UNIVERSITY OF SOUTHAMPTON

FACULTY OF ENGINEERING AND THE ENVIRONMENT

Aeronautics, Astronautics and Computational Engineering

Computational Engineering Design Research Group

An Investigation into Information Reuse for Cost Prediction - from Needs to a Data Reuse Framework

by

Joshua Jeeson Daniel

Thesis for the degree of Doctor of Philosophy

February 2014

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF ENGINEERING AND THE ENVIRONMENT
Aeronautics, Astronautics and Computational Engineering

Doctor of Philosophy

AN INVESTIGATION INTO INFORMATION REUSE FOR COST PREDICTION - FROM NEEDS TO A DATA REUSE FRAMEWORK

by Joshua Jeeson Daniel

The need to be able to reuse a wide variety of data that an organise has created, constitutes a part of the challenge known as 'Broad Data'. The aim of this research was to create a framework that would enable the reuse of broad data while complying with the corporate requirements of data security and privacy.

A key requirement in enabling reuse of broad data is to ensure maximum interoperability among datasets, which in Linked Data depends on the URIs (Uniform Resource Identifier) that the datasets have in common (i.e. reused). The URIs in linked data can be dereferenced to obtain more information about it from its owner and hence dereferencing can have a profound impact on making someone reuse a URI. However, the wide variety of vocabulary in broad data means the provenance and ownership of URIs could be key in promoting its reuse by the data creators. The full potential offered by linked data cannot be realised due to the fundamental way the URIs are currently constructed. In part, this is because the World Wide Web (Web) was designed for an open web of documents, not a secure web of data.

By making subtle but essential changes to the building blocks one can change the way data is handled on the Web, thereby creating what has been proposed in this thesis as the World Wide Information web (WWI). The WWI is based on a framework of people and things that are active contributors to the web of data (hereinafter referred to as 'active thing'), identified by URIs. The URI for an active thing is constructed from its path in the organisational stakeholder hierarchy to represent the provenance of ownership. As a result, it becomes easier to reference data held in sparse and heterogeneous resources, to navigate complex organisational structures, and to automatically include the provenance of the data to support trust based data reuse and an organic growth of linked data. As a result, the new data retrieval technique referred to as 'domino request' was demonstrated, where sparsely located linked data can be reused as though it was from a single source. With the use of a domino request on WWI web there is no more need to include the name of the organisation itself to maintain a catalogue of all the data sources to be queried and thus, making the ‘security by privacy’ on the Web a reality. At the same time, WWI allows the data owner or its stakeholder to maintain their privacy not only on the source of data, but also on the provenance of individual URIs that describe the data.

The thesis concludes that WWI is a suitable framework for broad data reuse and in addition demonstrates its application in managing data in the air travel industry, where security by privacy could play a significant role in controlling the flow of data among its ‘internet of things’ that have multiple stakeholders.

Contents

List of Figures

List of Tables

Abbreviations

ANN	Artificial Neural Networks
API	Application Programming Interface
ATC	Air Traffic Control
CAD	Computer Aided Design
CBR	Case Based Reasoning
CSS	Cascading Style Sheet
CSV	Comma Separated Values
DNS	Domain Name System
DOM	Document Object Model
DU	Data Unit
EP3	Engineering Product Price Predictor
FAA	Federal Aviation Administration
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IMDB	Internet Movie Database
IT	Information Technology
LDMG	Labelled Directed Multigraph
LDRT	Labelled Directed Rooted Tree
LOD	Linked Open Data
NLP	Natural Language Processing
OBU	Operational Business Unit
OS	Operating System
OWL	Web Ontology Language
PDF	Portable Document Format
PHP	Hypertext Preprocessor
PILM	Product Introduction Lifecycle Management
PL	Planning Level
PNG	Portable Network Graphics
RAM	Random Access Memory
RDF	Resource Description Framework
RFI	Request For Information

RFP	Request For Proposal
SILOET	Strategic Investment in Low Emissions Technology
SPARQL	SPARQL Protocol and RDF Query Language
UI	User Interface
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
VB	Visual Basic
Web	World Wide Web
WWI	World Wide Information Web
WWW	World Wide Web
XHTML	Extensible HyperText Markup Language
XML	Extensible Markup Language

Declaration of Authorship

I, Joshua Jeeson Daniel , declare that the thesis entitled *An Investigation into Information Reuse for Cost Prediction - from Needs to a Data Reuse Framework* 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;
- none of this work has been published before submission

Signed:...

Date:..

Acknowledgements

"Beauty is as old as the universe and as new as the moment."

– From a poster in Velliamma's sweet cottage

I have had to learn to appreciate beauty and time has always been my companion. I am sincerely grateful to Prof. Jim Scanlan and Dr. Gary Wills, my supervisors, for the opportunity to learn from them. I am thankful to Prof. Jim Scanlan for enabling me to use the freedom of thought and for having taken me through the process of finding a problem worth solving. I am equally thankful to Dr. Gary Wills for helping me grow all the seeds of thoughts, for motivating me to persevere and for helping me to connect the dots.

I am grateful to nature, the balance of life, for now I am able to appreciate the love that I have received from my family as ever and for ever. *Amma*, *Appa* and *Chikku* - your love has been my foundation.

I would like to thank University of Southampton for having hosted me and covered my international fee difference through the scholarship. I am thankful to Rolls-Royce Plc for the scholarship to fund the tution fees and my living and also for the unparalleled access to its workings. I am also grateful to Dr. Steve Wiseall and Dr. Rob Marsh for all their timely advices. Thanks to Frank McQuade for showing me what it means to be self confident. Thanks is also due to Tope Omitola, Priyanka Singh and Jonathan Butters for having spent your time helping me learn. I am also grateful to Prof. John Domingue, the crew and the participants of 2012 ESWC Summer School for giving me the purpose to persevere.

The poet Rumi once noted "This moment is all that there is". Sanjay Pant, Aditya Deshpande, Amrith Surendra, Alejandro Ortega Ancel, Yoshiyuki Sakai, Sairajan, Florencia Angelica, Aniruddha Kaushal, Ankita Kankani, Raj Shah, Wijendra Gnanendran, Shevan Algama, Ricky Patel, David Cooper, Chris Dodd, Aditya Karnik, Moresh Wankhede, Dilesh Bhardwa, Athanasios Makrodimopoulos, Giorgos Ragousis, Becca Mollie, Tran-Thi Hien, Pin-ru Huang, Manan Thakkar, Rahul Menon, Harbhajan Kaur Benning, Rukmangadhan Govindarajan, Jie Zhou, Dong Li, Aditya Sundaravadivel, Vijaylakshmi Easwar, Sriram Boothalingam, Srija Bhagavatula, Peasy Photos, and Southampton - thank you for making the moments wonderful. And finally I thank every one who has ever touched my life, as you are a reason for how I am in this moment.

... and I think to myself, what a wonderful world.

Chapter 1

Introduction

A gas turbine can be considered to be the pinnacle of human engineering. Its extreme complexity in design and manufacturing rarely hinders the operational reliability, which has led us to take air travel for granted over the last decades. This rare combination is an engineer's 'dream come true'.

Rolls-Royce is known for their excellence in design and manufacturing of gas turbine engines for a worldwide market in civil aerospace, military aerospace, marine, and energy sector. With a standard modern gas turbine having up to 18,000 components, designing a new engine is not at all an easy task. However the average time to bring an engine into commercial production from the time of customer-request is about 5 years. The last major and noted redesign work carried out by Rolls-Royce was in the development of three-shaft axial flow engine (RB-211), which was put into service for the first time in 1972. The research and development was so intensive that expenditure lead the company to bankruptcy. However the new design was a market success with its notable benefits of being much lighter and smaller compared to a similar two-shaft engine. This changed the course of Rolls-Royce; within a decade they had turned back into a success story. The achievement had been so remarkable that all their commercial large engines (Trent series) have evolved from the original three-shaft design of RB-211.

With the advent of the digital revolution and its adoption by corporates for cheaper, faster and reliable communication, a new door of opportunity has risen, an opportunity to reuse all the data and information. These data are being generated by the various functioning processes within the company, from engineering design and service to human resources and marketing. A crucial step to reinvent the way a company works would be to have a data reuse capability. This can support someone, who has taken a new role in the company, with vital information which previously was the benefit of the few who have had gained experience being in that role. This can empower a company with the ability for quick and efficient restructuring to meet the demands of the changing market, making them ready for the next leap in innovation.

As a test case in data and information reuse, the activity of costing is being taken as the focus of this research. Rolls Royce has recently made good progress in the ability to predict component unit costs using parametric cost models to support product design. However, a number of difficult challenges remain, including the ability to easily, quickly and flexibly reuse cost information from previous projects, (such as the Trent XWB) to guide the emergence of new programs such as the future large engine and open rotor programme (Wiseall, 2010). This is especially true in the early stages of design. A new engine project could have a new team with a new Chief Cost Engineer. This scenario is most often the reality, since many people from the previous projects move on to new roles. The situation is further compounded when predictions are needed quickly for either novel components, or if significant scaling is required. "An extreme example of parametric cost estimating and product design was when Sir Ralph Robins was able to commit the company to this ultimately successful venture, Gulfstream Aerospace, knowing that the proposed engine (Rolls-Royce Tay series) would draw heavily on the existing product base whose costs and performance he understood. In fact the RB 183 Mk 555 Spey design provided the basic core and gearbox and the RB211-535E4 provided a low pressure system and fan design for the new engine"(Scanlan et al., 2006).

This research therefore primarily aims to understand the issues pertaining the activity of data reuse for predicting unit cost of a gas turbine engine, with a subsequent focus on the primary issue that requires the attention in order to achieve it, i.e., finding 'the problem worth solving'.

1.1 Research Objectives

A data reuse capability to be able to predict the unit cost of a gas turbine is the focus of this research. This aim can be further broken down into the following research objectives:

1. Understand the requirements of data reuse for cost prediction.
2. Gap analysis of the existing data reuse framework.
3. Evolve the current methods and design a framework for data reuse that complies the basic requirements of an organisation like Rolls-Royce for activities like gas turbine unit cost prediction.

1.2 Report Structure and Research Contributions

This section will act as a guide to the rest of the thesis and summarise the primary content and highlight the contributions of every chapter that follows. To understand the state of the art, Chapter 2 is a review of the literature to understand and define cost. With the primary focus of being able to predict cost, various data reuse techniques are explained. Considering the need to be able to quickly reuse data, data mining was concluded to be the method of choice and it is further discussed in detail, examining the procedure and challenges involved. To understand how these challenges take form in the context of cost, chapter 3 reviews how the process of costing fits in with the bidding process of Rolls-Royce. This leads to the identification of the issue of data storage as being heterogenous and sparse. Taking these into consideration a high level framework is formulated to support quick costing. Consequently, in chapter 4 a method to extract data from existing documents for reuse is defined and demonstrated. However, a strategy to make the extracted data reusable is understood to be necessary and the subsequent discussion notes the requirement for a system that can support creation, storage and reuse of data and the need for a standard in data model to make data reusable. The major research contribution from this chapter is the formulation of the research question:

> ‘What is an appropriate framework for data reuse in an organisation to support a broad data reuse activity ?’

Chapter 5 then leads to discuss and review the concept called Linked Data. Linked Data is explored as a plausible solution to help machines automatically recognise and reuse data. A study into how linked data can support data mining is then examined. Further detailed discussion follows and reveals that even though linked data can be hosted on the World Wide Web, the issue of sparseness in the location of data is noted to be unsolved. Subsequently, chapter 6 explores the concept of semantic web for its perception and what it is. A case study is carried out to see how semantic web technologies could be applied to answer a question using the data from multiple sources.

The research contribution of chapter 5 is that that it opens up the view of how a web of data services is something that needs to be envisioned, along with the need to use the World Wide Web (Web) to support the creation of a web of data. The principles formed from analysing the needs and expectations, are then used to describe an evolution of the Web to form World Wide Information (WWI) Web.

Hitherto, having reviewed a wide spectrum of topics the consolidation of the understanding was deemed necessary to find the answer to the research question. Chapter 7 is, therefore, an analysis of the discussion so far. The notable research contribution of this chapter is the establishment of the need for a URI to reflect its provenance and description of the need for a framework that can support the existence of a web of data in an organisation. The specific areas of focus that this framework needs to help overcome to create the web of linked data were concluded to be (1) URI discovery and reuse through provenance, (2) URI ownership, (3) URI maintenance, (4) Ability to pull data by dereferencing a URI and (5) Ability to discover data through pull.

