which gives a P-value of 0.001. This means that there is a 99.9% chance that there is a significant difference in the data. From the F distribution table, we can see the critical point value with 3 degrees of freedom, error at 11 degrees of freedom and $\alpha = 0.05$ is 3.59 ($F = \mu_1, \mu_2, \alpha = 3, 11, .05 = 3.59$), so we reject the null hypothesis because computed F > critical point F and the result we derived based on the framework dataset is significant. We are 95% confident that there is significant variance between both ASOG and ASOC variables.

The alternative hypothesis ($\boldsymbol{H_a}$) is accepted. The nonlinear equation shows a positive value for the $\beta 1$ value, this means a positive effect and confirms our hypothesis (H2.B) that states ASOG has a positive effect on ASOC.

The relationship patterns that can be defined between ASOC and ASOG in the small-scale service domain are that the increase of service granularity (ASOG) will result either in decrease or no change in cohesion (ASOC). For example, figure 8-5.B and C indicated that the increase in the value of ASOG results in a decrease in ASOC at different level of granularity; when the ASOG value increases by 0.224, the ASOC value decreases by 0.123. Although we were able to find a mathematical relationship between ASOG and ASOC, we found that there are other factors that might affect the cohesion measurement more than the granularity factor. When there is no cohesion among operations or exchanged messages in a set of services, re-factoring the service to produce a different service granularity will have no effect. For example, in figure 8-5.A, the value of ASOC (cohesion) was equal to zero regardless of the change of the level of granularity, while the value of ASOG fluctuated between 0.27 and 1.14. We can conclude that the result of the F-test and the graphical chart is that the ASOG variable can be used to predict the level of cohesion in a set of services.

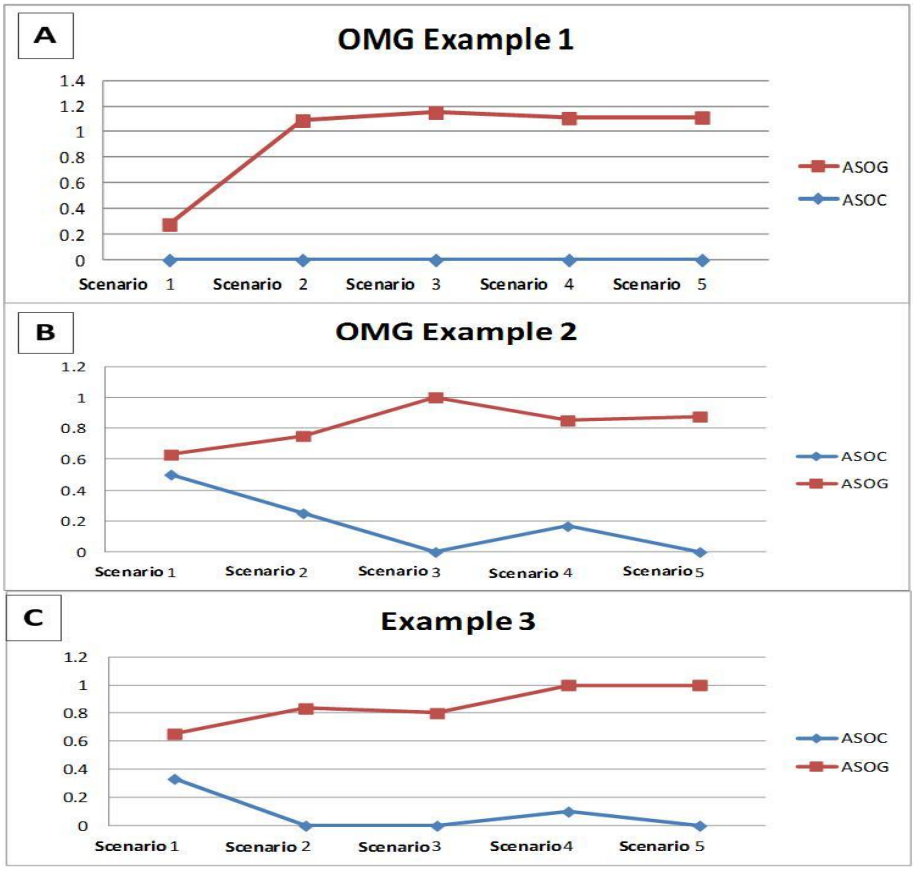


Figure 8-5 The relationship between Granularity (ASOG) and Cohesion (ASOC) variables

8.8.1..3 Impact of Service Granularity on Coupling (H2:C)

Null *Hypothesis* $(\boldsymbol{H_0})$: *high value of ASOG has* ***no*** *effect on ASOU*
Alternative *Hypothesis* $(\boldsymbol{H_a})$: *high value of ASOG has a negative effect on ASOU.*

Using the framework dataset: The null hypothesis $(\boldsymbol{H_0})$ is rejected because we noticed that there is statistically a significant difference in the distributions $F_{1,13} = 29.854$, $\mathrm{P-vaule} < 0.05$. Thus, this linear equation can assist in predicting the service coupling (ASOU). A high value of service granularity (ASOG) would correspond with effect on the quality attributes coupling (ASOU).

The t-score computed value is 5.46 (t= β1/SE = 1.813/0.332), based on the slope β1 being equal to 1.813, the standard error (SE) = .332, and the degree of freedom (df) =14 (based on our dataset df = n -15) (table 13). We used the t-table distribution to determine the two-tailed P-value (t > 5.46) =

0.0001. Based on conventional criteria, this difference is considered to be statistically significant. Thus, the P-value (.0001) is less than the significance level (0.05); so we can reject the null hypothesis. (At 95% confidence interval of this difference is from 1.10093 to 2.52507).

The alternative hypothesis $(\boldsymbol{H_a})$ is rejected. The linear equation shows a positive value for the regression coefficient (intercept), this means a positive effect and disconfirms our hypothesis (H2.C) that states ASOG has a negative effect on ASOU.

The increase in the level of service domain granularity (ASOG) might increase the value of coupling (ASOU) among services with different degrees. In figure 8-6.B, the value of coupling increases from zero to 0.5 unit corresponds to the increase in granularity by 0 .12 unit in scenario 2, where it increases to 1 unit corresponding to the increase in granularity by 0.25 unit in scenario 3. However, we found that operations with similar behaviour that re-factored within new services processing different data size and types minimized the dependencies among services because reduce invocation of other services. The dependencies among services might reach a point where further granularity has no effect; for example, in figure 8-6.A (scenarios 3 and4) the coupling value did not change despite the increase in the granularity value.

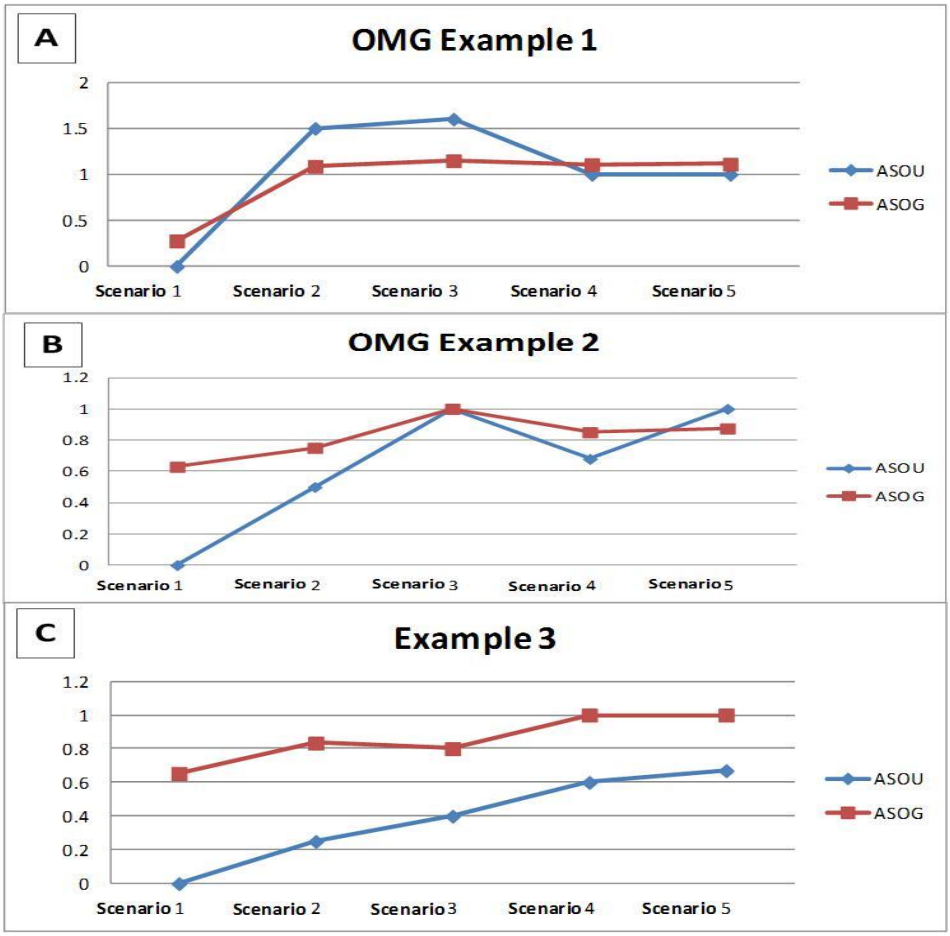


Figure 8-6 The relationship between Granularity (ASOG) and Coupling (ASOU) variables

8.8.2 Dependencies between Quality attributes (H3)

In this section, we discuss the analysis results of the third hypothesis concerned with dependencies between quality attributes of complexity, cohesion and coupling. This presents relationships between attributes using correlation coefficient on the dataset generated from our framework. The third hypothesis discussion can be divided into three sub-hypotheses, as follows:

H3:A: *the architectural quality attributes of complexity (ASOM) and cohesion (ASOC) are correlated.*

H3:B: *the architectural quality attributes of complexity (ASOM) and coupling (ASOU) are correlated.*

H3:C: *the architectural quality attributes of cohesion (ASOC) and coupling (ASOU) are correlated.*

8.8.2..1 Correlation of Complexity and Cohesion (H3:A)

Null *Hypothesis* $(\boldsymbol{H_0})$: *ASOM and ASOC has no association.*
Alternative *Hypothesis* $(\boldsymbol{H_a})$: *ASOM and ASOC are correlated.*

Using the framework dataset: the null hypothesis $(\boldsymbol{H_0})$ is rejected because we observed that the T = 2.271 which is greater than the T-critical value of 2.160, thus there is a statistically small significant association between service complexity (ASOM) and service cohesion (ASOC). The statistic is computed as follow:

- $T = \frac{rho\sqrt{\mathrm{n}-2}}{\sqrt{1-rho^2}} = \frac{-0.533\sqrt{15-2}}{\sqrt{1-(-0.533)^2}} = 2.271$
- T-critical value = 2.160 , at $\alpha = .05\ (2-\text{tailed})$, $df = n - 2 = 15 - 2 = 13$,

The alternative hypothesis $(\boldsymbol{H_a})$ is accepted, because the null hypothesis is rejected and there is a significant association between ASOM and ASOC. The difference between the observed T and T-critical value is 0.1. This difference is low which means the association might be an indirect association. This is confirmed by the fact that the p-value equal .041 (close to 0.05) and $r = -0.533$. Nevertheless one of the selected OMG examples has no cohesion among services in the proposed scenarios (figure 8-7 A, B and C).

It is clear that there is a negative relationship between ASOM and ASOC, as an increase in ASOC causes a decrease in ASOM (figure 8-7.A, B and C). Thus, this means there is a correlation between ASOM and ASOC and confirms our hypothesis (H3.A) that states ASOM and ASOC are correlated.

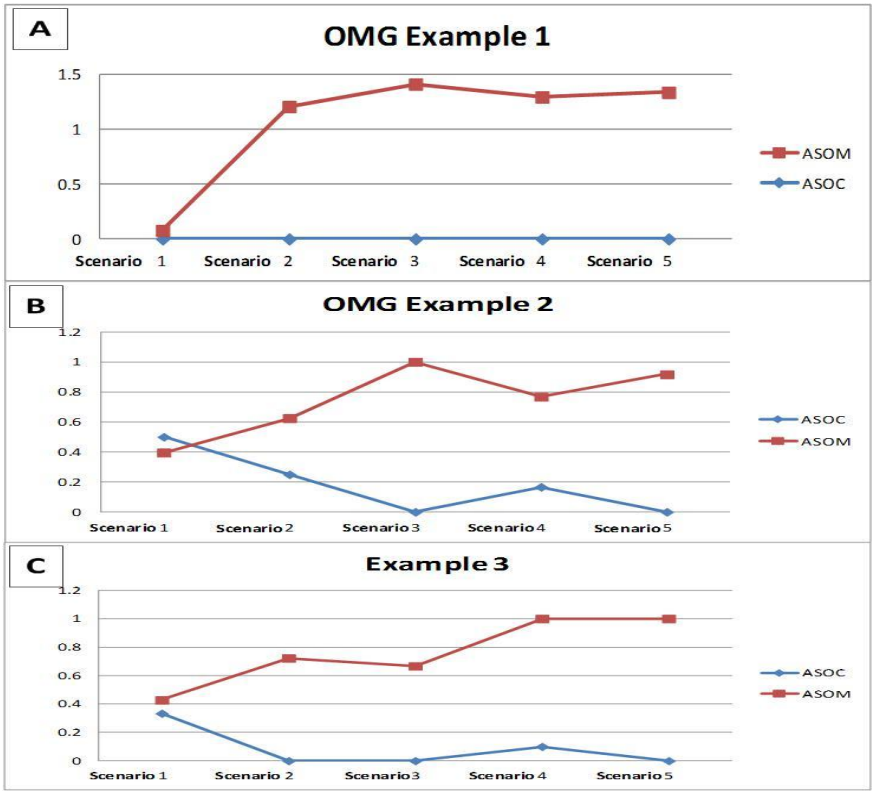


Figure 8-7 The relationship between Complexity (ASOM) versus Cohesion (ASOC)

8.8.2..2 Correlation of Complexity and Coupling (H3:B)

Null *Hypothesis*	$(\boldsymbol{H_0})$: *ASOM and ASOU has no association.*
Alternative *Hypothesis*	$(\boldsymbol{H_a})$: *ASOM and ASOU are correlated.*

Using the framework dataset: the null hypothesis $(\boldsymbol{H_0})$ is rejected because we observed that the T = 7.234 which is greater than the T-critical value of 2.160, thus there is a statistically highly significant association between service complexity (ASOM) and service coupling (ASOU). The statistic is computed as follows:

- $T = \frac{r\sqrt{n-2}}{\sqrt{1-r^2}} = \frac{-0.895\sqrt{15-2}}{\sqrt{1-(-0.895)^2}} = 7.234$
- T-critical value= 2.160 , at $\alpha = .05\ (2-\text{tailed})$, $df = n - 2 = 15 - 2 = 13$.

The alternative hypothesis $(\boldsymbol{H_a})$ is accepted, because the null hypothesis is rejected and there is a significant association. The difference between the observed T and T-critical value is 5.1 which more likely significant. This is confirmed by the fact that the p-value is very small (close to 0.) and $r =$

0.895 . Thus, this means there is a correlation between ASOM and ASOU and confirms our hypothesis (H3.B) that states ASOM and ASOU are correlated.

Based on the P-value, the association is a very strong positive relationship, so high coupling among services seemingly increases the overall complexity. In a service domain with a coarser-grained service, the coupling value will always be zero. In contrast, in a fine-grained set of services, the coupling value will be maximum unless overall complexity is decreased, which implies that there are factors that might affect coupling besides complexity (see figure 8-8.A, B and C). Since coupling and complexity metrics were driven based on different concepts, such a strong relationship implies that controlling coupling is a very important factor to reduce complexity.

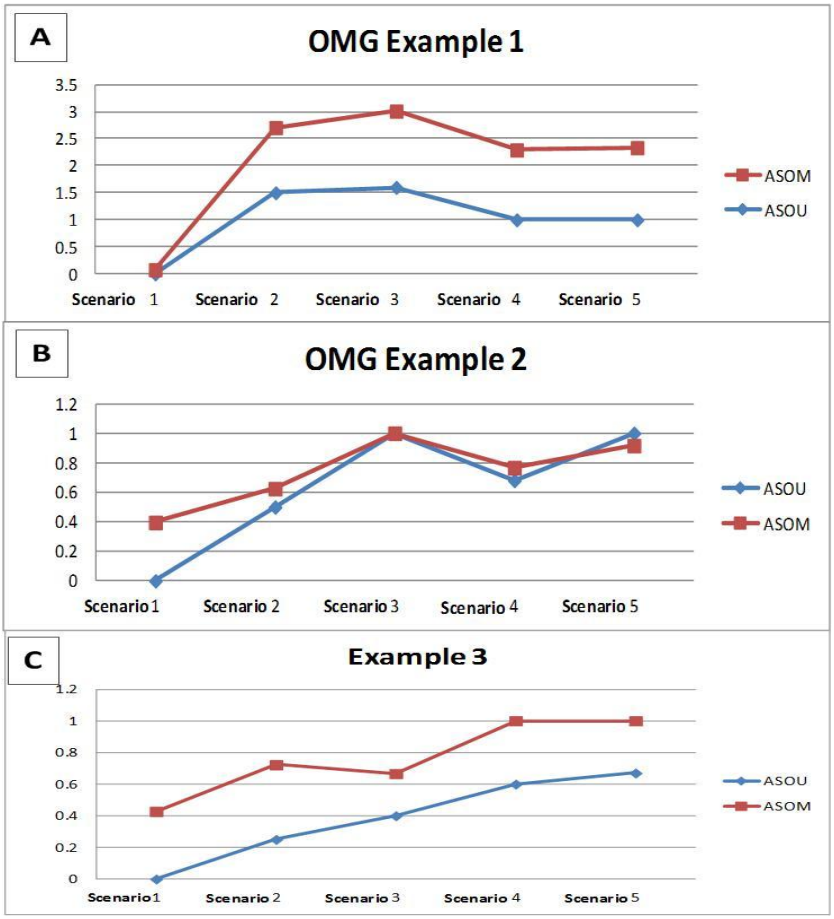


Figure 8-8 The relationship between Complexity (ASOM) versus Coupling (ASOU)

8.8.2..3 Correlation Association of Coupling and Cohesion (H3:C)

Null *Hypothesis*	$(\boldsymbol{H_0})$: *ASOU and ASOC has no association.*
Alternative *Hypothesis*	$(\boldsymbol{H_a})$: *ASOU and ASOC are correlated.*

Using the framework dataset: the null hypothesis ($\boldsymbol{H_0}$) is rejected because we observed that the T = 3.607 which is greater than the T-critical value of 2.160, thus there is a statistically significant association between service coupling (ASOU) and service cohesion (ASOC). The statistic is computed as follows:

- $T = \frac{r\sqrt{\mathrm{n}-2}}{\sqrt{1-r^2}} = \frac{-0.525\sqrt{15-2}}{\sqrt{1-(-0.525)^2}} = 3.607$
- T-critical value= 2.160 , at $\alpha = .05\ (2-\text{tailed})$, $df = n - 2 = 15 - 2 = 13$,

The alternative hypothesis ($\boldsymbol{H_a}$) is accepted, because the null hypothesis is rejected and there is a significant association. However, this accepting result is weak since $rs = -0.525$, with p-value = .044 (close to 0.05). However, this means there is a correlation between ASOU and ASOC and confirms our hypothesis (H3.C) that states ASOU and ASOC are correlated.

We confirm that the relationship between coupling and cohesion is an inverse one. The main difference between the values of coupling and cohesion is that cohesion might always occur among services regardless of the level of granularity, whereas coupling does not occur in a service domain consisting of a coarse-grained service (see scenario 1 in figure 8-9.A, B and C).

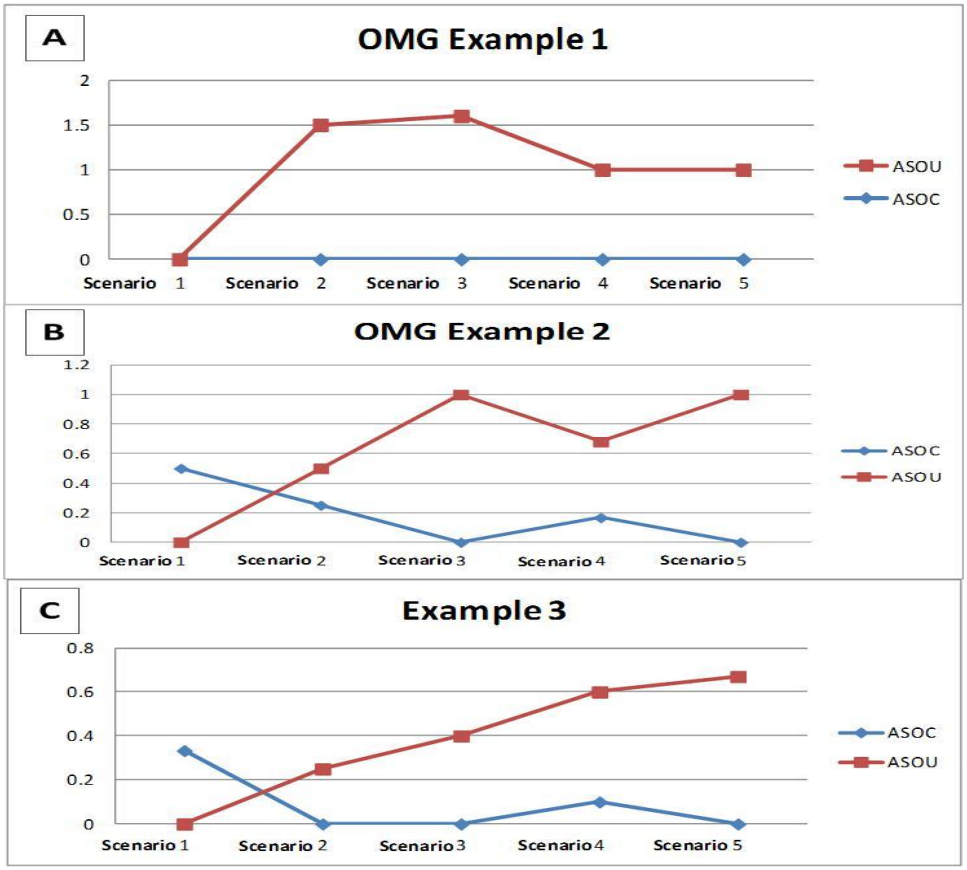


Figure 8-9 The relationship between Cohesion (ASOC) versus Coupling (ASOU)

8.9 Limitations of Empirical Evaluation

8.9.1 Dataset size

The size of the dataset is a limitation on the validity of our study results. To increase the reliability and accuracy of the results of the study, the sample size needs to be sufficiently large. The framework dataset that was collected from the syntax elements of service interfaces and generated using the algorithm described in section 6.3.3. The defined algorithm generates five cases (five different designs of service interfaces) for every WS-CDL document generated from the model transformation. The algorithm could generate 15 cases for our framework's dataset from the three application scenarios, as every case is considered to be as autonomous. We consider these cases as variances of potential service interfaces. In a perfect dataset, these cases might be representing all possible re-factored designs of service interfaces for one scenario, and through the evaluations of all these cases, we are guided to the

most optimal case. The number of all possible cases for a scenario can be calculated by for all possible combinations of choreography tasks in a scenario. The number of all possible combinations can be calculated using the "Bell number", which is the number of possible partitions of a set with n numbers (Yang 1996). For example, for the Incident Management scenario that has nine choreography tasks, thus $n = 9$, and all possible combinations equal results in 21140 subsets. The implementation of such a large number of service interfaces is difficult, and not feasible with the current methodology.

With respect to this issue, we applied non-parametric methods such as Shapiro-Wilk (test normality) and Spearman's correlation coefficient (correction test) whenever possible because they are usually more suitable and effective for a small sample size. Removing potential outliers can increase the accuracy of the result but the existing number of outliers is high particularly in a small data set (15 pair of values). In fact, the focus of the current study is on the relationships and the impact of service granularity (independent) on other architectural attributes (dependent), not to draw conclusions about those particular scenarios. This means that it is reasonable to address this limitation when we tested the second and third research hypotheses.

8.10 Summary

In this chapter, we evaluated empirically the second part of the implementations of our framework which is the service quality model. This study used datasets generated from our framework for the empirical evaluation.