The major research contribution presented in chapter 8 is the understanding of 'security by privacy' and the formalisation of the concept and the principles of the proposed framework of WWI. Chapter 8 draws from the experience at Rolls-Royce and notes the need for security by ensuring privacy as the key requirement of data reuse in an organisation. The WWI framework is then described and detailed as a solution to the challenges and issues identified. Chapter 9 then leads to discuss the practical challenges, of reusing data from the web of data, in addition to those imposed by the sparseness and 'security by privacy'. An analyses of WWI framework for reuse shows the impact it can have on reuse of data especially in solving the challenge of sparseness in the location of data and how it can support the activity in a corporate environment. However, the conventional reuse techniques were found to be not useful to preserve the 'security by privacy' of the WWI framework due to the inherent need for techniques to know the location of data. The major research contribution presented in chapter 9 is a new technique of querying data known as 'domino request' that utilises the inherent structure of WWI is introduced and further demonstrated through a use case. Subsequently, a practical case study is presented and discussed on how WWI can handle access and provenance when multiple stakeholders are applicable to the same dataset. In addition, the case study also presents how useful the application of WWI can be to a world of internet of things due to the individuality provided by ensuring 'security by privacy'.

The thesis concludes by summarising the document with significant attention given to details that help expose the limitations that the proposed solution may impose due the various assumptions that have been during the course of this research, this is presented in Chapter 10. Chapter 10 concludes with a discussion and a list of the future work that needs to be carried out. The major contributions of the thesis were also listed and in summary are :

1. a new method for creation of Linked Data by consolidating one's data around their Identity,
2. a new way of referencing and dereferencing a URI (Uniform Resource Identifier),
3. a new method to hold provenance information of linked data by embedding it into the URI structure,
4. a new family of methods for retrieving data, known as 'domino requests', from multiple sources where there is no requirement to maintain a catalogue of the source locations,
5. demonstrated how WWI helps to create the paradigm of 'locally distributed data and centrally distributed applications' with respect to the data owner in an internet of things environment for privacy while being able to apply the respective access control policies from multiple stakeholders.

Chapter 2

Data Reuse & Cost: A Literature Review

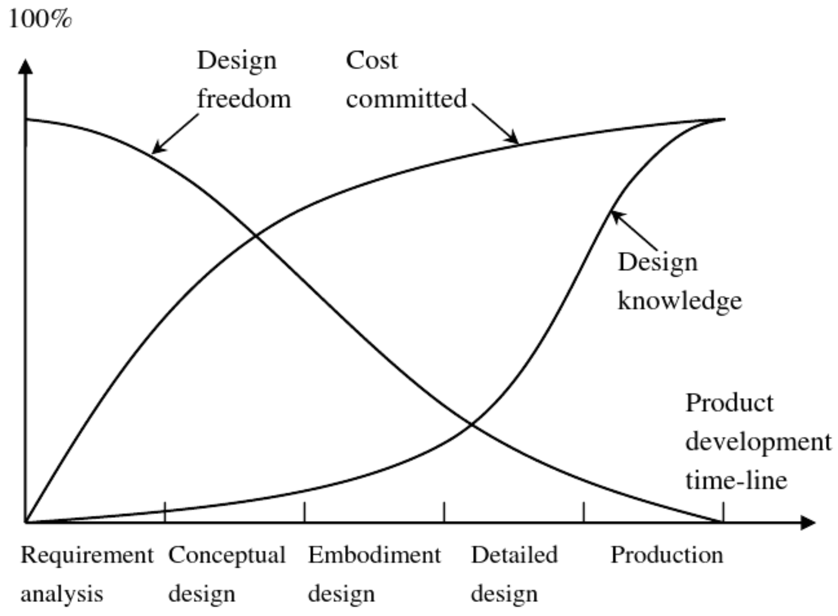


Figure 2.1: Product life cycle viewpoint of design freedom, product cost and knowledge availability (Ong, 2008).

Support at the right time and in the right form is very critical especially in the preliminary design stage where a significant amount of knowledge employed is gained from design reuse (Wiseall, 2010). Figure 2.1 illustrates the relative availability of product cost knowledge and the design freedom from a life cycle point of view. The current global market is characterised by high competition, where the success or survival depends on how fast the manufacturers are able to deliver their products with diversity, high quality, and environment compliance at a low cost, as noted by Ong (2008). This has increased the appetite to use robust design systems among the designers, like computer-aided design (CAD), computer-aided manufacturing (CAM), Product data management (PDM), expert systems, etc. There has also been noted advances in computer based collaboration environment for design and manufacturing. These computer aided systems have opened up an opportunity to have a computer aided design reuse systems, but Rezayat (2000) described the situation as “the age where the designers drown in data but thirsty for knowledge”. Since the motive of the research started with an aim to attain cost knowledge from information reuse, the first step is to understand the various ways in which design information can be reused (Crowder et al., 2003).

2.1 Design Reuse Systems and Enabling Tools

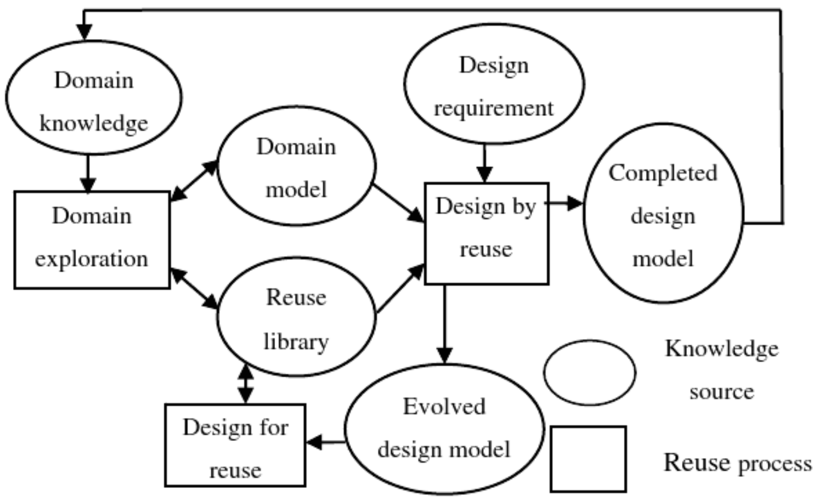


Figure 2.2: A design reuse process model (Duffy et al., 1995).

Design reuse utilises the existing technologies to address a new design problem and Figure 2.2 illustrates a design reuse process model conceptualised by Duffy et al. (1995). The reuse can be of the product itself at the end of its life or reuse of existing manufacturing resources or reuse of product information and design knowledge. Sivaloganathan and Shahin (1999) defines the ultimate aim of design reuse as being able to assist designer in developing new products that maximises the ‘value’ of the designed product minimising the ‘cost’. Achieving this ultimate aim can be very beneficial since according to Ullman (1997) about 75% of the manufacturing cost is committed at the product conceptual (preliminary) design stage.

Ong (2008) describes design reuse as an experience-oriented approach and as recognised, it has been adopted by designers/engineers consciously or sub-consciously. Various approaches that have been previously employed are:

- *Case-based-reasoning(CBR):* This approach retrieves and reuses similar designs based on the new design requirements. Watson (1999) called CBR an approach rather than a technology and he described the procedure consisting of four major steps, as illustrated in Figure 2.3 : retrieve, repair, reuse and retain. Ong (2008) appreciates that CBR covers all the significant aspects of design reuse but stresses that it is more concerned with the selection of modification of instances making it more suitable for variant design.

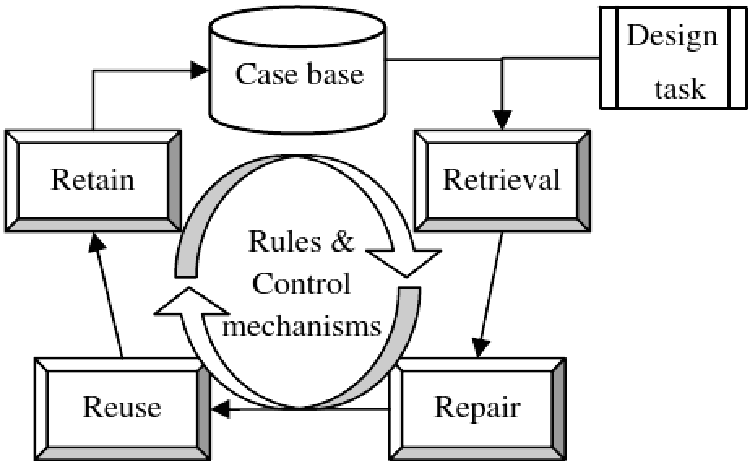


Figure 2.3: A case-based reasoning process model (Ong, 2008).

- *Catalog-based-design:* Also known as component-based design, is based on the establishment of a catalog of previous designs which is then reused. The reusability though depends on how well the catalog is structured. This was found to be an issue by Ong (2008) on the ability to apply this approach in a dynamic design environment.

- *Expert-systems:* Rychener (1988) and Anderson et al. (1985) defines expert systems as computer program with domain-specific knowledge of one or few human experts, which is then processed to solve the same way a human expert would do. This approach is knowledge intensive and is more suited to knowledge capture during design process.

- *TRIZ:* A methodology developed by Altshuller (1984), former Soviet engineer, which is based on his study of patents. TRIZ can be considered as a knowledge base, tool set, and methodology for generating innovative ideas and solution. As Savransky (2000) noted it is more of a problem solving tool since it provides solutions to overcome contradictions which are the driving force of inventions.

These are the major design reuse mechanisms. There are a few other approaches where the reusability is kept in mind, when the product is designed, in the first place. Modular design and adaptable design are two approaches that follow this philosophy. The former focuses on creating a design which is essentially a combination of a set of building blocks (modules), where the modules are the elements that are designed to be reusable. Adaptable design is based on the philosophy that a new design should be able to extend its service/utility to address new requirements. Ong (2008) appraised these various approaches and apprises that these can been made computable by various approaches like machine learning, data mining, design structure matrix and artificial neural network (ANN), which are data-intensive. Before exploring the possibility of using these methods to predict cost it is important to understand what cost is.

2.2 Introduction to Cost

Cost is an important decision making element in a design process. Tammineni (2007) has shown the importance of cost in aerospace engineering and Keane and Nair (2005) represented the importance of cost in their design spiral as in Figure 2.4. Tammineni (2007) defines cost as "the resource spent for the achievement of an objective", where the objective can be a product or a service. Brinke (2002), Putnam (1978), and Dean (1995) described cost to be the measure of work in monetary units associated with human endeavour. For the purpose of this research cost is defined as the net monetary value of resources consumed to make a product or achieve a state.

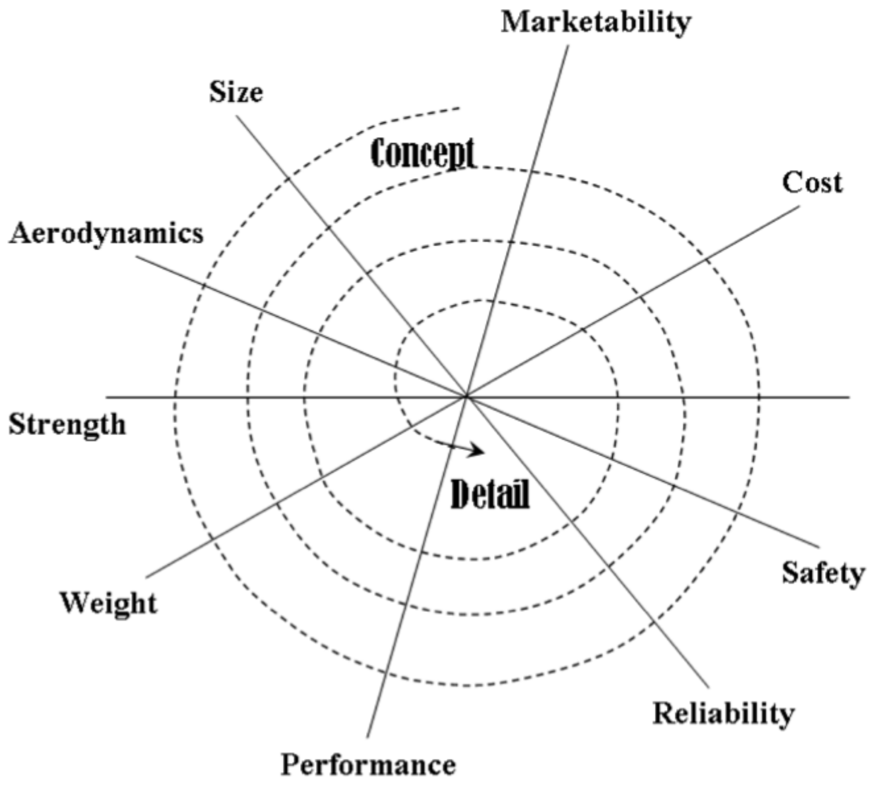


Figure 2.4: The design spiral by Keane and Nair (2005).