Using regression analysis we were able to find linear/nonlinear relationships between service granularity (ASOG) and other architectural quality attributes of complexity (ASOM), cohesion (ASOC) and coupling (ASOU) within the dataset. For the dataset the relationships were described as three mathematical equations for every dependent variable of ASOM, ASOC and ASOU that can assist in predicting the value of ASOG. The nature of the regression relationship (linear or nonlinear) between ASOG and other quality attributes is variable and depends on the data distribution (normal or non-normal). We found that statistically there are significant variances between ASOG and ASOM, ASOC and ASOU at a 95% confidence interval. It can be observed that the values for all dependent variables (ASOM, ASOC and ASOU) supported the proposed research hypotheses. The dataset confirms

clearly the same trend toward positive and negative directions, apart from the relationship between ASOG and ASOU. The results fall into a relatively close scale; e.g., the relationship between cohesion (ASOC) and complexity (ASOM) within the framework dataset is a strong negative. This supports the second research hypothesis in which the ASOG impacts positively the service quality attributes of ASOM and ASOC and contradicts ASOG has a negative effect on ASOU.

We used correlation analysis to investigate in detail the relationships between the different service quality attributes of ASOM, ASOC and ASOU. This analysis showed that all correlations are statistically significant between these ASOM, ASOC and ASOU attributes with various degrees. These findings emphasise the importance of being able to measure service quality attributes and select trade-offs that suit the underlying requirements of the business. This supports the third research hypothesis in which there are correlated relationships between ASOM, ASOC and ASOU.

In conclusion, the service quality model provides significant evidence with respect to the effect of service granularity on other service quality attributes and confirms previous findings of the relationships between the different attributes of service quality. The model also contributes to the field of service computing and can be used to evaluate the service quality aspects of any service interface design. While, the service quality model has been successfully supporting our hypotheses and the model transformation architecture, identifying the most optimum service interface design is still not possible currently. However, in section 9.2.1, we propose a solution that defines successfully a range of values with optimum values for a service interface design.

Chapter 9 will summarise the research and discuses the potential future work that can be used to extend contributes of this thesis.

Chapter 9 Conclusions and Future Work

This final Chapter concludes the thesis with a review of its contributions to the field of service computing and a presentation of extensions for future work. This Chapter is structured as follows: section 9.1 summarises the details of our research findings and scientific contributions and Section 9.2 explains a number of potential extensions for development based on the framework developed.

9.1 Research Summary

The main objective of this thesis was to identify the optimum services for a service-oriented system. In this thesis, the appropriate (optimum) services refer to services with the precise level of granularity that balances the trade-off between service quality attributes according to the user-system requirements. While there has been intensive research in service-oriented systems, the service identification process is still implicit in the service development cycle, with no solid methodology or service quality measurements. The definitions of a service need to consider different architectural levels, which results in an abstraction gap. One feasible solution is to develop a methodology that identifies services using a model-driven approach (MDA) with a special focus on service granularity and service quality attributes.

The integration of SOA and MAD enables us to establish a new theoretical base of the choreography concept for generating service interface designs from a business process model. In order to bridge the abstraction gap, several challenges need to be resolved. An analysis of the current research in this field determined that most of the current approaches focus on analysis techniques (such as clustering) to fill the abstraction gap. This isolates services

implementation from the business process modelling and does not consider the importance of measuring service quality attributes and granularity. In our thesis, we bridged the gap by automatically transforming the business process choreography to service choreographies and then generated the service interface designs (potential service implementations), using the service choreography code to generate the service interface that contrasts the traditional practice of generating service choreography from service interfaces in WSDL.

The chain of transformation programs in ATL used the semantics of the service choreographies (WS-CDL) as a mediator to link the semantics of the business process models (BPMN 2.0) and the service interface design (WSDL) chain to automatically generate service interface designs (WSDL) from a business process model (BPMN 2.0) using the service choreographies. To realise this chain of transformation, the semantics of source and target models must be compatible. In Chapter 4, we introduced a new extension for BPMN 2.0 specifications to facilitate the mapping between BPMN 2.0 and WS-CDL, the extension was used in the BPMN meta-model for modelling transformation and in the service quality model for metrics computations. The meta-models of WS-CDL and WSDL are defined and supported by developing the theory of potential usage of choreography in model transformation. We demonstrated the transformation implementation using three application examples that showed service choreography (WS-CDL) can be used to enable the transformation from BPMN 2.0 standards to WSDL. In chapter seven, we showed pragmatically that the transformation between source and target models generated are valid XML files and that the consistency of semantics and behaviour was satisfied throughout the transformation chain.

It is worth noting that our vision (in 2009, at the early stage of the PhD research) of the importance of choreography in the context of business process modelling was confirmed when OMG BPMN 2.0 included choreography specifications as a new element in the BPMN 2.0 standard (BPMN 2.0 was released in January 2011).

The service quality model is essential for quantifying the service granularity and quality attributes that affect the selection of the optimum service interface design. Existing research in service quality models ignores the measurements for service granularity; despite the importance of the service granularity issue in service design being addressed. In Chapter 5, we developed a service quality model based on a definition of service granularity as a metric,

by which three metrics for internal service quality attributes of complexity, cohesion and coupling are derived. In Chapter 8, the metrics were computed using a generated dataset from our three application examples. We found that there is a relationship between service granularity and other internal architectural quality attributes of complexity, cohesion and coupling for both datasets and there are statistically significant correlations between these ASOM, ASOC and ASOU attributes. The empirical findings in this thesis provide the ability to quantify important factors in service design, such as service granularity, and confirm our understanding of the relationships between attributes of service quality.

As a summary, the main contributions of our research work include:

- Model transformation software was developed to generate a service interface design (WSDL) automatically from business process model (BPMN 2.0) using service choreographies (WS-CDL). Required meta-models to bridge the semantic gaps are described.
- A service quality model was developed to provide metrics for measuring the service granularity and service quality attributes of complexity, cohesion and coupling. The service quality model can be used to select an optimum service interface design for a set of services. We developed theories of these metrics based on our understanding and existing literature in software quality measurements. We provided a measurement for service granularity that can be enhanced to include additional factors in the future.
- The integration of the implementations of model transformation and service quality model can be used to deliver an optimum service interface designs as shown in future work 9.2.1

In conclusion, this thesis discusses a framework that automatically generates an optimum service interface design from a business process model based on service choreography using model-driven technology and provides a quality model for quantifying service quality attributes to reach the optimum service interface design at an early stage, and in this way, contributes to the field of the model-driven development of service modelling. We found that identifying one optimum service interface design is not possible. However we were successful in defining a range of values mathematically that generate optimum values for a service interface design. This framework improves the productivity of SOA development by automating traditional service-oriented

development, integrating the service quality assurance within the development cycle, and increasing the robustness of developing service interface concerning service granularity. Although some limitations remain before the framework can be applied generally to Service Computing applications, we believe the framework proposed, designed, implemented, and evaluated in this thesis presents an important step in the modelling of service-oriented systems.

9.2 Future Work

This thesis can be extended in a variety of ways. The primary extension of this work would be to identify the optimum service interface designs accurately using the mathematical equations generated via the service quality model. Another extension would be to develop a robustness digital dashboard that integrates the chain of transformation programs with the service quality model. A third extension would be having access to a large-size of data by enhancing the extraction mechanism and re-factoring algorithm.

9.2.1 Finding Optimum Service Interface Designs

The optimum service design with the appropriate level of granularity that balances trade-offs between the service quality attributes of complexity, cohesion and coupling can be achieved using mathematical relationships. Since we derived the mathematical relationships between the service granularity factor and each service quality attribute individually, it is possible to find an intersection point that satisfies different linear and nonlinear equations of the quality attributes. The graphical method is used to specify the pair of points where those simultaneous equations intersect and are satisfied. We assume that the derived mathematical linear and nonlinear equations in the previous chapter are valid equations and the best fit for one example.

Figure 9-1 shows three mathematical relationships for a particular service interface of a set of services. The X-axis represents the values of the service granularity and the Y-axis represents the values of the service quality attributes. The intersection points between the equations presented coordinates that correspond to a unique pair of values through a point (x, y). We have the three equations, two linear and one nonlinear, with four unknown variables: ASOG, ASOM, ASOC and ASOU. The optimum service design with the appropriate level of granularity (ASOG) should have high cohesion (ASOC),

low coupling (ASOC), and low complexity (ASOM). In figure 9-1, the three nonlinear/linear equations do not intersect at a unique point; therefore, no pair of values (x, y) exists that might satisfy all three equations simultaneously. According to our problem space, we only consider the pair of points located in the top right quadrant, where both x and y are positive values because our quality attributes always must have positive values. In earlier Chapter 8, we derived the three equations for complexity, cohesion and coupling as follows:

- Complexity (ASOM) = (-0.534) + 1.588 * ASOG
- Cohesion (ASOC) = (-2.0573) + (11.5661) * ASOG + (-16.81) * ASOG * ASOG + (7.2671) * ASOG * ASOG * ASOG
- Coupling (ASOU) = (-0.906) + (1.813) * ASOG

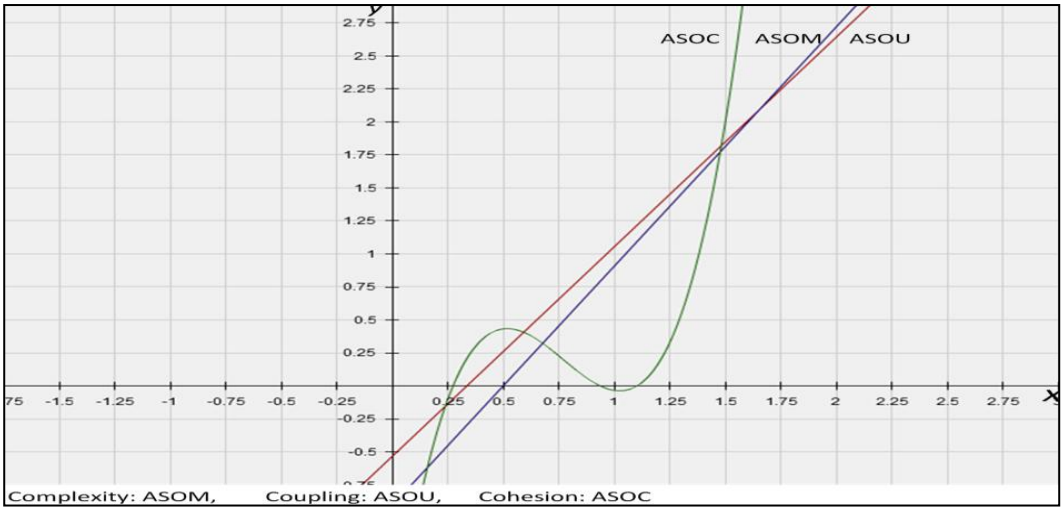


Figure 9-1 Graph of three linear/nonlinear equations: Complexity, Coupling, and Cohesion

The appropriate values for different quality attributes compared to the service granularity values are defined as important pairs of coordinates. It is not possible to find one point where all three equations intersect. Nevertheless, good service design ideally aims to minimize the values of complexity and coupling and maximize the value of cohesion. To simplify the demonstration, we will present firstly complexity and cohesion attributes against the service granularity and then add the coupling attribute to the chart.

Figure 9-2 shows two intersecting points (dots-line) between the linear complexity equation and nonlinear cohesion equation. The first point (1.48, 1.82) shows that the level of service granularity (1.48) is where we can achieve maximum values for complexity and cohesion at 1.82. In contrast, the second point (0.59, 0.41) shows that the level of service granularity that is equal to

0.59 is where we can reduce values of complexity and cohesion to 0.41. In order to select the optimum level of service granularity, we need to either select between high cohesion and a high level of complexity at the first point, or else accept a low level of cohesion with low complexity.