Tammineni (2007) has clearly explained the various useful classifications that facilitate a better perception of cost:

- *Direct and indirect costs* are based on the whether the cost can be identified and measured with respect to an objective (Shuford, 1995; Cooper and Kaplan, 1991).
- *Fixed and variable costs* are based on the direct relation of the cost to the volume of production (Curan et al., 2004; Colin, 2005).
- *Non-recurring and recurring costs* are based on whether the cost incurred is capital expenditure incurred prior to first production or is repetitive expenditure during the repetitive production phase and lifecycle (Tammineni, 2007; Curan et al., 2004; Stewart, 1995).
- *Relevant and irrelevant costs* are based on whether any future costs will be incurred based on a decision (Colin, 2005).
- *Incremental and marginal costs*, according to (Tammineni, 2007) "incremental cost refers to the difference between the two corresponding activities under each alternative considered, while marginal cost is incurred by adding one extra unit of output".

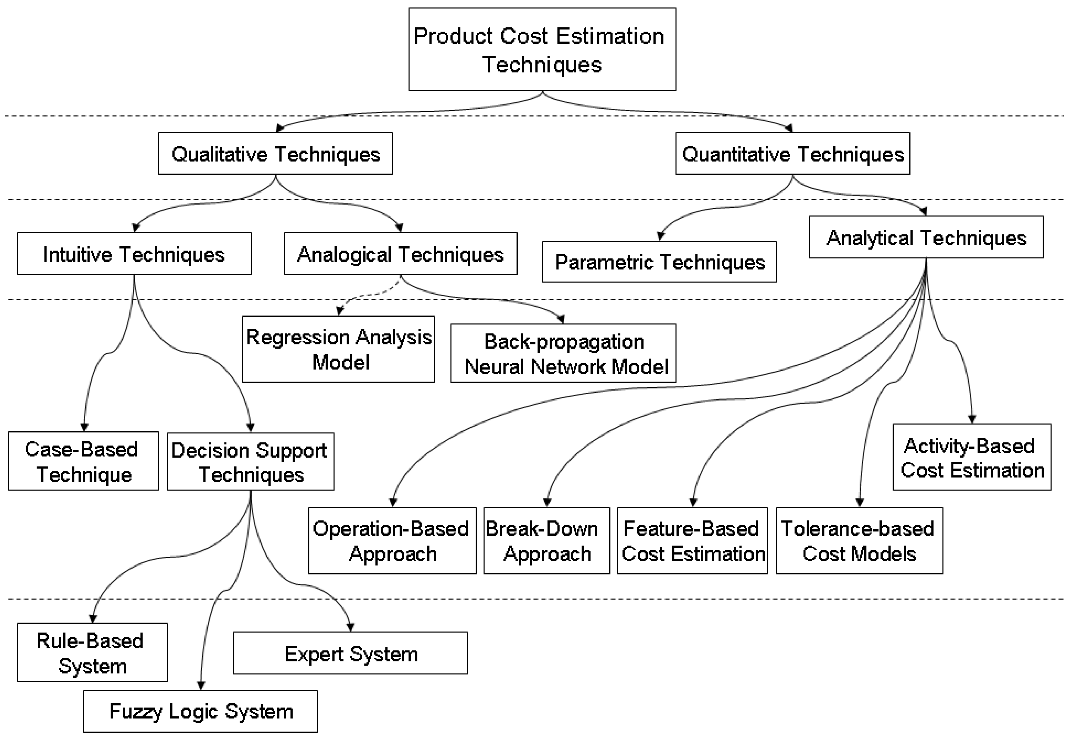


Figure 2.5: Product cost estimation techniques by Niazi et al. (2006).

2.3 Cost Estimation: Methodology and Tools

Cost estimation came to the focus during the Second World War as a tool to control cost, with Wight (1936) describing the various factors that affected the cost of an airplane. Niazi et al. (2006) describes the various product cost estimation techniques in Figure 2.5. Younossi et al. (2002) explains that there are three major cost estimation methods:

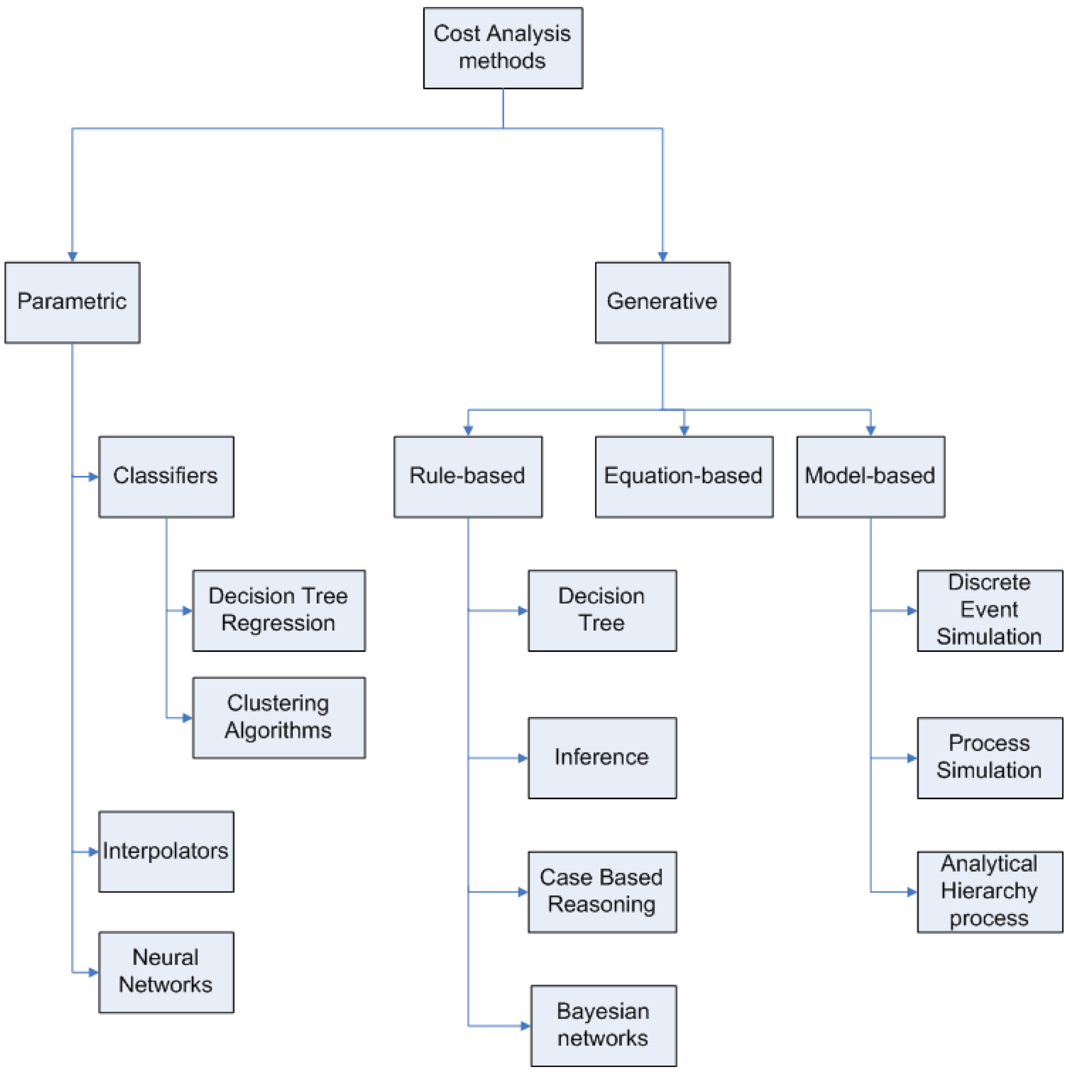


Figure 2.6: Cost tool classification hierarchy (Scanlan et al., 2005).

- *Bottom-up Method* relies on detailed engineering analysis and calculations to calculate an estimate. This is a time consuming process and a lot of time has to be spend to generate conceptual design and corresponding cost estimate. Younossi et al. (2002) argues that the user has to be an expert in the design of the technology to use it.

- *Estimate by analogy* is an approach where only the changes or differences of the new design compared to a similar design is estimated. This is a quick approach but the analyst must have a thorough knowledge of the applicable technology.
- *Parametric approach* is based on statistical technique that attempts to explain the cost drivers, i.e., the factors that significantly affect the cost as explained by Roy et al. (2001), as a function of several other variables such as engine characteristics (e.g., weight, size).

(Scanlan et al., 2005) illustrates in Figure 2.6 how various computational methods could be categorised based on the costing approach taken. The broad categorisation is based on whether the approach is parametric or generative in nature.

In summary, there are various ways to categorise the numerous methods depending on what the intended outcome one is, or based on what means one is required to take. Though it is to be noted that, it is not necessary to make a choice from these various methods presented. The solution could rather well also be a combination of a few.

Considering the possibility that an engineering design company would have a lot of data, a data reuse technique is what this research is looking for. The methods to reuse data (from a computer science perspective) is broadly known as data mining and therefore, is now important to understand the possibilities and limitations of it.

2.4 Data mining

Availability of 'prime' data exposes the opportunity for a company to make itself 'smarter'; The data can be used to find and understand complex relationships between various parameters, which in turn could support decision making activities. Data mining is a term used to describe this activity of identifying characteristic relationships or trends within a dataset. This understanding of relationships/trends could then be used as a mathematical model to predict and infer an outcome dependent on these parameters.

There are other commonly quoted approaches to define Data mining:

- "Data mining is the non-trivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data."(Fayyad et al., 1996)
- "Data mining is the process by which accurate and previously unknown information is extracted from large volumes of data."(Lee and Siau, 2001)
- "Data mining is the process of extracting previously unknown, comprehensible, and actionable information from large databases and using it to make crucial business decisions." (Gatnar and Rozmus, 2005)

- "Data mining is a set of methods used in the knowledge discovery process to distinguish previously unknown relationships and patterns within data." (Gatnar and Rozmus, 2005)
- "Data mining is the search for valuable information in large volumes of data. It is the process of nontrivial extraction of implicit, previously unknown and potentially useful information such as knowledge rules, constraints, and regularities from data stored in repositories using pattern recognition technologies as well as statistical and mathematical techniques."(King, 2006)
- "A technique to derive knowledge from data, to establish how parameters in a process are related, a method to assist in the understanding and visualisation of complex data."(King, 2006)
- "An analytic process to explore data (typically business or market related) in search of consistent patterns and/or relationships between variables, and then to validate the findings by applying the detected patterns to new subsets of data."(King, 2006)

2.4.1 Data mining: The process

Although Data mining algorithms are at the heart of Data mining process, it constitutes only about 20% of the entire process (Hernandez, 2008). Data mining consists of five major steps with the first and most important step being identification and understanding of the goals and the application domain. Step 2, involves selecting, collating, and integrating the data set stored in various databases. This is carried out considering the attributes of the target data set, that is of interest to the decision maker. Once the data has been collated, the step 3 involves pre-processing of the data. This includes data transformation, data cleaning, and handling of missing/incorrect data. The hypothesis is then developed in step 4 and the appropriate data mining algorithm is then chosen for extraction of patterns. This is then followed by the final step, where the extracted patterns are interpreted and validated using a test data set. Some of these steps would have to be reiterated before extracted knowledge can be used for productive decision making as in Figure 2.7.

A data set for any data mining activity can be described by variables and cases. Cases are the different records whose attributes are described by the variables. The data mining process can have different outcomes depending on the goals of carrying out the activity. Braha (2001) noted that most of the goals fall into one of the following main categories:

- *Data-processing* is concerned with selection, integration, filtration, sampling, cleaning and/or transformation of data.