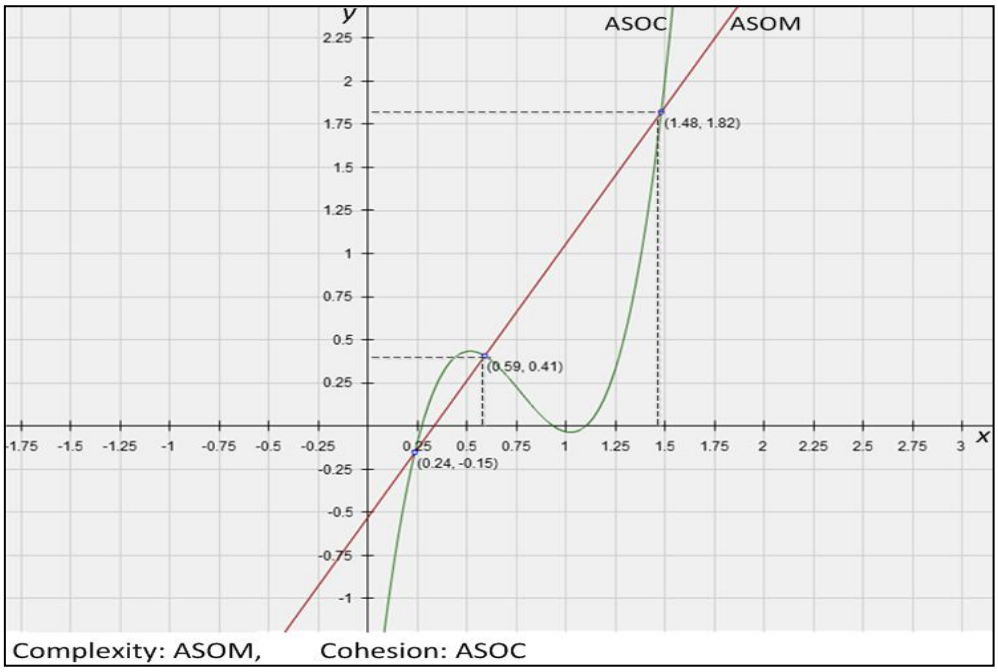


Figure 9-2 Graph of three linear/nonlinear equations: Complexity and Cohesion attributes

To complete the study of all intersected points by adding the coupling attribute to the coordinates, there are three new intersecting points (blue dots) between couplings, cohesion, and complexity on the y-axis, and service granularity on the x-axis (Figure 9-3). Point (1.65, 2.09) represents the worst service design scenario: high values of coupling and complexity equal to 1.65 at the fine-grained level of service granularity at 2.09. Point (1.48, 1.82) shows that at the level of service granularity that is equal to 1.48 we can achieve high values of coupling where coupling is equal to 1.82. However, at the same level of service granularity (1.48), the intersection of cohesion and complexity also shows high complexity. Finally, point (0.68, 0.32) represents the service level of service granularity equal to 0.32 where we can obtain minimum values of coupling and cohesion. Consequently, we can infer from figure 9-3 that the optimum service interface design is located in the grey area, representing the values of the best levels of service granularity (between 0.59 and 1.48) and quality attributes (between 0.32 and 1.82).

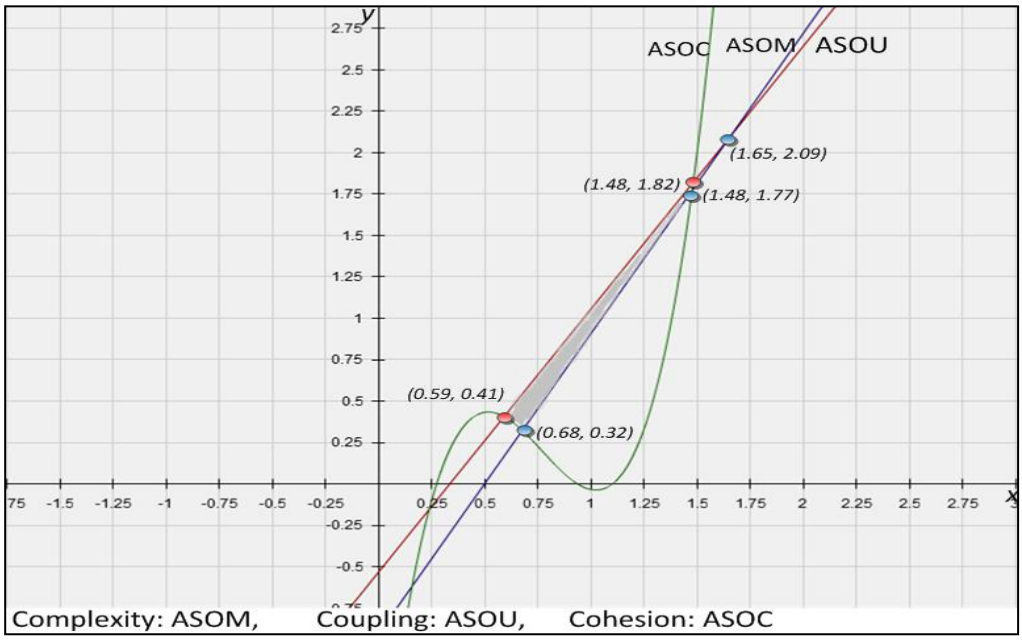


Figure 9-3 intersected points of three linear/nonlinear equations: Complexity, cohesion and coupling attributes

We will assume that the 15 design scenarios generated for different OMG examples represent the service interface design for a similar set of services. Now, we can use these intersected points to evaluate scenarios of the service interface design generated based on our metric models table (Table 1). As shown in Table 1, scenario 6 reports the optimum service interface design result that balances quality attributes of complexity, cohesion and coupling. This provides the ideal solution for this example even with the value of coupling equal to zero, suggesting that the right level of granularity for that service set is 0.62906. In general, the results presented in this section further highlight the promise of integrating quality metrics at the service modelling phase and shows our experimental process can provide accurate results.

Table 9-1 Generated datasets for different scenarios of an OMG example based on the quality metrics

Scenarios	ASOG	ASOM	ASOC	ASOU
1	0.722	0.076	0	0
2	1.087	1.204	0	2
3	1.148	1.408	0	1.6
4	1.106	1.291	0.142	1
5	1.111	1.333	0	1
6	**0.629**	**0.395**	**0.5**	**0**
7	0.75	0.625	0.25	1
8	1	1	0	0
9	0.850	0.767	0.166	0.333
10	0.875	0.916	0	1
11	0.653	0.426	0.333	0
12	0.833	0.722	0	0.25
13	0.8	0.666	0	0.4
14	1	1	0.1	0.6
15	1	1	0	0.5

9.2.2 An Intelligent Digital Dashboard

The objective of developing an intelligent digital dashboard is to provide a robust interface for designers of service-oriented systems. The interface allows a system designer to upload a service interface design in WSDL. The system computes the service granularity value of the current service interface design and internal quality attributes. When the system designer is able to define the targeted values of complexity, cohesion and coupling, the system should provide the range of values required to achieve the appropriate service interface design that balances the trade-off between the service quality attributes. This extension depends on the completeness of the extension proposed in section 9.2.1. The dashboard should provide functionalities in two ways:

- Complete service identification process. With a given a business process model, the system designer can upload the business process model and generate the service interface designs. The metrics would then be calculated to generate the underlying mathematical equations. The normalisation of the intersecting points will generate the area of the optimum range of values that can be used to select the appropriate service interface design.

- Partial service identification process. This assumes that service interface designs (WSDL) already exist that can be processed for metric calculations to attain the appropriate service interface design.

9.2.3 Expand the Dataset of the Study

The dataset size in this thesis was limited and this factor inevitably affect the reliability of the research findings, especially in the results generated from studying the relationships between the service quality attributes. Access to an large sized database could be achieved by first improving the extraction mechanism for online web services (such as Amazon Web Services (AWS)) that was introduced in section 6.5, and second, enhancing the suggested algorithm to generate more than five service-interface designs for each service choreography file (WS-CDL code).

References

Akiyama, F. (1972). “An Example of Software System Debugging.” Information Processing 71 Proceedings of the IFIP Congress 1.

Alba, M. and S. Gil (2011). Validation and Calibration of Quantitative Models for Software Development Effort and Size Estimation. Computing Congress (CCC), 6th Colombian.

Alistair, B., M. Dumas, et al. (2005) “A Critical Overview of the Web Services Choreography Description Language (WS-CDL).” BPTrends Newsletter

Alistair, B., M. Dumas, et al. (2005). Service Interaction Patterns, Springer Berlin / Heidelberg. 3649: 302-318.

Arsanjani, A. (2004) “Service-Oriented Modeling and Architecture How to Identify, Specify, and Realize Services for your SOA.”

Arsanjani, A. (2005) “Toward A Pattern Language for Service-Oriented Architecture and Integration, Part 1: Build a service eco-system.”

Arsanjani, A. and A. Allam (2006). Service-Oriented Modeling and Architecture for Realization of an SOA. The IEEE International Conference on Services Computing, SCC ’06. .

Arsanjani, A., S. Ghosh, et al. (2008). “SOMA: A Method for Developing Service-oriented Solutions.” IBM Systems Journal 47(3): 377-396.

Aversano, L., L. Cerulo, et al. (2008). Mining Candidate Web Services from Legacy Code. The 10th International Symposium on Web Site Evolution, 2008. WSE 2008. .

Baeza-Yates, R. and B. Ribeiro-Neto (1999). Modern Information Retrieval, Addison-Wesley-Longman.

Barker, A., C. Walton, et al. (2009). “Choreographing Web Services.” IEEE Transactions on Services Computing 2(2).

Basci, D. and S. Misra (2009). “Measuring and Evaluating a Design Complexity Metric for XML Schema Documents.” Journal of Information Science and Engineering 25(5): 1405-1425.

Bell, M. (2008). Service-Oriented Modeling : Service Analysis, Design, and Architecture. Hoboken, NJ, USA, John Wiley & Sons.

Bell, M. (2010). SOA Modeling Patterns for Service Oriented Discovery and Analysis. New Jersey, John Wiley & Sons, Inc.

Benaben, F., J. Touzi, et al. (2008). Mediation Information System Design in a Collaborative SOA Context through a MDD Approach. The First International Workshop on Model Driven Interoperability for Sustainable Information Systems (MDISIS'08). Montpellier, France.

Benedicto, J., I. Rosenberg, et al. (2010). Analysis of Standard Process Models:D6.3-C, EDIANA Consortium.

Berners-Lee, T. (2003). Web Services Program Integration across Application and Organization Boundaries, W3C.

Bezivin, J., H. Bruneliere, et al. (2005). Model Engineering Support for Tool Interoperability. The 4th Workshop in Software Model Engineering (WiSME 2005). Montego Bay, Jamaica.

Bezivin, J. and O. Gerb (2001). Towards a Precise Definition of the OMG/MDA Framework. The 16th Annual International Conference on Automated Software Engineering. Los Alamitos, IEEE Computer Soc: 273-280.

Bezivin, J., S. Hammoudi, et al. (2004). Applying MDA approach for Web service platform. Los Alamitos, IEEE Computer Soc.

BEzivin, J., F. Jouault, et al. (2003). "First Experiments with the ATL Model Transformation Language: Transforming XSLT into XQuery." Conference on Object-Oriented Programming Systems, Languages, and Applications.

Bianchini, D., C. Cappiello, et al. (2009). "P2S: A Methodology to Enable Inter-organizational Process Design through Web Services." Conference on Advanced Information Systems Engineering: 334-348.

Bieberstein, N., S. Bose, et al. (2005). Service-Oriented Architecture Compass: Business Value, Planning, and Enterprise Roadmap, IBM Press.

Biehl, M. (2010). Literature Study on Model Transformations. Stockholm, Sweden, , Technical Report, Royal Institute of Technology.

Biron, P., K. Permanente, et al. (2004). "XML Schema Part 2: Datatypes Second Edition, W3C Recommendation." from http://www.w3.org/TR/xmlschema-2/.

Boehm, B. W. (1976). "Software Engineering." Computers, IEEE Transactions on C-25(12): 1226-1241.

Boerner, R. and M. Goeken (2009). Identification of Business Services Literature Review and Lessons Learned. AMCIS 2009 Proceedings.

Boerner, R. and M. Goeken (2009). Service Identification in SOA Governance Literature Review and Implications for a New Method. The 3rd IEEE International Conference on Digital Ecosystems and Technologies, 2009. DEST '09. 2009.

Booch, G., R. Maksimchuk, et al. (2007). Object-Oriented Analysis and Design with Applications, Pearson Education.

Bordbar, B. and A. Staikopoulos (2004). Modelling and Transforming the Behavioural Aspects of Web Services. The 3rd Workshop in Software Model Engineering (WiSME@UML 2004) at The 7th IEEE International Conference on the UML 2004. Lisbon, Portugal.

Brahe, S. and B. Bordbar (2006). "A Pattern-Based Approach to Business Process Modeling and Implementation in Web Services." International Conference on Service Oriented Computing: 166-177.

Braunwarth, K. and B. Friedl (2010). Towards a Financially Optimal Design of IT Services. The 2010 International Conference on Information Systems (ICIS).