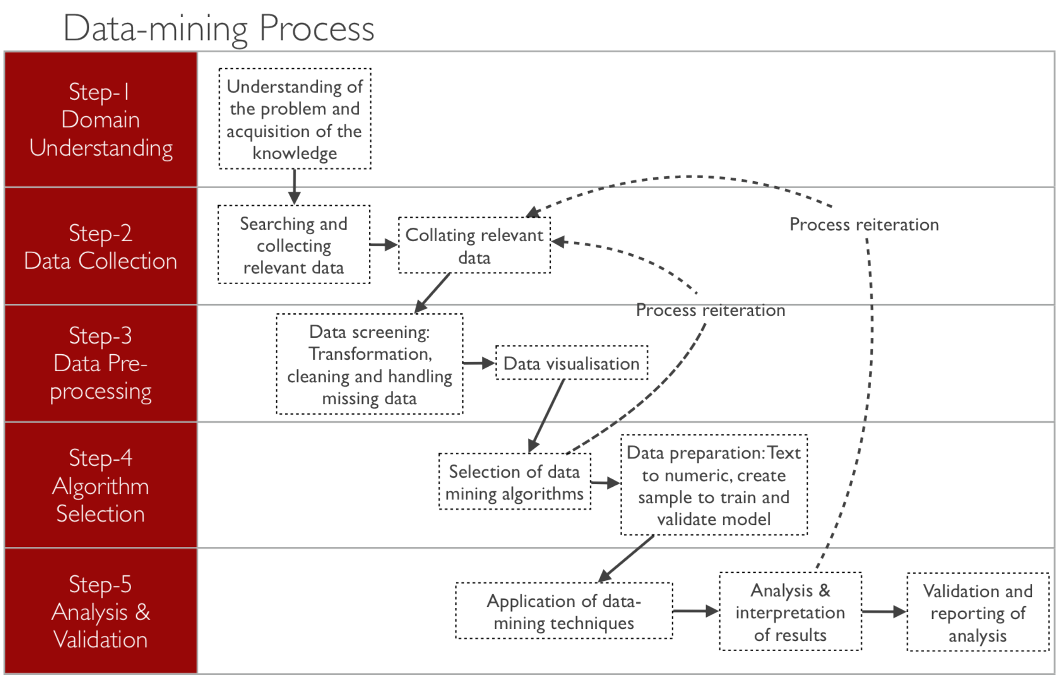

Data-mining Process
Step-1 Domain Understanding
Step-2 Data Collection
Step-3 Data Pre-processing
Step-4 Algorithm Selection
Step-5 Analysis & Validation
Understanding of the problem and acquisition of the knowledge
Searching and collecting relevant data
Collating relevant data
Process reiteration
Process reiteration
Data screening: Transformation, cleaning and handling missing data
Data visualisation
Selection of data mining algorithms
Data preparation: Text to numeric, create sample to train and validate model
Application of data-mining techniques
Analysis & interpretation of results
Validation and reporting of analysis

Figure 2.7: The common steps in the use of data-mining (Hernandez, 2008).

- *Verification* involves testing preconceived hypothesis and fitting of models to data.
- *Regression* means the analysis of relationship between attribute values within the same case and the automatic creation of a model that can predict the attribute values for future cases.
- *Classification* involves assigning a case to predetermined classes.
- *Clustering* focuses on partitioning a set of cases based on similarity in characteristics described by the variables.
- *Association* involves the discovery of rules that associate one variable to another set of variables in the data set.
- *Sequential-pattern-Analysis* focuses on the decision makers attempt to convey the discovered knowledge in an understandable form.
- *Deviation-Analysis* is used to detect deviation over time, from the mean, and between the observed value and a reference value.

2.4.2 Challenges in Data mining

Data mining being a data intensive activity and there are a number of factors that can hinder the process. First and foremost, in most cases data is not readily available. Collating data together from multiple sources can be a tedious task which could be further hindered by copyright, privacy and security aspects of the data being used. Kaufman and Michalski (1996) states that most of the data mining activities assume that all the data are located in the same database and the concepts that is to be learned have a precise description that does not change over time; But these assumptions do not hold always and the issues that one could face are:

1. *Learning from incorrect data:* Learning from examples that contain noise data (e.g., (Michalski et al., 1991; Quinlan, 1990)). This is an unavoidable situation for a real world problem.
2. *Learning from incomplete data* Learning from examples in which the values of some variables are unknown (e.g., (Lakshminarayan et al., 1996; Dontas and Zemankova, 1990)).
3. *Learning from distributed data* Learning from separate collections of data that must be brought together if the patterns within them are to be exposed (e.g., (Ribeiro et al., 1995)).
4. *Learning drifting or evolving concepts* Learning concepts that are not stable but changing over time, randomly or in a certain general direction (e.g., (Widmer and Kubat, 1996)).
5. *Learning concepts from data arriving over time* Incremental learning to account for the new data to update the current hypothesis (e.g., (Maloof and Michalski, 1995)).
6. *Learning from biased data:* Learning from a data set that does not reflect the actual distribution of events (e.g., (Feelders, 1996)).
7. *Learning flexible concepts:* "Concepts that inherently lack precise definition and whose meaning is context-dependent; some ideas concerned with this topic include fuzzy sets (e.g., (Zadeh, 1965; Dubois et al., 1993)), two-tiered concept representations (e.g., (Bergadano et al., 1992)), and rough sets (e.g., (Ziarko, 1994; Slowinski, 1992))".
8. *Learning concepts at different levels of generality:* Learning descriptions that involve concepts from different levels of generalization hierarchies representing background knowledge (e.g., (Kaufman and Michalski, 1996)).
9. *Integrating qualitative and quantitative discovery:* Determining sets of equations that fit a given set of data points, and qualitative conditions for the application of these equations (e.g., (Falkenhainer and Michalski, 1990)).
10. *Qualitative prediction:* Discovering patterns in sequences or processes and using these patterns to qualitatively predict the possible continuation of the given sequences or processes (e.g., (Anderson et al., 1986)).

2.5 Summary

This review of the literature has given a basic understanding of the methods and tools that can be employed to reuse the information of a product to carry out a new design process. Numerous methods to obtain cost knowledge were also discussed. Considering the presence of vast amounts of data in Rolls-Royce, it was important to explore the opportunity of data mining (a parametric approach for information reuse). Data Mining being a data intensive process, the possible challenges were also discussed in Section 2.4.2. The next step is to use these various methodologies to come up with an approach where the designer is able to use the vast amounts of data available to support their need for cost knowledge without getting drowned in data. It should be noted that these varied methodologies discussed previously are more appropriate to a specific purpose to attain the maximum benefit since each methodology has different levels of accuracy. Therefore, to come up with the best possible solution it is important to understand the people who are in need of such cost knowledge.

Chapter 3

Data Reuse for Costing: Challenges & Framework

"We could develop the newer workforce much faster if we allowed them to access past information. This would give them broader exposure to the business and data for predicting current trends."

– A diversified industrial manufacturer (Lavelli et al., 2008)

On average, a Rolls-Royce civil aircraft gas turbine engine consists of about 18,000 parts. Hence, vast amounts of data are generated from the processes of design, manufacturing, sales, and maintenance of an engine. The reuse of such data through data mining processes and methods is beneficial, because it can be used to abstract the data into a reusable predictive model. Data mining is an activity which is used to construct, update, and validate a model that is adaptable to new data as and when new engine programme data becomes available (Daniel and Marsh, 2010).

Rolls-Royce has recently made good progress in the ability to predict component unit costs using parametric and geometry driven cost models to support detailed product design. However, in the early stages of design, the engine definition is less defined and so a number of difficult challenges remain in the ability to easily, quickly, and flexibly reuse cost information from previous projects to guide the emergence of new programmes. This is compounded with new design team members who are less familiar with past experiences, information sources, and key expertise (Wiseall, 2010). These issues have motivated the exploration of Data Mining processes and methods applied to predicting engine unit cost as an initial test case (Daniel and Marsh, 2010).

This Chapter, in Section 3.1, will first look at the definition of cost and places the importance of cost from the perspective of one who needs to know. The first major problem of not having enough cases is described in Section 3.2 and Section 3.3 describes the second problem of finding the relevant data. Subsequently, Section 3.4 discusses the new costing system.

3.1 Cost, Estimation and its Relevance

One now must be able to appreciate that there are various ways in which cost can be understood and differentiated and that it is done to understand its consequences in the most appropriate way. In this thesis, cost is interpreted as the net monetary value of resources consumed to make a product or achieve a state. In the context of Rolls-Royce, this research aims to predict the unit cost of an engine in the shortest time possible and so it is also important to understand how the quick prediction of unit cost can be of value to the company.

3.1.1 Cost from the Perspective of its Seeker

To acquire as many orders as possible, Rolls-Royce invests time examining the current and future market opportunities while building relationship with customers. This is a continuous process and forms the first stage of an engine's lifecycle (Bolton and Clegg, 2011). Its successful execution helps to anticipate potential business opportunities and future market needs.

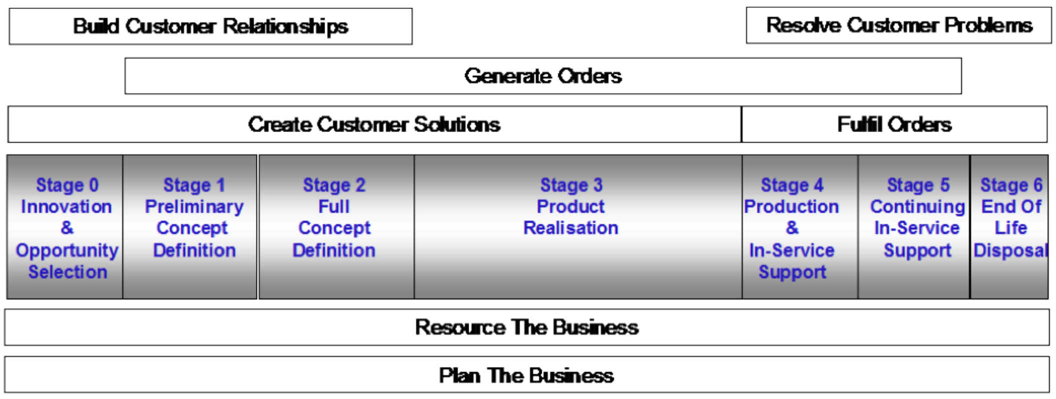
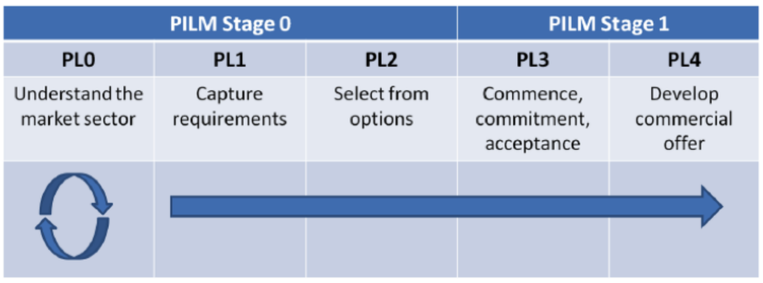

Build Customer Relationships
Resolve Customer Problems
Generate Orders
Create Customer Solutions
Fulfil Orders
Stage 0 Innovation & Opportunity Selection
Stage 1 Preliminary Concept Definition
Stage 2 Full Concept Definition
Stage 3 Product Realisation
Stage 4 Production & In-Service Support
Stage 5 Continuing In-Service Support
Stage 6 End Of Life Disposal
Resource The Business
Plan The Business

Figure 3.1: PILM process depicting the six stages of an engine's lifecycle. (Courtesy of Rolls-Royce plc.)

Figure 3.1 represents the Product Introduction and Lifecycle Management (PILM) process depicting the six different stages of an engine's lifecycle (Bolton and Clegg, 2011). The New Product Introduction (NPI) process resides within stages 0-3. PILM stages 0 and 1 are again subdivided further into five different Planning Levels (PL0, PL1, PL2, PL3 & PL4), as revealed in Figure 3.2. PL0 is the stage where the markets are analysed continuously, while the requirements get captured in PL1, with the final concept selection in PL2.



PILM Stage 0			PILM Stage 1	
PL0	PL1	PL2	PL3	PL4
Understand the market sector	Capture requirements	Select from options	Commence, commitment, acceptance	Develop commercial offer

Figure 3.2: The five Planning Levels (PL) in PILM stage 0 & 1. (Valid as of June 2011, Courtesy of Rolls-Royce plc.)

A new business relationship starts with a Request for Information (RFI) from a potential customer. This occurs during PL2 and a reply is submitted within a few weeks, if not, days. On acceptance, it is then followed by a Request for Proposal (RFP) from the customer. When a RFI is received by Rolls-Royce, the CCE is the one responsible for creating a response and organising the bidding process in later stages (Bolton and Clegg, 2011). The RFI contains generic requirements from the customer like the thrust range (market), noise range, maximum weight, airframe type, engine location, and expected market launch. Looking from the point of view of CCE, the thought process can be studied from the rationale of six W's: who, when, where, what, how, and why (Kipling and Kliros, 1993). The first step of capturing the requirements can be viewed from the three questions:

1. Who is it for?
    - **(a)** Which aircraft manufacturer or airline is it for?
    - **(b)** Which market is it for? (Small, mid, and large)
2. When do they want it?
    - **(a)** When is the market launch?
    - **(b)** What is the time frame for the engine development?
3. Where do they want it? (Location of the engine on the airframe.)
    - **(a)** What type of airframe is it/ engine configuration?
    - **(b)** What are the operational conditions? (Noise, weight, emissions, etc.)