Brereton, P. and D. Budgen (2000). "Component-Based Systems: a Classification of Issues." Computer 33(11): 54-62.

Briand, L., S. Morasca, et al. (1996). "Property-Based Software Engineering Measurement." IEEE Trans. Softw. Eng. 22(1): 68-86.

Bucchiarone, A. and S. Gnesi (2006). A Survey on Services Composition Languages and Models. The International Workshop on Web Services Modeling and Testing (WS-MaTe2006), Palermo, Sicily, ITALY.

Cambronero, M. E., G. Diacuteaz, et al. (2009). "A Comparative Study Between WSCI, WS-CDL, and OWL-S." The 2009 IEEE International Conference on e-Business Engineering. ICEBE 2009.

Chen, F., S. Y. Li, et al. (2005). Feature Analysis for Service-Oriented Reengineering.

Chen, F., Z. Zhang, et al. (2009). Service Identification via Ontology Mapping. The 3rd Annual IEEE International Computer Software and Applications Conference, 2009. COMPSAC '09. .

Chinosi, M. and A. Trombetta (2011). "BPMN: An Introduction to the Standard." Computer Standards & Interfaces 34(1): 124-134.

Clark, J., C. Casanave, et al. (2001) "The ebXML Business Process Specification Schema Version 1.01."

Coalition, W. M. (2008) "XML Process Definition Language, version 2.1."

Cohen, J. (1988). Statistical Power Analysis for the Behavioral Sciences. Hillsdale, N.J., L. Erlbaum Associates.

Cohen, S. (2007). "Ontology and Taxonomy of Services in a Service-Oriented Architecture." Microsoft. Architecture Journal 11.

Costagliola, G., F. Ferrucci, et al. (2005). "Class Point: An Aapproach for the Size Estimation of Object-Oriented Systems." IEEE Transactions on Software Engineering 31(1): 52-74.

Czarnecki, K. and S. Helsen (2006). "Feature-Based Survey of Model Transformation Approaches." Ibm Systems Journal 45(3): 621-645.

Davis, M., R. Sigal, et al. (1994). Computability, Complexity, and Languages : Fundamentals of Theoretical Computer Science. Boston, Academic Press, Harcourt, Brace.

Debnath, N., F. A. Zorzan, et al. (2007). "Transformation of BPMN Subprocesses based in SPEM Using QVT." The 2007 IEEE International Conference on Electro/Information Technology: 170-175.

Decker, G., O. Kopp, et al. (2008). "An Introduction to Service Choreographies." IT - Information Technology 50(2): 122-127.

Decker, G., O. Kopp, et al. (2007). BPEL4Chor: Extending BPEL for Modeling Choreographies.

Decker, G., O. Kopp, et al. (2009). "Interacting Services: From Specification To Execution." Data & Knowledge Engineering 68(10): 946-972.

Decker, G., H. Overdick, et al. (2006). On the Suitability of WS-CDL for Choreography Modeling. In Proceedings of Methoden, Konzepte und Technologien Fur die Entwicklung von dienstebasierten Informationssystemen (EMISA 2006), Hamburg, Germany.

Decker, G. and M. Weske (2011). "Interaction-Centric Modeling of Process Choreographies." Inf. Syst. 36(2): 292-312.

Delessy, N. and E. Fernandez (2008). A Pattern-Driven Security Process for SOA Applications. Los Alamitos, Ieee Computer Soc.

Deutsch, M. and R. Willis (1988). Software Quality Engineering: A Total Technical and Management Approach, Prentice-Hall, Inc.

Dijkman, R. and M. Dumas (2004). "Service-oriented Design: A multi-viewpoint Approach." International Journal of Cooperative Information Systems 13(4): 337-368.

Dobson, G., R. Lock, et al. (2005). QoSOnt: a QoS Ontology for Service-centric Systems. The 31st EUROMICRO Conference on Software Engineering and Advanced Applications, 2005.

Dobson, G., R. Lock, et al. (2005). Quality of Service Requirement Specification using an Ontology. The 13th Int'l Requirements Engineering Conf. (RE 05), 2005,. .

DongSu, K., S. Chee-yang, et al. (2008). A Method of Service Identification for Product Line. The Third International Conference on Convergence and Hybrid Information Technology, 2008. ICCIT '08, 2008.

Dwivedi, V. and N. Kulkarni (2008). A Model Driven Service Identification Approach for Process Centric Systems. The SERVICES-2. IEEE Congress on Services Part II, 2008,.

Endrei, M., J. Ang, et al. (2004). Patterns : Service-Oriented Architecture and Web Services IBM Redbooks Durham, NC, USA, IBM.

Erl, T. (2005). Service-Oriented Architecture Concepts, Technology, and Design. Crawfordsville, Indiana., Prentice Hall/PearsonPTR

Erl, T., A. Karmarkar, et al. (2008). Web Service Contract Design & Versioning for SOA, Prentice Hall, 2008.

Erradi, A., S. Anand, et al. (2006). Evaluation of Strategies for Integrating Legacy Applications as Services in a Service Oriented Architecture. The Proceedings of the IEEE International Conference on Services Computing, IEEE Computer Society.

Erradi, A., S. Anand, et al. (2006). SOAF: An Architectural Framework for Service Definition and Realization. The IEEE International Conference on Services Computing, 2006. SCC '06.

Erradi, A., N. Kulkarni, et al. (2009). Service Design Process for Reusable Services: Financial Services Case Study. Service-Oriented Computing, ICSOC 2007: 606-617.

Fenton, N. E. and M. Neil (1999). "Software Metrics: Successes, Failures and New Directions." Journal of Systems and Software 47(2-3): 149-157.

Fischer, L. (2005). Workflow Handbook, Layna Fischer.

Fraley, C. and A. E. Raftery (1998). "How Many Clusters? Which Clustering Method? Answers Via Model-Based Cluster Analysis." The Computer Journal 41(8): 578-588.

Frankel, D. (2003). Model Driven Architecture: Applying MDA to Enterprise Computing, John Wiley & Sons.

Funk, C., C. Kuhmunch, et al. (2005). A Model of Pervasive Services for Service Composition. On the Move to Meaningful Internet Systems 2005: Otm 2005 Workshops, Proceedings. 3762,: 215-224.

Galster, M. and E. Bucherer (2008). A Business-Goal-Service-Capability Graph for the Alignment of Requirements and Services. IEEE Congress on Services 2008, Pt I, Proceedings: 399-406.

Genon, N., P. Heymans, et al. (2011). Analysing the Cognitive Effectiveness of the BPMN 2.0 Visual Notation. Proceedings of the Third international conference on Software language engineering. Eindhoven, The Netherlands, Springer-Verlag.

Gu, Q. and P. Lago (2010). Service Identification Methods: A Systematic Literature Review Towards a Service-Based Internet, Springer Berlin / Heidelberg. 6481: 37-50.

Haeng-Kon, K. (2008). "Modeling of Distributed Systems with SOA & MDA." IAENG International Journal of Computer Science 35(4).

Haesen, R., M. Snoeck, et al. (2008). On The Definition of Service Granularity and its Architectural Impact. Advanced Information Systems Engineering, Proceedings.

Haines, M. and M. Rothenberger (2010). "How a Service-Oriented Architecture May Change the Software Development Process." Communications of the ACM 53(8): 135-140.

Hidaka, S., Z. Hu, et al. (2009). Towards a Compositional Approach to Model Transformation for Software Development. Proceedings of the 2009 ACM symposium on Applied Computing. Honolulu, Hawaii, ACM.

Hirzalla, M., J. Cleland-Huang, et al. (2009). A Metrics Suite for Evaluating Flexibility and Complexity in Service Oriented Architectures. Service-Oriented Computing, ICSOC 2008 Workshops, Springer-Verlag: 41-52.

Hwang, S., W. Liao, et al. (2010). Web Services Selection in Support of Reliable Web Service Choreography. The 2010 IEEE International Conference on Web Services (ICWS), 2010.

ISO/IEC (2001). The ISO/IEC 9126-1:2001 Software Engineering: Product quality-Quality model.

ISO/IEC (2007). ISO/IEC 25020:2007:Software Engineering - Software Product Quality Requirements and Evaluation (SQuaRE) -Measurement Reference Model and Guide.

Jamshidi, P., M. Sharifi, et al. (2008). To Establish Enterprise Service Model from Enterprise Business Model. The IEEE International Conference on Services Computing, 2008. SCC '08.

Jianzhi, L., Z. Zhuopeng, et al. (2005). A Grid Oriented Approach to Reusing Legacy Code in ICENI Framework. The IEEE International Conference on Information Reuse and Integration, Conf, 2005. IRI -2005

Johnson, R. A. and G. Bhattacharyya (1986). Statistics: Principles and Methods. New York, NY, USA, John Wiley& Sons, Inc.

Jouault, F. and I. Kurtev (2006). Transforming models with ATL. Satellite Events at the Models 2005 Conference. J. M. Bruel. Berlin, Springer-Verlag Berlin. 3844: 128-138.

Kamari, S. and M. Khayyambashi (2010). A Semantic Algorithm for Automatic Interface Generation of Services Participating in Choreographies. The 2nd International Conference on Education Technology and Computer (ICETC), 2010.

Kan, S. and C. Jones (2004). Metrics and Models in Software Quality Engineering, Addison Wesley.

Kim, H. and R. Lee (2008). MS2Web: Applying MDA and SOA to Web Services, Springer Berlin / Heidelberg. 149: 163-180.

Kim, S., M. Kim, et al. (2008). Service Identification Using Goal and Scenario in Service Oriented Architecture. The 15th Asia-Pacific Software Engineering Conference, 2008. APSEC '08. .

Kim, Y. and K. Doh (2007). The Service Modeling Process Based on Use Case Refactoring. Business Information Systems, Proceedings. W. Abramowicz. 4439: 108-120.

Kim, Y. and K. Doh (2009). Formal Identification of Right-Grained Services for Service-Oriented Modeling. Proceedings of the 10th International Conference on Web Information Systems Engineering. PoznaD, Poland, Springer-Verlag.

Klazar, M. (2003). "Bell Numbers, Their Relatives, and Algebraic Differential Equations." Journal of Combinatorial Theory, Series A 102(1): 63-87.

Kleppe, A., J. Warmer, et al. (2003). MDA Explained. The Model Driven Architecture: Practice and Promise, Addison-Wesley.

Klose, K., R. Knackstedt, et al. (2007). Identification of Services – A Stakeholder-Based Approach to SOA Development and its Application in the Area of Production Planning. The 15th European Conference on Information Systems. St. Gallen, Switzerland,2007.

Kohlborn, T., A. Korthaus, et al. (2009). "Identification and Analysis of Business and Software Services : A Consolidated Approach." IEEE Transactions on Services Computing 2(1): 50-64.

Kohlborn, T., A. Korthaus, et al. (2009). Service Analysis - A Critical Assessment of The State of the Art. The 17th European Conference on Information Systems. Verona.

Kohlmann, F. and R. Alt (2007). Business-Driven Service Modelling - A Methodological Approach from the Finance Industry. The First International Working Conference on Business Process and Services Computing, 2007,, Leipzig, Germany., GI.

Kokash, N. (2006). A Comparison of Web Service Interface Similarity Measures. Proceeding of the 2006 conference on STAIRS 2006: Proceedings of the Third Starting AI Researchers' Symposium, IOS Press.

Kolovos, D., L. Rose, et al. (2012). The Epsilpon Book.

Kopp, O. and F. Leymann (2009) "Do we need internal behavior in choreography models?" Proceedings of the 1st CentralEuropean Workshop on Services and their Composition ZEUS 2009 438, 68-73.

Kopp, O., F. Leymann, et al. (2011). Modeling Choreographies: BPMN 2.0 Versus BPEL-based Approaches. Proceedings of the 4th International Workshop on Enterprise Modelling and Information Systems Architectures, EMISA 2011. Hamburg, Germany, GI. 190: 225-230.

Kopp, O., F. Leymann, et al. (2010) "Mapping Interconnection Choreography Models to Interaction Choreography Models." Proceedings of the 2nd CentralEuropean Workshop on Services and their Composition ZEUS 2010 (2010), 81-88.

Korherr, B. and B. List (2007). Extending the EPC and the BPMN with Business Process Goals and Performance Measures.

Kowalski, C. (1972). "On the Effects of Non-Normality on the Distribution of the Sample Product-Moment Correlation Coefficient." Journal of the Royal Statistical Society. Series C (Applied Statistics) 21(1): 1-12.