The next step would be the preliminary design selection process based on the answers from these 3 questions where a select collection of engines gets chosen for its suitability to the request. The final expected outcome of this process would be a pruned collection of engines or an engine. This process can be examined from the next three questions in continuation of the 6w activity:

4. What would they want?

    (a) Which type of engine?
    (b) What configuration?

5. How would it be made? (What type of components?)
6. Why would they want it so? (Why would this selection/choice of components be beneficial?)

At the final stage of this progression the closing question asked is: 'What does it cost?'. It is apparent that a value for cost is outlined only as the 7th question, meanwhile it is very important to comprehend that all these decisions does affect the final cost in succession. Keeping this in mind, it can understood that choosing the appropriate cost estimation methodology for predicting the unit cost of engine specific to a situation is important.

3.1.2 Cost Estimation

Cost estimation came to focus during the Second World War as a tool to control cost, with Wight (1936) describing the various factors that affected the cost of an airplane as previously discussed in Section 2.3. Bottom-up method, estimation by analogy, and parametric method were noted to be the three major categories to classify costing (Younossi et al., 2002).

As previously stated, in the early stages of design, the engine definition is less defined and so a number of difficult challenges remain in the ability to easily, quickly and flexibly reuse cost information from previous projects to guide the emergence of new programmes. This is compounded with new design team members who are less familiar with recent past experiences, information sources and key expertise.

Bottom-up method is a time consuming process and would require some previous experience or expertise on the product (Younossi et al., 2002). Such cost models would also be very expensive to maintain and adapt to a new design. Likewise estimation by analogy is only useful when a the new design doesn't have any major difference from the previous design. Even though this approach can be quicker, the accuracy of cost prediction could vary based on the expertise of the analyst. Meanwhile, the parametric approach since based on statistical techniques the cost can be valued as a function of several other variables that have enough correlation with it. The risk with this method though is that statistical correlation among variables doesn't necessarily mean causation. Nevertheless, this method can be significantly quicker to use and since its based on statistics there are metrics that can be used to help one gauge the significance of a prediction. These reasons have therefore motivated the exploration of Data Mining processes and methods (a parametric approach).

3.2 Predicting Unit Cost by Data Reuse

From the discussion in Section 3.1.1, one can understand that in the early stages of bidding process the cost prediction needs to be quick in order to submit a reply in response to the RFI as early as possible. The importance is not just on the predicted value of unit cost, but how confident CCE is about the prediction. The mathematical measurement of confidence can be represented by confidence or uncertainty intervals (Loftus, 1994).

As for the simplest of all the analyses that can be done with the data points, a line is fit through these data points. Figure 3.3 and Figure 3.4 shows the engine cost plotted against maximum take-off thrust and basic engine weight respectively. The analyses are being carried out on a dataset of 15 in-service engines. The standard deviation was found to be in millions of British Pounds, which makes simple statistical analyses not sufficient for a reasonable cost prediction by any form of regression. Considering the total number of types of production engines that can be used are approximately 20, this also brings to light the issue that the numbers of cases are few that can be subjected to data analyses.

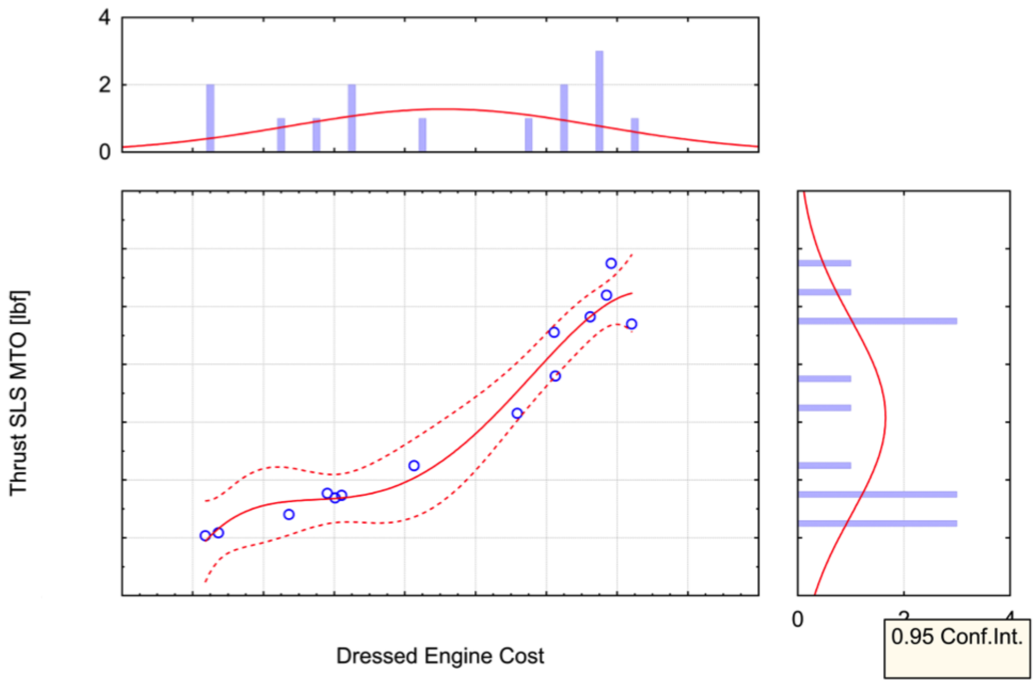


Figure 3.3: A plot of Maximum take off thrust [lbf] vs Engine Cost with the dotted red line representing the 0.95 confidence interval. The subplot on top represents the number of engines at a particular cost value. While, the subplot to the right represents the number of engines at a particular thrust value.

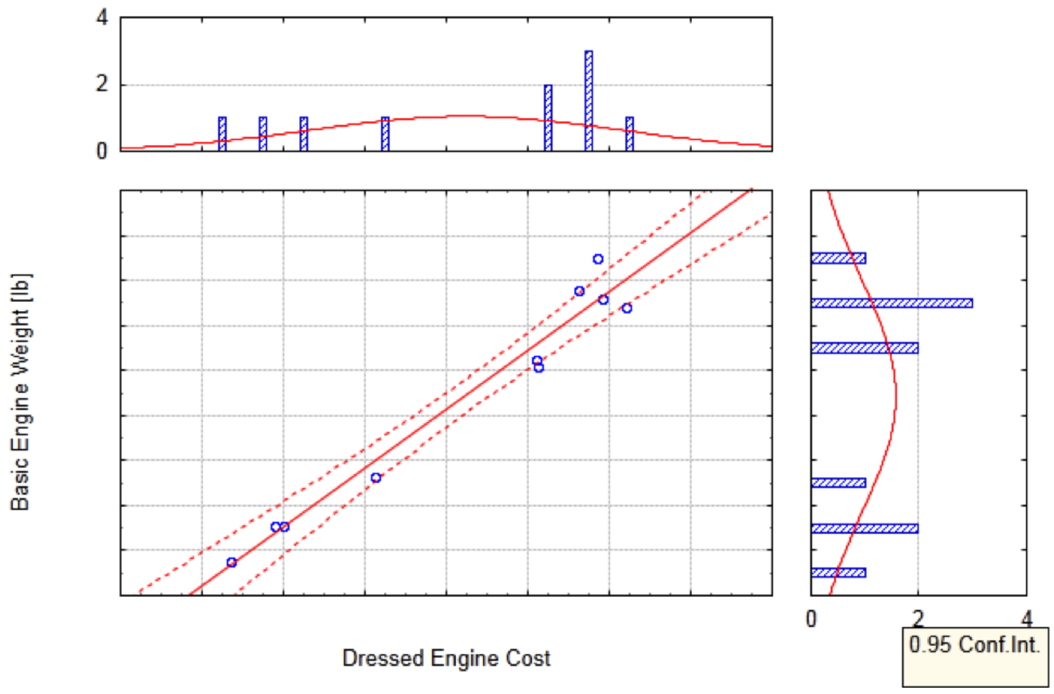


Figure 3.4: A plot of Basic Engine Weight [lb] vs Engine Cost with the dotted red line representing the 0.95 confidence interval. The subplot on top represents the number of engines at a particular cost value. While, the subplot to the right represents the number of engines at a particular weight value.

Figure 3.5 shows the number of engines against the distribution of engine unit cost and it can be noted that there is clustering of data points towards the edges of the cost value range represented in the Figure. This clustering is due to the basic structural differences in the engines, the data represented in Figure 3.5 contains both 2-shaft and 3-shaft engines. This difference is more evident in Figure 3.6, where all the 2 shaft engines could be identified as statistical outliers. Removing the two shaft engines, the standard deviation of cost for just the three shaft engines was reduced to a quarter compared to that of the entire dataset.

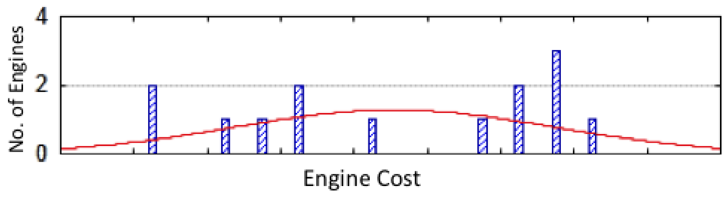


Figure 3.5: A plot that represents the Number of engines against the engine cost value.

For a gas turbine there is no linear relationship between the cost and the performance parameters, i.e., engines with similar performance (in terms of overall performance parameters) do not necessarily share similarly performing individual components. This is deductible from Brooks (2011) statement, "Generally, some of the factors that affect gas turbine performance are constrained. The planned site location and the plant configuration (such as simple- or combined-cycle) determine most of these factors. In the event additional output is needed, several possibilities to enhance performance may be considered".

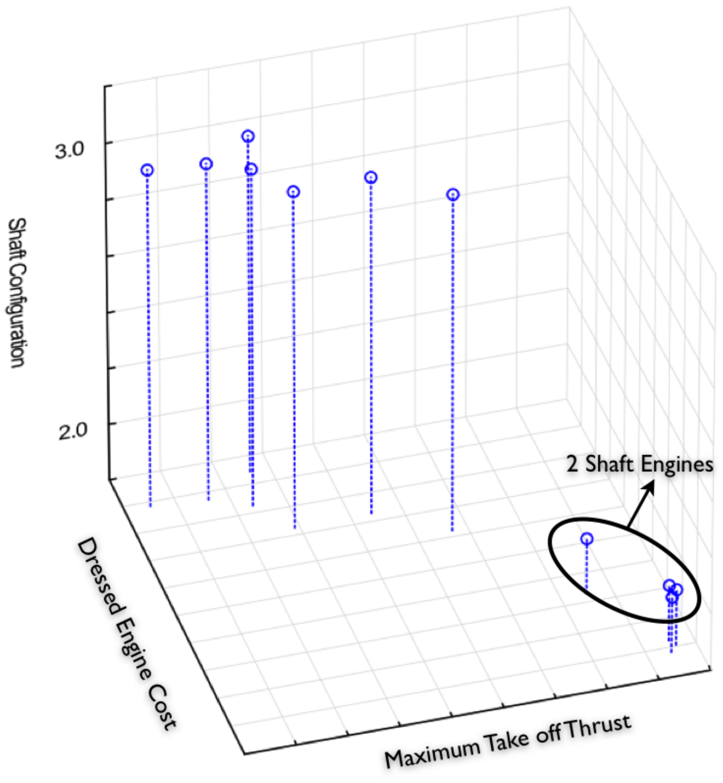


Figure 3.6: A plot of Shaft configuration vs. Cost vs. Maximum take-off thrust of engines.

It can be understood that association of cost to parameters that define an engine needs to be explored. From the fact that an engine on average consists of 18000 parts, there is the existence of a huge variety of data within the company for every engine and so relationships between the cost and other parameters could be established using data mining techniques.

3.3 Challenges in Data Reuse

Use of various types of data related to an engine seems to be a requirement in order to establish the best possible unit cost prediction model. So one needs to analyse the availability and accessibility of the data within the company.