Krafzig, D., K. Banke, et al. (2005). Enterprise SOA: Service Oriented Architecture Best Practices. Upper Saddle River, NJ, Prentice Hall.

Kulkarni, N. and V. Dwivedi (2008). "The Role of Service Granularity in A Successful SOA Realization A Case Study." The 2008 IEEE Congress on Services Part 1 (SERVICES-1).

Lane, S. and I. Richardson (2011). "Process Models for Service-Based Applications: A Systematic Literature Review." Information and Software Technology 53(5): 424-439.

Li, S. and H. Miao (2008). "Modeling the Patterns of WS-CDL Interactions Based on Process Algebra." 2008 International Seminar on Future Information Technology and Management Engineering.

Linthicum, D. (2003). Next Generation Application Integration: From Simple Information to Web Services, Addison Wesley Press.

List, B. and B. Korherr (2006). "An Evaluation of Conceptual Business Process Modelling Languages." Applied Computing 2006. 21st Annual ACM Symposium on Applied Computing.

Liu, Y. and I. Traore (2007). Complexity Measures for Secure Service-Oriented Software Architectures. Proceedings of the Third International Workshop on Predictor Models in Software Engineering, IEEE Computer Society.

Luo, W. H. and Y. A. Tung (1999). "A Framework for Selecting Business Process Modeling Methods." Industrial Management & Data Systems 99(7-8): 312-319.

Ma, Q., N. Zhou, et al. (2009). Evaluating Service Identification with Design Metrics on Business Process Decomposition. The IEEE International Conference on Services Computing, 2009. SCC '09. .

Martin, D., M. Burstein, et al. (2004). OWL-S: Semantic Markup for Web Services. 22 November 2004, W3C.

McCall, J., P. Richards, et al. (1977). "Factors in Software Quality." Nat'l Tech.Information. Service. 1, 2 and 3.

Medjahed, B., B. Benatallah, et al. (2003). "Business-to-Business Interactions: Issues and Enabling Technologies." The VLDB Journal 12(1): 59-85.

Mellor, J. and M. Balcer (2002). Executable UML: A Foundation for Model-Driven Architecture, Addison-Wesley Professional.

Mellor, S., K. Scott, et al. (2004). MDA Distilled: Principles of Model-Driven Architecture, Addison Wesley.

Mendling, J. and M. Hafner (2008). "From WS-CDL Choreography to BPEL Process Orchestration." Journal of Enterprise Information Management 21(5): 525 - 542.

Milanovic, M. (2007). Modeling Rules on the Semantic Web. Faculty of Organizational Sciences. Belgrade, University of Belgrade. Master Thesis.

Moha, N., S. Sen, et al. (2010). "Evaluation of Kermeta for Solving Graph-based Problems." International Journal on Software Tools for Technology Transfer (STTT) 12,(3): 273-285.

Mohagheghi, P. and V. Dehlen (2008). Developing a Quality Framework for Model-Driven Engineering. Models in Software Engineering. G. Holger, Springer-Verlag: 275-286.

Mowbray, T. and R. Malveau (1997). CORBA Design Patterns, John Wiley & Sons.

Munson, J. (2003). Software Engineering Measurement, Auerbach Publications.

Nayak, N., A. Nigam, et al. (2006). Concepts for Service-Oriented Business Thinking. The IEEE International Conference on Services Computing, 2006. SCC '06. 2006.

Nguyen, V. (2010). Improved Size and Effort Estimation Models for Software Maintenance. The 2010 IEEE International Conference on Software Maintenance (ICSM), 2010.

OASIS Standard (2007) "Web Services Business Process Execution Language (WSBPEL) 2.0."

OMG (2002). Meta Object Facility (MOF) Specification, OMG.

OMG (2002). OMG/RFP/QVT MOF, 2.0 Query/Views/Transformations, OMG.

OMG (2003). MDA Guide Version 1.0.1. Framingham, Massachusetts, OMG, Tech.

OMG (2006). Object Constraint Language (OCL) Specification, OMG.

OMG (2008). Meta Object Facility (MOF) 2.0 Query/View/Transformation Specification, OMG.

OMG (2008) "OMG Business Process Modeling Notation , V1.1."

OMG (2009) "OMG Business Process Modeling Notation , V1.2."

OMG (2010) "BPMN 2.0 by Example."

OMG (2011) "Business Process Model and Notation (BPMN 2.0)."

Pandian, R. (2003). Software Metrics: A Guide to Planning, Analysis, and Application, Taylor & Francis.

Papazoglou, M. and W. Van (2006). "Service-Oriented Design and Development Methodology." International Journal of Web Engineering and Technology 2(4).

Papazoglou, M. P., P. Traverso, et al. (2007). "Service-Oriented Computing: State of the Art and Research Challenges." Computer 40(11): 38-45.

Peltz, C. (2003). "Web Services Orchestration and Choreography." Computer 36(10): 46-52.

Perepletchikov, M., C. Ryan, et al. (2007). Cohesion Metrics for Predicting Maintainability of Service-Oriented Software. The Seventh International Conference on Quality Software, 2007. QSIC '07.2007.

Perepletchikov, M., C. Ryan, et al. (2007). Coupling Metrics for Predicting Maintainability in Service-Oriented Designs. The 18th Australian Software Engineering Conference, 2007. ASWEC 2007. .

Perepletchikov, M., C. Ryan, et al. (2005). The Impact of Software Development Strategies on Project and Structural Software Attributes in SOA. The OTM 2005 Workshops On the Move to Meaningful Internet Systems 2005,. Berlin, Germany, Springer. 3762: 442-451.

Perepletchikov, M., C. Ryan, et al. (2010). "The Impact of Service Cohesion on the Analyzability of Service-Oriented Software." IEEE Transactions on Services Computing 3(2): 89-103.

Qian, K., L. Jigang, et al. (2006). Decoupling Metrics for Services Composition. The 5th IEEE/ACIS International Conference on Computer Information Science, Honolulu, HI,, IEEE Comput. Soc.

Rabhi, F., H. Yu, et al. (2006). A Service-Oriented Architecture for Financial Business Processes: A Case Study in Trading Strategy Simulation. Information Systems and e-Business Management, Springer-Verlag.

Ramollari, E., D. Dranidis, et al. (2007). A Survey of Service Oriented Development Methodologies. The Second European Young Researchers Workshop on Service Oriented Computing, Leicester, UK.

Recker, J., M. zur Muehlen, et al. (2009). Measuring method complexity: UML versus BPMN. The 15th Americas Conference on Information Systems AMCIS 2009. San Francisco, California, Association for Information Systems.

Reldin, P. and P. Sundling (2007). Explaining SOA Service Granularity: How IT-Strategy Shapes Services Institute of Technology Linkoping University. Master Thesis.

Rolland, C. and R. CentreKaabi (2007). An Intentional Perspective to Service Modeling and Discovery. The 31st Annual International Computer Software and Applications Conference, 2007. COMPSAC 2007. .

Rosen, M., B. Lublinsky, et al. (2008). Applied SOA: Service-Oriented Architecture and Design Strategies. New York, Wiley.

Ross-Talbot, S. (2004). Web Services Choreography and Process Algebra. SWSL Meeting,2004.

Rossi, P. and G. Fernandez (2003). Definition and Validation of Design Metrics for Distributed Applications. Proceedings of the 9th International Symposium on Software Metrics, IEEE Computer Society.

Rud, D., A. Schmietendorf, et al. (2006). Product Metrics for Service Oriented Infrastructures. The 6th International Workshop on Software Measurement and DASMA Metrik Kongress (IWSM/MetriKon 2006). Potsdam, Germany: 161-174.

Seidewitz, E. (2003). "What models mean." Software, IEEE 20(5): 26-32.

Senivongse, T., N. Phacharintanakul, et al. (2010). "A Capability Granularity Analysis on Web Service Invocations." Proceedings 2010 World Congress on Engineering and Computer Science (WCECS 2010).

Shapiro, S. and M. Wilk (1965). "An Analysis of Variance Test for Normality." Biometrika 3,(52).

Shim, B., S. Choue, et al. (2008). A Design Quality Model for Service-Oriented Architecture. The 15th Asia-Pacific Software Engineering Conference, 2008. APSEC '08.

Shirazi, H. M., N. Fareghzadeh, et al. (2009). "A Combinational Approach to Service Identification in SOA." Journal of Applied Sciences Research 5(10): 1390-1397.

Sindhgatta, R., B. Sengupta, et al. (2009). Measuring the Quality of Service Oriented Design. Proceedings of the 7th International Joint Conference on Service-Oriented Computing. Stockholm, Springer-Verlag.

Sneed, H. M. (2001). Wrapping Legacy COBOL Programs Behind an XML-interface. Reverse Engineering, 2001. Proceedings. Eighth Working Conference on.

Stewart, G. and A. Chakraborty (2010). Service Identification Through Value Chain Analysis and Prioritization. Proceedings of the 16th Americas Conference on Information Systems : Sustainable IT Collaboration around the Globe. Lima, Peru, Association for Information Systems (AIS).

Sweeney, R. (2010). Achieving Service-Oriented Architecture : Applying an Enterprise Architecture Approach, Wiley.

Taylor, I., M. Shields, et al. (2003). "Distributed P2P computing within Triana: a galaxy visualization test case." Proceedings International Parallel and Distributed Processing Symposium.

Tetlow, P., J. Pan, et al. (2006) "Ontology Driven Architectures and Potential Uses of the Semantic Web in Systems and Software Engineering."

Van Nuffel, D. (2007). "Towards a Service-Oriented Methodology: Business-Driven Guidelines for Service Identification." On the Move to Meaningful Internet Systems 2007: OTM 2007 Workshops: 294-303.

W3C (1999). XSL Transformations (XSLT), W3C.

W3C (2004) "The W3C Web Services Architecture."

W3C (2005). Web Services Choreography Description Language Version 1.0, W3C.

Wang, X., S. HU. , et al. (2007). Integrating Legacy Systems within The Service-oriented Architecture. Power Engineering Society General Meeting, 2007. IEEE.

Wiersma, R. (2010). Finding an Optimum in Service Granularity, HU University of Applied Sciences. Master Thesis.

Xiao-jun, W. (2009). "Metrics for Evaluating Coupling and Service Granularity in Service Oriented Architecture." The 2009 International Conference on Information Engineering and Computer Science. ICIECS 2009.

Yan, X. and X. Su (2009). Linear Regression Analysis: Theory and Computing, World Scientific Publishing Company.

Yang, H., Z. Cui, et al. (1999). Extracting Ontologies from Legacy Systems for Understanding and Re-Engineering. The 23rd International Computer Software and Applications Conference, IEEE Computer Society.

Yang, W. (1996). "Bell Numbers and K-Trees." Discrete Mathematics 156: 247-252.

Yousef, R., M. Odeh, et al. (2009). BPAOntoSOA: A Generic Framework to Derive Software Service Oriented Models From Bsiness Process Architectures. The Second International Conference on the Applications of Digital Information and Web Technologies, 2009. ICADIWT '09.

Zaha, J., A. Barros, et al. (2006). Let's Dance: A Language for Service Behavior Modeling. On the Move to Meaningful Internet Systems 2006: CoopIS, DOA, GADA, and ODBASE, Springer Berlin / Heidelberg. 4275: 145-162-162.

Zdun, U. and S. Dustdar (2007). "Model-Driven and Pattern-Based Integration of Process-Driven SOA Models." International Journal of Business Process Integration and Management (IJBPIM) 2(2): 109–119.

Zhang, L. J., N. Zhou, et al. (2008). "SOMA-ME: A platform for the model-driven design of SOA solutions." Ibm Systems Journal 47(3): 397-413.

Zhang, Q. and X. Li (2009). Complexity Metrics for Service-Oriented Systems. The Second International Symposium on Knowledge Acquisition and Modeling, 2009. KAM '09.

Zhang, Z., R. Liu, et al. (2005). Service Identification and Packaging in Service Oriented Reengineering. The Conference of SEKE.

Zhang, Z. and H. Yang (2004). Incubating Services in Legacy Systems for Architectural Migration. The 11th Asia-PacificSoftware Engineering Conference, 2004.

Zheng, L. and J. Keung (2010). Software Cost Estimation Framework for Service-Oriented Architecture Systems Using Divide-and-Conquer Approach. The 2010 Fifth IEEE International Symposium on Service Oriented System Engineering (SOSE),.