3.3.1 Data Availability

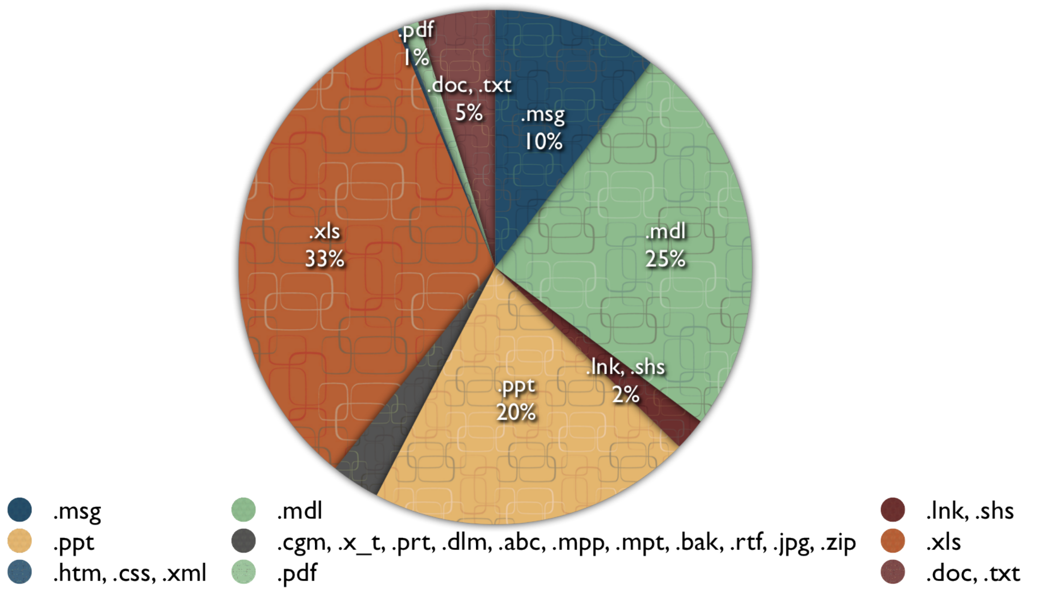


Figure 3.7: Percentage of data files in an engine cost dataset by file type.

Having carried out the investigation on availability of data within Rolls-Royce, it was noted that there is existence of a wide variety of data. The data storage mediums were found to be heterogeneous and formats non-standard. From Figure 3.7, which represents the analysis of the data files from an engine program cost dataset, heavy use of Microsoft Office documents, PDF's and emails are evident. Semi-structured and structured data can found locked away in these localised storage mediums created by the fore mentioned desktop applications into different formats.

It is also important to keep in mind that there are certain sets of data, which are stored in centralised databases and can be accessed from a computer within the intranet. Such centralised database systems are generally established in order to monitor the flow of data. They help to exercise the ability to control the changes applied to a certain dataset and also to limit the access to a specified group of people within the company who have been authorised. Existence of such centralised database systems also helps people to identify the existence of certain datasets and would be able to use the data once authorised.

3.3.2 Data Accessibility

Excel spreadsheets were seen in use as localised databases, due to the possibility of linking different Microsoft Excel files together and the ability to get the data updated by sending the relevant spreadsheets to the corresponding people via email. Its popularity was high due to the widespread availability and people's familiarity with the tool.

Spreadsheets can be especially useful when one has created a dataset and wants to share it with a specific group of people, compared to centralised databases where prior permissions are required. Databases serve the purpose of sharing-with-monitoring that can be useful for the datasets that are being used by many people or when data creator is not necessarily specifically aware of its end users. A database also ensures that the data creator complied with the database requirements, which made sharing of data simpler.

Though sharing of data using a database is easier, reusing the data was found to be tedious. One had to download the relevant data onto a spreadsheet manually. Then the relevant data can be extracted using a visual basic (VB) script or by writing an application programming interface (API), but this process was time-consuming and tedious. Considering not every engineer would be well versed in programming or wouldn't be interested in writing an API, it is necessary to find other ways of data extraction.

3.3.3 From the Data User's Perspective

Even though sparse nature and heterogeneous forms of storage are an issue, it is important to place the data reuse procedure in the perspective of the person who is trying to collate the data. The procedure of data reuse can be broken down into the following sequential steps in order to be able to approach the solution better:

1. Clearly understand the question that needs to be answered. (Example question: What is the cost of Trent XWB-93k?)
2. Identify the data locations and acquire them.
3. Recognise and clean the relevant data.
4. Feed the data or statistics that describes them to the statistical model building tool and validate the model.

Step-1 is the primary and an important step as steps 2 & 3 are dependent on the step-1, since collating the required dataset from sparse data is vital. Step 1 also describes the motivation and defines the purpose of pursuing this activity. When these steps are compared to the general steps of any data mining activity, steps 1-3 takes up about 80% of the effort (Daniel and Marsh, 2010). So any change in the question/motivation (step-1) can be daunting, since all the subsequent steps would have to be repeated.

3.4 An Approach to Costing with Data Reuse

As noted from the Figure 3.7 it can be seen that most of the data is locked away in numerous formats. The data needs to be extracted from these heterogeneous formats. Moreover, the nature of availability of data is sparse.

3.4.1 EP3: A data acquisition activity

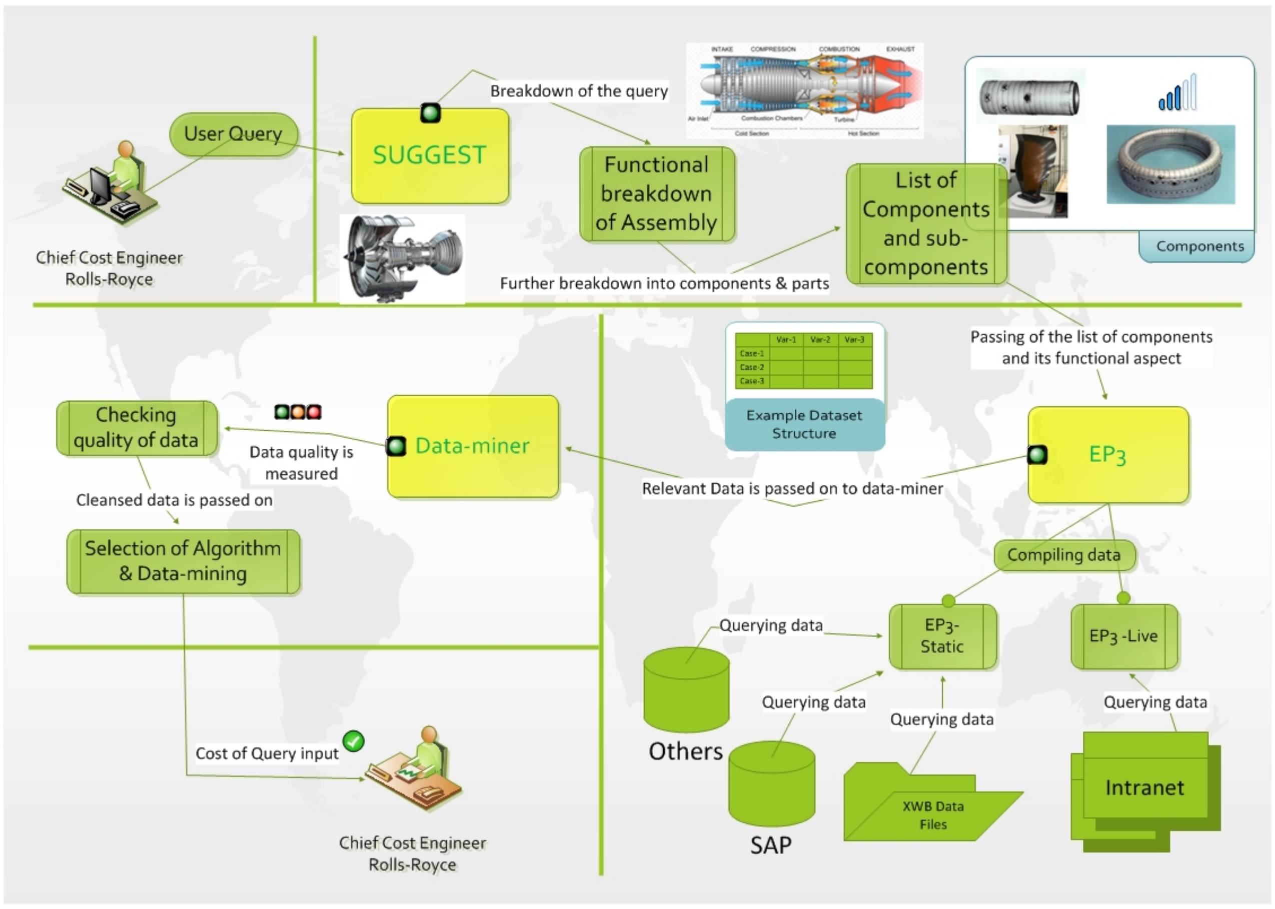


Figure 3.8: Illustration of a costing system (SUGGEST), supported by EP3.

In order to understand the reusability of data, a data acquisition activity was established and was called EP3 (Engineering Product Price Predictor). EP3 primarily has two modules, namely, EP3-static and EP3-live. EP3-static has two sub-modules. Sub-module 1 to extract data that had been store in a tabulated format. Sub-module 2 to act as a wrapper and as a connector to the databases. While EP3-live is a web crawler & scraper and is aided by an in-built browser. EP3-live is to extract the data that is hosted in the company intranet websites.

The aim of EP3 is to help complete the second step from the steps followed when reusing data, as described previously. The step 3 involves identifying and cleaning the relevant data from the tables of data that have been extracted from various documents, websites and databases. Further, linked data concept can be employed here in order to identify and bring the relevant data together with the help of existing Rolls-Royce ontology. More can be read about linked data in Chapter 5.

3.4.2 SUGGEST: A costing framework

In order to establish a costing framework, a module called SUGGEST is added ahead of EP3 as illustrated in Figure 3.8. SUGGEST can be described as a product definition browser which uses the Rolls-Royce engine ontology in order to guide the user to carry out the design selection process. This addition is in consideration with the discussion in Section 2.2, which shows the importance of cost being the last of the priorities in the design selection process. SUGGEST aims to emulate the design selection process with the user being guided by the cost, where the product definition takes more shape with every step in the process. SUGGEST is actively supported by EP3 in acquiring the required data, which is then processed by a data mining tool for predicting the cost.

By employing EP3 supported by linked data, the steps required to reuse data will be:

0. Identify the data locations and convert to reusable data. (This step is done only once and so denoted as step 0.)
1. Clearly understand the question that needs to be answered. (Example question: What is the cost of Trent XWB-93k?)
2. Recognise and clean the relevant data.
3. Feed the data, or statistics that describes them, to the statistical model building tool and validate the model.

In the linked data world, the data needs to be converted into a standard linked data format only once and that completes the step of recognising the data, this helps the person to complete a data reuse procedure quicker with fewer steps in case his/her motivation changes. Finally, the step of bringing the relevant data together is further made easier with the use of linked data, since the required data could be brought together potentially by a single SPARQL query (Patni and Henson, 2010) (SPARQL being a W3C standard for querying linked data).

3.5 Summary

For a company like Rolls-Royce, cost is an important engineering variable. In this chapter, the cost knowledge that's to be acquired was unit cost of engine and thus cost was interpreted as the net monetary value of resources consumed to make a product or achieve a state.

The unit cost prediction is intended to be used in the early stages of a design process and Section 3.3.1 explains the relevance of quick costing in relation to the bidding process. Considering the engine definition is less defined at this stage, from the discussion on various costing methods, data mining (parametric costing) was chosen to be the appropriate one. Though, there are two main hurdles that need to be overcome. Section 3.2 describes the first issue of not having enough data points, i.e., enough number of engines for a reasonably accurate statistical prediction. The need to find a relationship between as many features that define an engine through data mining was also established in Section 3.2. Section 3.3, analyses the availability and accessibility of the data and Figure 3.7 represents a pie chart with the percentage of all the formats in which the data is contained. Excessive use of Microsoft Excel and PowerPoint were noted. While considering the rest of the data was contained in similar data stores of numerous engineers and databases, it was concluded that the nature of data availability is sparse and the storage mediums heterogeneous. Therefore, when establishing a process for costing, it is equally important that one takes in consideration the reusability of existing data.

Section 3.3.3 breaks the whole process of reuse of data for data mining into four major steps of: a) Clearly understanding the question that needs to be answered. b) Identifying the source of data and acquiring them, c) Recognising the relevant data and cleaning them, and d) Creating the predictive model by feeding the data, or the statistics that describe them, into a data mining tool and then validating the model. It was argued that once the motivation in the reuse of data changed, all the four steps had to be repeated. In order to speed up this entire process, a costing system was suggested in Section 3.4, as in Figure 3.8. This costing system in short had two major parts, EP3 and SUGGEST. EP3 is a data acquisition procedure, where tools were developed to extract data locked in various formats like .doc, .ppt, .msg, .pdf, html and so on. The data once extracted were then converted into a standard format in accordance with linked data principles. Once the data was liberated from the various formats, the CCE could compile the necessary data required just from a single SPARQL query, with the help of SUGGEST. SUGGEST is introduced with an aim to emulate the design selection process where the user being guided by the cost.