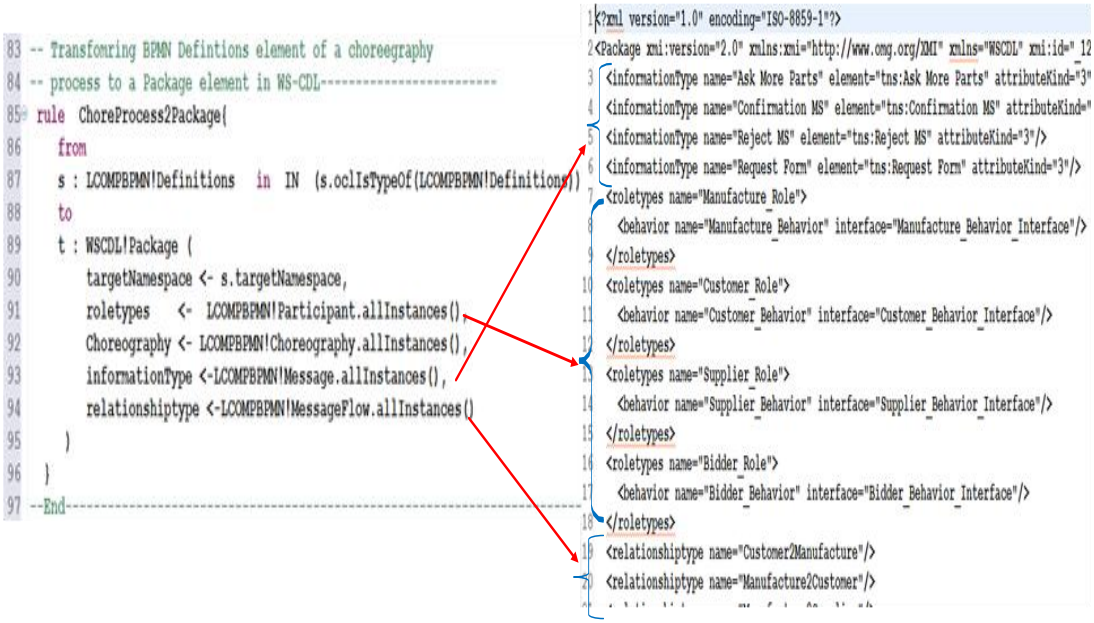
Zou, Y. and K. Kontogiannis (2001). Towards a Web-centric Legacy System Migration Framework. The IEEE Workshop on Network Centric, held in conjunction with ICSE 2001, Toronto, ON.

Appendix A

A-1. An Example for BPMN-to-WS-CDL

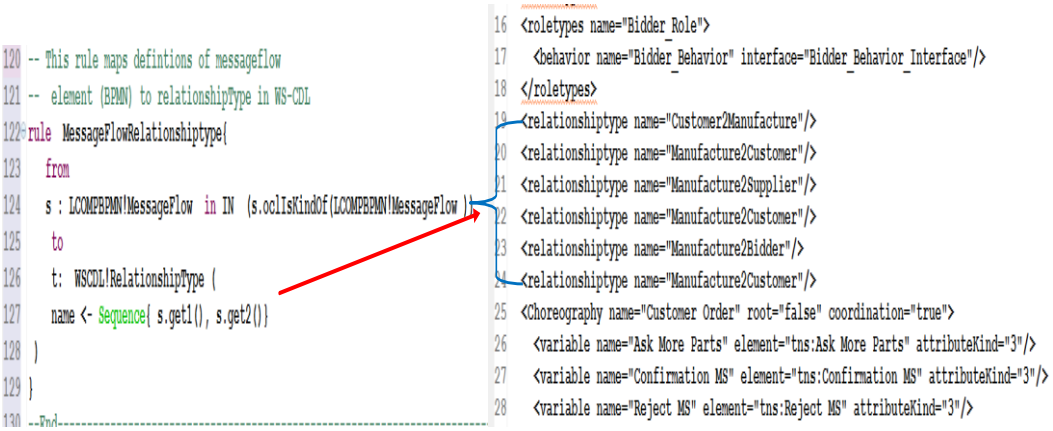
Due to space limitations, we will give some example code of the code transforming BPMN-to-WS-CDL. Listing A-1.1 shows the ATL rule "ChoreProcess2Package" shows the creations of the basic elements of the code of service choreography in WS-CDL. The meta-models "LCOMPBPMN" and WSCDL provide properties and definitions of elements in BPMN and WS-CDL, respectively. The rule creates an instance for elements of WS-CDL corresponding to matched BPMN elements. These instances points to the definitions of these elements in next rules. For example, every instance of the "LCOMPBPMN.Paticipant" element is mapped to an instance called a "relationshiptype".

Listing A-1.1 ATL rule for ChoreProecss2Package



As an example, after creating the instance "relationshiptype", we map the definitions of BPMN:Messageflow element to WS-CDL:RolationsipType in ATL rule "MessageFlowRelationshiptype". The definitions of the RolationsipType element include the *name* attributes. (At line 124) the *name* is defined based on invoking two helpers (a function) shown in listing A-1.2, which are "get1" and "get2" helpers.

Listing A-1.2 ATL rule for MessageFlowRelationshiptype



```
-- This rule maps defintions of messageflow
-- element (BPMN) to relationshipType in WS-CDL
rule MessageFlowRelationshiptype{
  from
  s : LCOMPBPMN!MessageFlow in IN (s.oclIsKindOf(LCOMPBPMN!MessageFlow))
  to
  t: WSCDL!RelationshipType (
  name <- Sequence{ s.get1(), s.get2()}
 )
}
--End
```

```
<roletypes name="Bidder_Role">
  <behavior name="Bidder_Behavior" interface="Bidder_Behavior_Interface"/>
</roletypes>
<relationshiptype name="Customer2Manufacture"/>
<relationshiptype name="Manufacture2Customer"/>
<relationshiptype name="Manufacture2Supplier"/>
<relationshiptype name="Manufacture2Customer"/>
<relationshiptype name="Manufacture2Bidder"/>
<relationshiptype name="Manufacture2Customer"/>
<Choreography name="Customer Order" root="false" coordination="true">
  <variable name="Ask More Parts" element="tns:Ask More Parts" attributeKind="3"/>
  <variable name="Confirmation MS" element="tns:Confirmation MS" attributeKind="3"/>
  <variable name="Reject MS" element="tns:Reject MS" attributeKind="3"/>
```

Listing A-1.3 shows the "get1" and "get2" helpers which map the *sourceRef* and *targetRef* attributes in MessageFlow element of BPMN into a combined name of the relationshiptype element. At line 9, for the instance of Messageflow, map the value of *sourceRef* when the name exits. At line 14 in "get2" helper, for the instance of Messageflow, map the value of *targetRef* when the name exits.

Listing A-1.3 ATL helpers for get1 and get

```
7  -- This helper map the name of the source intiates the interaction
8  helper context LCOMPBPMN!MessageFlow  def : get1() :  String  =
9   self.sourceRef ->collect( e | e.name )->debug() ;
10 --end
11
12 -- This helper map the name of the receiver of the interaction
13 helper context LCOMPBPMN!MessageFlow  def : get2() :  String  =
14   self.targetRef ->collect( e | e.name )->debug() ;
15 --end
```

A-2 An Example for WS-CDL-to-WSDL

Here is an ATL rule example for mapping WS-CDL to WSDL. Listing A-2.1 gives an example of matched rule which is "Package2Description" for the Package element of WS-CDL!Package of the WS-CDL meta-model. It maps the Package of the WS-CDL model into the Description element in the WSDL!Description meta-model. It maps the attribute *name* of the package element and two instances of informationType and Choreography elements in BPMN into Types and Interface elements in WSDL. Line 104-106 shows the "do" statement invoked for code structure. Within the instance informationTyep creating, we invoked a lazy rule "EXMessageTypes".

Listing A-2.1 ATL rule for Package2Description

```
-- This rule maps the defintions of the package element
-- (BPMN) to Descirption element in WS-CDL
rule  Package2Description{
 from
   s : WSCDL!Package in  IN  (s.oclIsTypeOf(WSCDL!Package))
 to
   t : wsdl!Description (
    name       <- s.name + 'Service'
    types      <-  s.informationType ->collect(e| thisModule.EXMessageTypes(e)),
    interface  <-  s.Choreography ->collect(e| thisModule.EXChorTOInterface(e))
    )
    do {
        thisModule.t <- t;
    }
}
--End------------------------------------------------------------------------
```

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xmi:XMI xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI" xmlns:xsi="http
  <Description name="IncidentManagement">
    <types xsi:type="_1:XsSchema">

    </types>
   ..
   ..
    <interface name="IncidentManagementInterface">
      <operation name="Customer Has a Problem">

      </operation>
      ..
      ...
    </interface>
      ...
</Description>
```

Listing A-2.2 shows the lazy rule EXMessageTypes that tranformd detailed definitions of the InformationType of the WS-CDL meta-model into the XsSechmea element within the Types element of the WSDL meta-model. It shows the mapping of *the name* and *attributekind* attributes in WS-CDL into name and type in WSDL.

Listing A-2.2 ATL lazy rule for EXMessageTypes

```
lazy rule  EXMessageTypes {
    from s : WSCDL!InformationType  (s.oclIsTypeOf(WSCDL!InformationType))
      to t :    wsdl!XsSchema (
        elementDeclarations <-  elemdef
          ),
        elemdef : wsdl!XsElementDeclaration (
        name <- s.name,
        typeDefinition <- if  s.attributeKind = '3' then
                              complex
                         else if s.attributeKind = '2' then
                              userDefined
                        else if s.attributeKind = '1' then
                              simple
                        else  OclUndefined endif endif endif
          ),
         simple : wsdl!XsSimpleTypeDefinition (
         contents <- con
          ),
         complex : wsdl!XsComplexTypeDefinition (
             content <- con
          ),
         userDefined : wsdl!XsUserDefinedDefinition (
         contents <- con
          ),
         con : wsdl!XsParticle (
             term <- ter
          ),
         ter : wsdl!XsModelGroup (
             compositor <- #sequence
          )
}
```

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xmi:XMI xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI" xmlns:xsi="h
  <Description name="IncidentManagement">
    <types xsi:type="_1:XsSchema">
      <elementDeclarations name="Message_4">
        <typeDefinition xsi:type="_1:XsComplexTypeDefinition"/>
      </elementDeclarations>
    </types>
    <types xsi:type="_1:XsSchema">
      <elementDeclarations name="Message_5">
        <typeDefinition xsi:type="_1:XsSimpleTypeDefinition"/>
      </elementDeclarations>
    </types>

</Description>
```

A.3 Service Element Extractor

Listing A-3.1 shows the parser that is developed on top of an open source SOA tool provided by a company called Predic8 (as mentioned previously). It extracts local .wsdl files and can be used to extract online web services such as AWS.

Listing A-3.1 a sample code for service element extractor method

```
43  private static void ListWSDLElements(){
44      int Total_Ser = 0;
45      int Total_Oper = 0;
46      int Total_Mes = 0;
47      int Total_Secms = 0;
48      WSDLParser parser = new WSDLParser();
49      // Definitions wsdl = parser.parse("resources/CopyofS1.wsdl");
50     Definitions wsdl =
51     parser.parse("http://mechanicalturk.amazonaws.com/AWSMechanicalTurk/AWSMechanical
52      for (Service service : wsdl.getServices()) {
53        System.out.println("Service:-" + service.getName() + ", ");
54          Total_Ser++;
55          for (Port port : service.getPorts()) {
56          System.out.println("Port:-" + port.getName() + ", ");
57         }
58        System.out.println("Total of Services: " + Total_Ser);
59
60      for (Operation op : wsdl.getOperations()) {
61           System.out.println("Operations:-" + op.getName() + " , ");
62        // System.out.println("Operations:-" + op.getTypeQName(s type) + ", ");
63           Total_Oper++;
64                   }
65         System.out.println("Total of Operations: " + Total_Oper);
66
67         for (Message ms : wsdl.getMessages()) {
68         System.out.println("Messages:-" + ms.getName() + ", ");
69          Total_Mes++;
70         }
71      System.out.println("Total of Messages: " + Total_Mes);
72      }
```

A.4 Computation of Service quality metrics

Listing A-4.1 shows the main method for metres analyser and calculator packages. We used the class Scanner for parsing elements from a txt files and then pass syntax to private method for metrics computation. The calculation these metrics are dependent on the calculation of service granularity which is presented in the calculations of ODG and SOA.