Chapter 4

Data Reuse: From Sparse & Heterogenous Data Sources

"Reuse is something that is far easier to say than to do. Doing it requires both good design and very good documentation. Even when we see good design, which is still infrequently, we won't see the components reused without good documentation."

– D. L. Parnas (1994)

The previous chapter has described a framework to reuse data for costing as illustrated in Figure 3.8 and the four step procedure that needs to be followed to use this framework was described in Section 3.4.2. The underlying vision of the framework being that a data reuse activity can be carried out much quicker in fewer steps, if one is able to get the data that they required as quick as possible. The reuse of data with the framework is then mainly dependent on the final three steps (step 1 to step 3 in Section 3.4.2), since only these steps need to be repeated for any change in motivation in data collation. The first step (step 0), however, needs to be considered to be as a continuous process by which all the existing and the new data needs to be identified and located.

4.1 Reusing Existing Data

Figure 3.7, which represents the analysis of the data files from an engine program cost dataset, a wide variety in use of storage mediums can be seen. Heavy use of Microsoft Office documents, PDF's and emails are evident. Moreover, these mediums do not have a common data storage standard, with a lot of semi-structured and structured data.

The primary challenge with reusing data for costing in Rolls-Royce, is being able to reuse the data from existing wealth of documents. The first major activity in this process is extracting data from these various documents. With one of the most popular model of representing data in these documents being tabular form, extraction such data will be the focus in this section.

4.1.1 Hypothesis

'Tabulated data can be extracted from PDF (.pdf), Microsoft Word(.doc, .docx), Microsoft Powerpoint (.ppt), webpage (.html) and email message (.msg) for reuse in a reusable format.'

4.1.2 The Metric

This experiment focuses on the ability to extract data and the metric that will be used to measure the success of the experiment will be its:

- *Recall Precision:* The recall precision of the experiment will be measured by counting the number of tables in the document and number of tables extracted after manual inspection to check that the extracted tables are exactly same as the ones in the document.
- *Repeatability:* The repeatability of this experiment will be measured by repeating the experiment and ensuring that the accuracy remains the same in both the attempts.

4.1.3 Implementation

The primary motive being to extract data form the list of documents, Figure 4.1 illustrates the steps that will be followed to extract the data. It consists of six major functions, namely:

A. Commander

B. Searcher

C. Analyser

D. Recurser

E. Parser

F. Storage

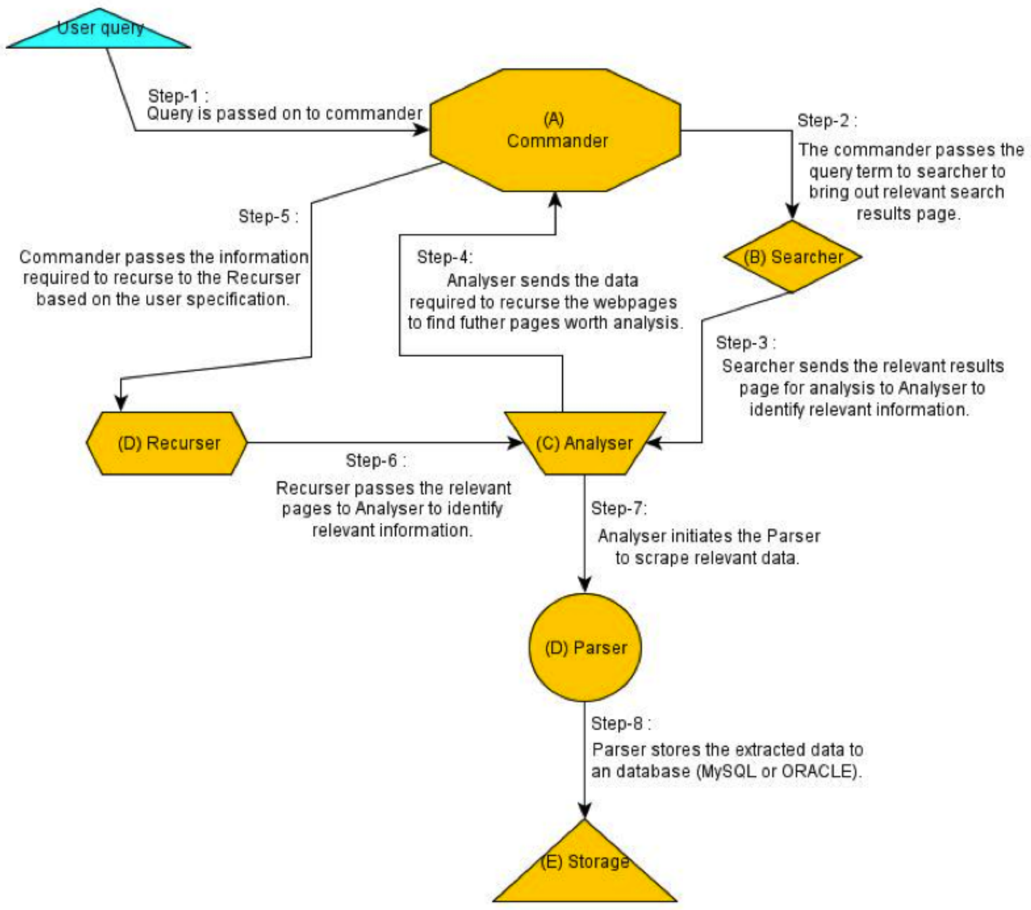

User query
Step-1 : Query is passed on to commander
(A) Commander
Step-2 : The commander passes the query term to searcher to bring out relevant search results page.
(B) Searcher
Step-3 : Searcher sends the relevant results page for analysis to Analyser to identify relevant information.
Step-4: Analyser sends the data required to recurse the webpages to find futher pages worth analysis.
Step-5 : Commander passes the information required to recurse to the Recurser based on the user specification.
(D) Recurser
(C) Analyser
Step-6 : Recurser passes the relevant pages to Analyser to identify relevant information.
Step-7: Analyser initiates the Parser to scrape relevant data.
(D) Parser
Step-8 : Parser stores the extracted data to an database (MySQL or ORACLE).
(E) Storage

Figure 4.1: Illustration of steps to complete the task of extraction of data.

The Commander (A) was written in python as the main program with which the user interacts and initiates the process of using the other functions. Based on the user query input, Searcher (B) is initiated to search the set of documents. The search function implemented was a simple text matcher, which matched the query keywords with the content of the documents, to demonstrate the use of the Seacher (B) functionality. The Documents retrieved by the Searcher (B) is then passed to the Analyser (C). The function of Analyser (C) is to find the relevant content which are then extracted by the Parser (E) which was implemented as a simple tabulated data extractor. Due to the availability of numerous tools, for the purpose of extracting tabulated content, the attempt to write a code to do the same was abandoned. After evaluation of various tool available for adaptability to various document formats, Solid PDF Tools[1] was chosen. The Analyser (C) was given the added function of looking for links to other files which would then be retrieved and followed by the Recurser (D) on command from the Commander (A). This is especially useful for web documents (.html) where more related content is provided by way of hyperlink. For every new document found by the Recurser (D), the Parser (E) is initiated to carry out data extraction procedure after confirmation of relevant data by

[1]www.soliddocuments.com/pdf/-to-pdf-a-converter/307/11

the Analyser (C). All the data extracted by the Parser (E) is then stored in a data base (Oracle[2], MySQL[3]) by the Storage (F) function.

From Figure 3.8 one can see that the framework of this implementation becomes part of EP3, where the EP3-Live (data extraction from web documents (.html)) differed from the execution of the implementation for other documents. Ep3-Live is carried out by the Commander (A) by initiating a different set of code by which the web document is loaded using Hypertext Transfer Protocol (HTTP)[4], which can then be subjected to extraction of tabulated data by the Parser (E) or extract data (non-tabulated) by making use the XML (extensible markup language)[5] markups as identifiers. This ability to extract data using identifiers for location of data on a web page was coded into the Parser (E) and was the addition on general implementation of EP3 (EP3-Static) that redefined it as EP3-Live.

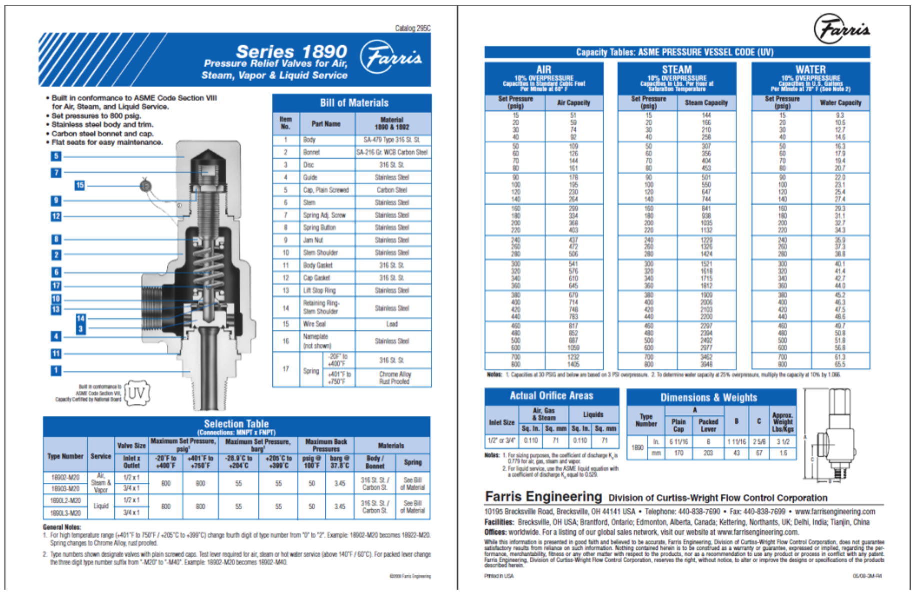
4.1.4 Experimental Results & Discussion

Table 4.1: Data representing the measure of data recall precision and repeatability of the experiment.

Document Source	**Document format**	**Number of Documents**	**Number of tables**	**Recall Precision**(%)	**Repeatability**(%)
Product catalog	.pdf	6	37	100	100
Rolls-Royce Documents	.pdf	8	10	100	100
Rolls-Royce Documents	.ppt	10	17	100	100
Rolls-Royce Documents	.doc	6	23	100	100
Rolls-Royce Documents	.docx	2	3	100	100
Engineering Product Catalog (Farnell)	.html	10	10	100	100
Engineering Product Catalog (RS Catalog)	.html	10	10	100	100

[2] `www.oracle.com`
[3] `www.mysql.com`
[4] `www.w3.org/Protocols/`
[5] `www.w3.org/XML/`

The experiment focused on the repeatability and the recall precision of the data tables from these various sources by the implementation. Since the test was not on the performance of the search functionality of the implementation as such, the Commander was made to use all the documents that it was given access to the files in the folder. As for the cases on which the test was applied were:



Catalog 295C

Series 1890
Pressure Relief Valves for Air, Steam, Vapor & Liquid Service

Farris

- Built in conformance to ASME Code Section VIII for Air, Steam, and Liquid Service.
- Set pressures to 800 psig.
- Stainless steel body and trim.
- Carbon steel bonnet and cap.
- Flat seats for easy maintenance.

5
7
15
9
12
8
2
6
17
10
13
14
3
4
11
1
Built in conformance to ASME Code Section VIII, Capacity Certified by National Board
UV

Bill of Materials

Item No.	Part Name		Material 1890 & 1892
1	Body		SA-479 Type 316 St. St.
2	Bonnet		SA-216 Gr. WCB Carbon Steel
3	Disc		316 St. St.
4	Guide		Stainless Steel
5	Cap, Plain Screwed		Carbon Steel
6	Stem		Stainless Steel
7	Spring Adj. Screw		Stainless Steel
8	Spring Button		Stainless Steel
9	Jam Nut		Stainless Steel
10	Stem Shoulder		Stainless Steel
11	Body Gasket		316 St. St.
12	Cap Gasket		316 St. St.
13	Lift Stop Ring		Stainless Steel
14	Retaining Ring-Stem Shoulder		Stainless Steel
15	Wire Seal		Lead
16	Nameplate (not shown)		Stainless Steel
17	Spring	-20°F to +400°F	316 St. St.
		+401°F to +750°F	Chrome Alloy Rust Proofed

Selection Table
(Connections: MNPT x FNPT)

Type Number	Service	Valve Size	Maximum Set Pressure, psig[1]		Maximum Set Pressure, barg[1]		Maximum Back Pressures		Materials	
		Inlet x Outlet	-20°F to +400°F	+401°F to +750°F	-28.9°C to +204°C	+205°C to +399°C	psig @ 100°F	barg @ 37.8°C	Body / Bonnet	Spring
18902-M20	Air, Steam & Vapor	1/2 x 1	800	800	55	55	50	3.45	316 St. St. / Carbon St.	See Bill of Material
18903-M20		3/4 x 1								
1890L2-M20	Liquid	1/2 x 1	800	800	55	55	50	3.45	316 St. St. / Carbon St.	See Bill of Material
1890L3-M20		3/4 x 1								

General Notes:
1. For high temperature range (+401°F to 750°F / +205°C to +399°C) change fourth digit of type number from "0" to "2". Example: 18902-M20 becomes 18922-M20. Spring changes to Chrome Alloy, rust proofed.
2. Type numbers shown designate valves with plain screwed caps. Test lever required for air, steam or hot water service (above 140°F / 60°C). For packed lever change the three digit type number suffix from "-M20" to "-M40". Example: 18902-M20 becomes 18902-M40.