Listing A-4.1 a sample code for main method for service metrics computation

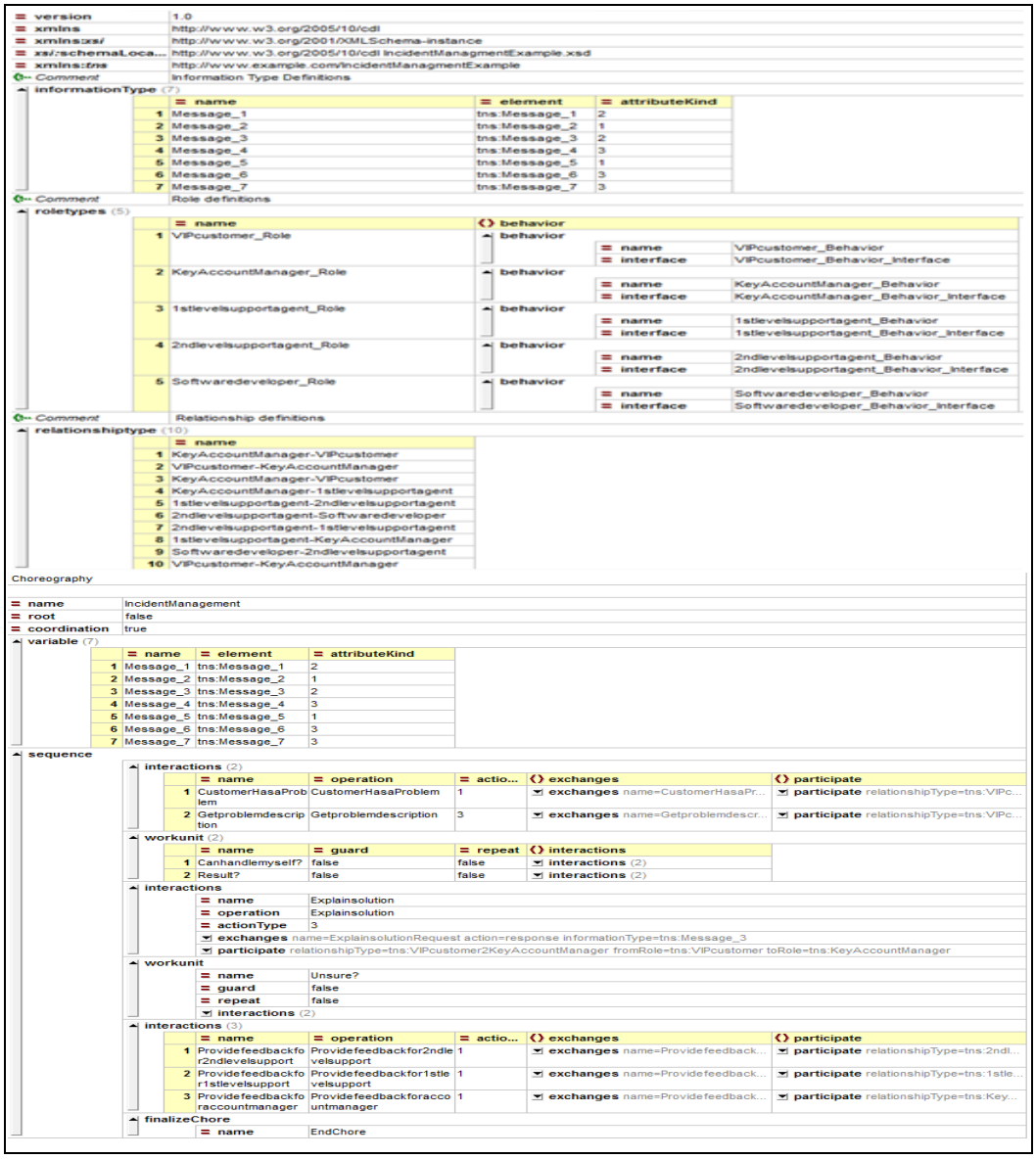
```
public static void main(String[] args) throws IOException {
    try {
    Scanner scanner = new Scanner(new BufferedReader(new
    FileReader("resources/Amazon Fulfillment Web Service/Input/Sci3/Scienario3,27.txt")));
    scanner.useDelimiter (System.getProperty("line.separator"));
    PrintWriter MyOutput = new PrintWriter("resources/Amazon Fulfillment Web Service/Output/Sci3
        {
        float[] Line_one = ParseLineOne(scanner.nextLine());
        float[] Line_tow = ParseLineTow(scanner.nextLine());
        float[] Line_thrid = ParseLineThird(scanner.nextLine());
        int Line_fourth = ParseLineFourth(scanner.nextLine());
        float[] ODG = CalculateODG(Line_one,Line_tow); //Data Granularity Score:
        float[] SOG = CalculateSOG(ODG,OFG); //Functionality Granularity Score
        float ASOG  = CalculateASOG(SOGtotal,NS); //Average Service Operations Granularity Score
        float ASOM  = CalculateASOM(SOGtotal,NS); //Average Service Operation Complexity (ASOM)
        float SOCOFG  = CalculateModeOFG(OFG,NO); //mode of OFG
        float SOCODG  = CalculateModeODG(ODG,NO); //mode of ODG
        double SOCMax  = CalculateMax(OFG,ODG,SOCOFG,SOCODG); //Max number of occurrences of 27
        double ASOC  = CalculateASOC(SOCMax,NS);
        double ASOU  = CalculateASOU(Line_fourth,NS);
        }
         MyOutput.println("This Service Amazon Fulfillment Service with Three web services");
         MyOutput.println("final line of result should be (ASOG),(ASOM), (ASOC),(ASOU)as follow"
         MyOutput.print("("+ ASOG + ", " + ASOM + ", " + ASOC + ", " + ASOU + ")");
         MyOutput.flush();
         MyOutput.close();

         scanner.close();
        }
     catch (FileNotFoundException e) {
       e.printStackTrace();
         System.err.println("Error:NUMBER_1"); }
      catch (IOException e) {
    e.printStackTrace(); System.err.println("Error:NUMBER_2");}
}
```

Appendix B

B.1 WS-CDL code for an Incident Management Scenario

In listing B-1.1, the hierarchical structure document shows the results of mapping between BPMN 2.0 and WS-CDL graphically for Incident Management scenario. We can see that transformed elements from BPMN diagram correctly transformed to corresponding WS-CDL element based on the suggested mapping.

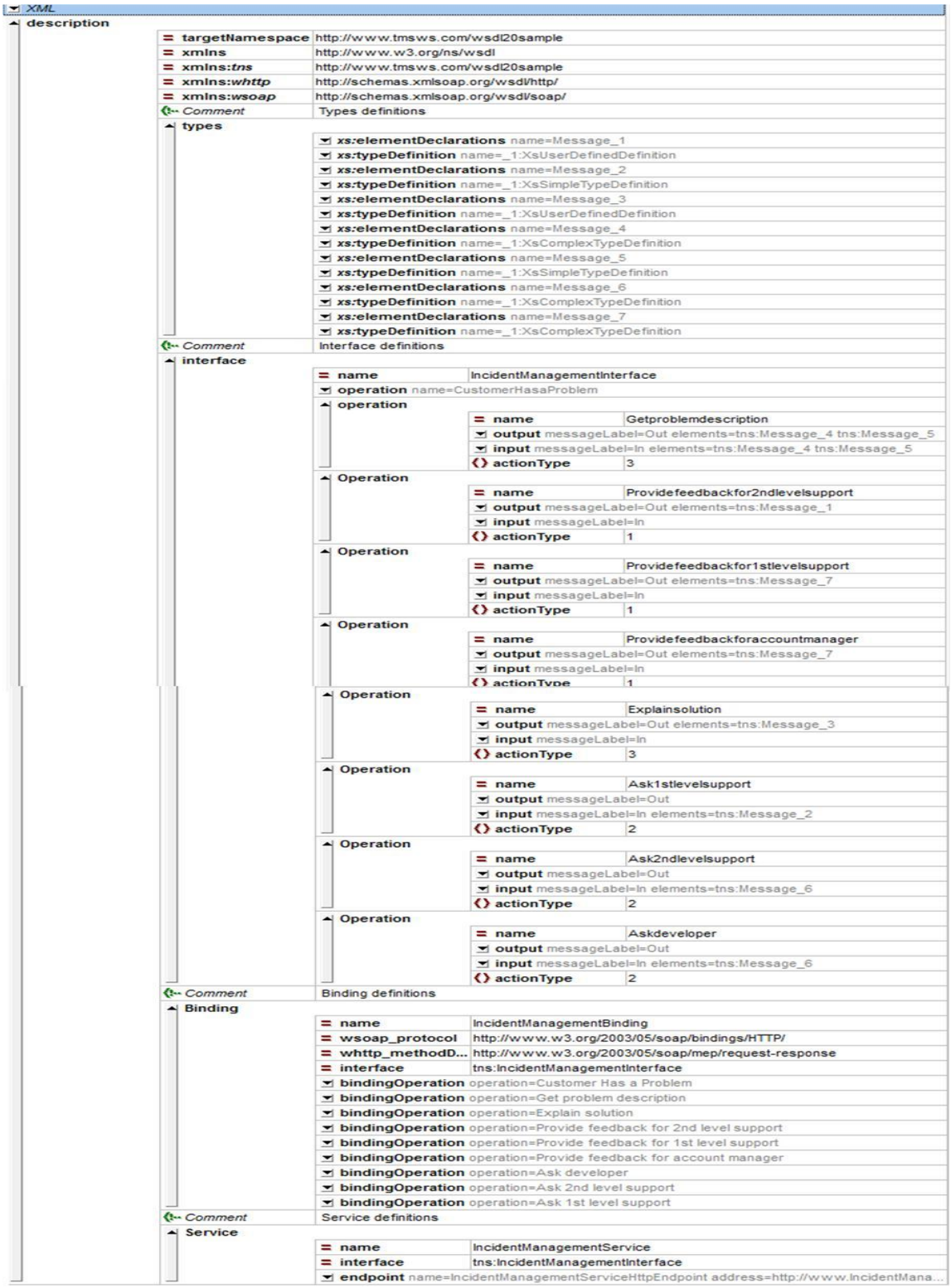
Listing B-1.1 The hierarchical structure of the IncidentMangment.cdl



B.2 WSDL code for an Incident Management Scenario

Listing B-2.1 shows the results of mapping between WS-CDL and WSDL graphically for Incident Management scenario.

Listing B-2.1 the hierarchical structure of the IncidentMangment.wsdl



Appendix C

C.1 ASOG/ASOM Relationships Framework's dataset

Table C-1.1: Tests of Normality

	Kolmogorov-Smirnov[a]			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
ASOM	.114	15	.200*	.966	15	.799

a. Lilliefors Significance Correction

*. This is a lower bound of the true significance.

Table C-1.2: Model Summary b

Model	R	R Square	Adjusted R Square	Std. Error of the Estimate
1	.976[a]	.953	.950	.085198140

a. Predictors: (Constant), ASOG

b. Dependent Variable: ASOM

Table C-1.3: ANOVA[b]

Model		Sum of Squares	df	Mean Square	F	Sig.
1	Regression	1.934	1	1.934	266.375	.000a
	Residual	.094	13	.007		
	Total	2.028	14			

a. Predictors: (Constant), ASOG

b. Dependent Variable: ASOM

C.2 ASOG/ASOC Relationships Framework's dataset

Table C-2.1: Tests of Normality

	Kolmogorov-Smirnov[a]			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
ASOC	.385	15	.000	.664	15	.000

a. Lilliefors Significance Correction

C.3 ASOG/ASOU Relationships Framework's dataset

Table C-3.1: Tests of Normality

	Kolmogorov-Smirnov[a]			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
ASOU	.135	15	.200*	.935	15	.322

a. Lilliefors Significance Correction

*. This is a lower bound of the true significance.

Table C-3.2: Model Summary

Model	R	R Square	Adjusted R Square	Std. Error of the Estimate
1	.835[a]	.697	.673	.290553058

a. Predictors: (Constant), ASOG

Table C-3.3: ANOVA[b]

Model		Sum of Squares	df	Mean Square	F	Sig.
1	Regression	2.520	1	2.520	29.854	.000[a]
	Residual	1.097	13	.084		
	Total	3.618	14			

a. Predictors: (Constant), ASOG

b. Dependent Variable: ASOU