©2008 Farris Engineering

Farris

Capacity Tables: ASME PRESSURE VESSEL CODE (UV)

AIR 10% OVERPRESSURE Capacities in Standard Cubic Feet Per Minute at 60° F		STEAM 10% OVERPRESSURE Capacities in Lbs. Per Hour at Saturation Temperature		WATER 10% OVERPRESSURE Capacities in U.S. Gallons Per Minute at 70° F (See Note 2)	
Set Pressure (psig)	Air Capacity	Set Pressure (psig)	Steam Capacity	Set Pressure (psig)	Water Capacity
15	51	15	144	15	9.3
20	59	20	166	20	10.6
30	74	30	210	30	12.7
40	92	40	258	40	14.6
50	109	50	307	50	16.3
60	126	60	356	60	17.9
70	144	70	404	70	19.4
80	161	80	453	80	20.7
90	178	90	501	90	22.0
100	195	100	550	100	23.1
120	230	120	647	120	25.4
140	264	140	744	140	27.4
160	299	160	841	160	29.3
180	334	180	938	180	31.1
200	368	200	1035	200	32.7
220	403	220	1132	220	34.3
240	437	240	1229	240	35.9
260	472	260	1326	260	37.3
280	506	280	1424	280	38.8
300	541	300	1521	300	40.1
320	576	320	1618	320	41.4
340	610	340	1715	340	42.7
360	645	360	1812	360	44.0
380	679	380	1909	380	45.2
400	714	400	2006	400	46.3
420	748	420	2103	420	47.5
440	783	440	2200	440	48.6
460	817	460	2297	460	49.7
480	852	480	2394	480	50.8
500	887	500	2492	500	51.8
600	1059	600	2977	600	56.8
700	1232	700	3462	700	61.3
800	1405	800	3948	800	65.5

Notes: 1. Capacities at 30 PSIG and below are based on 3 PSI overpressure. 2. To determine water capacity at 25% overpressure, multiply the capacity at 10% by 1.066.

Actual Orifice Areas

Inlet Size	Air, Gas & Steam		Liquids	
	Sq. In.	Sq. mm	Sq. In.	Sq. mm
1/2" or 3/4"	0.110	71	0.110	71

Notes: 1. For sizing purposes, the coefficient of discharge K_d is 0.779 for air, gas, steam and vapor.
2. For liquid service, use the ASME liquid equation with a coefficient of discharge K_d equal to 0.529.

Dimensions & Weights

Type Number		A		B	C	Approx. Weight Lbs/Kgs
		Plain Cap	Packed Lever			
1890	In.	6 11/16	8	1 11/16	2 5/8	3 1/2
	mm	170	203	43	67	1.6

Farris Engineering Division of Curtiss-Wright Flow Control Corporation

10195 Brecksville Road, Brecksville, OH 44141 USA • Telephone: 440-838-7690 • Fax: 440-838-7699 • www.farrisengineering.com

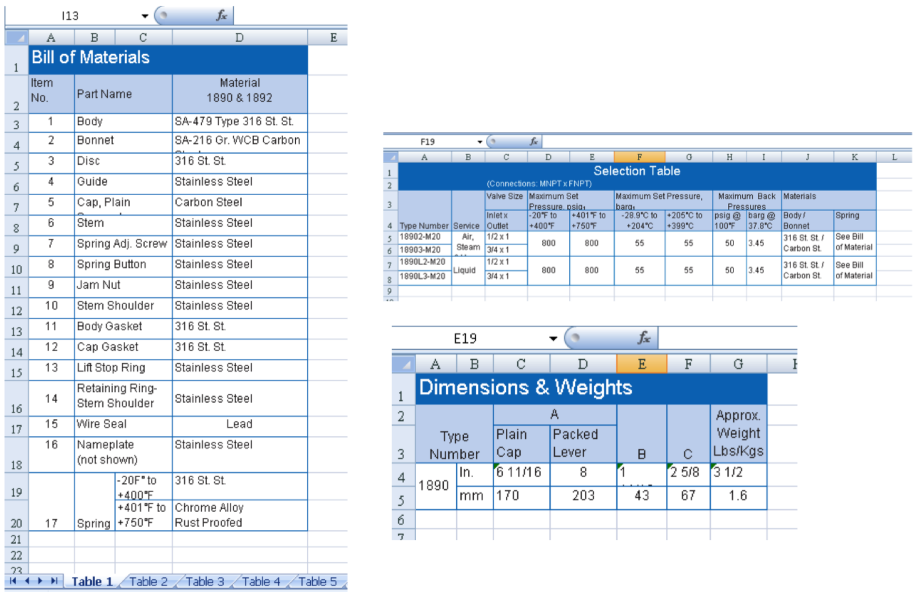
Facilities: Brecksville, OH USA; Brantford, Ontario; Edmonton, Alberta, Canada; Kettering, Northants, UK; Delhi, India; Tianjin, China

Offices: worldwide. For a listing of our global sales network, visit our website at www.farrisengineering.com.

While this information is presented in good faith and believed to be accurate, Farris Engineering, Division of Curtiss-Wright Flow Control Corporation, does not guarantee satisfactory results from reliance on such information. Nothing contained herein is to be construed as a warranty or guarantee, expressed or implied, regarding the performance, merchantability, fitness or any other matter with respect to the products, nor as a recommendation to use any product or process in conflict with any patent. Farris Engineering, Division of Curtiss-Wright Flow Control Corporation, reserves the right, without notice, to alter or improve the designs or specifications of the products described herein.

Printed in USA 05/08-3M-FM

(a) The Farris product catalog.



I13

Bill of Materials

Item No.	Part Name		Material 1890 & 1892
1	Body		SA-479 Type 316 St. St.
2	Bonnet		SA-216 Gr. WCB Carbon
3	Disc		316 St. St
4	Guide		Stainless Steel
5	Cap, Plain		Carbon Steel
6	Stem		Stainless Steel
7	Spring Adj. Screw		Stainless Steel
8	Spring Button		Stainless Steel
9	Jam Nut		Stainless Steel
10	Stem Shoulder		Stainless Steel
11	Body Gasket		316 St. St.
12	Cap Gasket		316 St. St.
13	Lift Stop Ring		Stainless Steel
14	Retaining Ring-Stem Shoulder		Stainless Steel
15	Wire Seal		Lead
16	Nameplate (not shown)		Stainless Steel
		-20F° to +400°F	316 St. St.
17	Spring	+401°F to +750°F	Chrome Alloy Rust Proofed

F19

Selection Table
(Connections: MNPT x FNPT)

Type Number	Service	Valve Size	Maximum Set Pressure, psig₁		Maximum Set Pressure, barg₁		Maximum Back Pressures		Materials	
		Inlet x Outlet	-20°F to +400°F	+401°F to +750°F	-28.9°C to +204°C	+205°C to +399°C	psig @ 100°F	barg @ 37.8°C	Body / Bonnet	Spring
18902-M20	Air, Steam	1/2 x 1	800	800	55	55	50	3.45	316 St. St. / Carbon St.	See Bill of Material
18903-M20		3/4 x 1								
1890L2-M20	Liquid	1/2 x 1	800	800	55	55	50	3.45	316 St. St. / Carbon St.	See Bill of Material
1890L3-M20		3/4 x 1								

E19

Dimensions & Weights

Type Number		A		B	C	Approx. Weight Lbs/Kgs
		Plain Cap	Packed Lever			
1890	In.	6 11/16	8	1	2 5/8	3 1/2
	mm	170	203	43	67	1.6

(b) The data extracted from Farris catalog as an Excel spreadsheet.

Figure 4.2: An example of tabulated data extraction from a catalog in PDF format.

1. *Product catalogs (PDF):* MTC Engineering[6], GE Industrial[7], Farris[8], and Price[9] were different candidates whose catalogues in PDF were used as input to the EP3-Static implementation. Once the EP3-Static was executed, all the tabulated data was successfully extracted, since the recall precision, verified by inspection of the PDF documents, and repeatability was found to be 100% (Table 4.1). Figure 4.2(b) is an example of data extracted from a PDF catalog as in Figure 4.2(a).

2. *Engineering product catalog (HTML):* This case was different from the others in a way that the search function inherently available in the websites to downsize the number of web pages that would be analysed and parsed for data. The websites used were RS Catalog[10] and Farnell Catalog[11]. The data extracted were verified by inspection of the webpages that were parsed and the results were found to be accurate. Likewise, the experiment for all the cases were repeated and the output data was verified with the source document by inspection. Table 4.1 confirms that the recall precision and the repeatability were 100%.

3. *Rolls-Royce documents of Cost Engineer (PDF, Microsoft Powerpoint, Microsoft Word):* Similarly the EP3 static was executed on the documents created by a Rolls-Royce employee related to engine costing. The results were again verified by inspection of the documents, which in this case were also in other formats of email message (.msg) and Microsoft Office[12] products like (Word, Powerpoint and Excel) along with PDF. The results in Table 4.1 show that the recall precision and repeatability was measured to be 100%.

4.2 Data Reuse Strategy

The results on the ability to extract tabulated data from heterogenous sources being a 100%, proves that the data extraction is possible. This capability to extract data used in this experiment partially completes task of EP3 as envisioned in Figure 3.8.

Examining the resultant tables extracted in Figure 4.2(a), even though the tabulated data have been extracted, they have been structured specifically for a human to read. Examining each of the tables extracted, the search for the required data was limited to the key word search of the sources. To make a computer read each data table and to be able to reuse a data point from any of these tables would then require a substantial amount of effort. Therefore, these data is still unusable considering there would be a lot of documents with numerous tables.

[6] `www.mtceng.com`
[7] `www.geindustrial.com`
[8] `http://farris.cwfc.com`
[9] `www.price-hvac.com`
[10] `uk.rs-online.com/web/`
[11] `uk.farnell.com`
[12] `office.microsoft.com`

Consequently, the next step in path to completion of the task of EP3, as described in Section 3.4, is to have the ability to 'recognise and retrieve the required data'. A common factor among all the data tables extracted is a lack of a standard data model which is a prime requirement in making data machine readable. This calls for the need for a framework to ensure machine readability of data.

For each source of data, the data would therefore then need to be manually mapped to a suitable data model. Once this has been done atleast once for the data in all the existing documents, they could all be stored in a database for reuse. This can be a tedious and expensive task for the organisation, considering the existence of numerous documents that contain data. Moreover, there are also issues that would be related to deduplication, trust, accuracy and maintenance of the datasets. Also, applying a data model can be a difficult task especially for a person who doesn't understand the context of a dataset.

A possible strategy that one could take is to crowd source the data modelling and maintenance tasks, by using the provenance information of the document from which a dataset was extracted. It was observed that every Rolls-Royce document had the user id information of the employee that created a document. This provenance information of a document can be specifically useful to narrow down the document search used in the company as noted by Baader et al. (2006). There is a likelihood that the data created by a person holding a position 'A' in the company would be reused by another person who would be doing a similar role or a role related to 'A'. The EP3 therefore can use this provenance information to get the people who are most likely to understand the data to model it as and when it needs to be reused.

4.3 Scope of Research

Even after all the data has been converted into a data model, the question of how should one make this data available to a reuser, still exists. This question could be viewed in a simplified way that if one was to create a system to host all the newly created data, what would that system be?

A direction that the research can focus on is extraction of potential data from paragraphs written in natural language which would require intense use of Natural Language Processing (NLP) techniques. Another direction would be to use the other information in a document and try and predict the structure of the data stored tabular format, but these will have to be considered as out of scope for this research work.

With a lot of focus on the topic named 'Big Data' recently, there are a number of tools that are being created to analyse and reuse data. The author identifies the challenges faced in collating information for costing as a different problem. Costing with reuse of data as understood from the analysis in Chapter 3, requires a system where a person can get answers to a variety of questions specific to the customer for whom the engine is being designed. This requires collation of a wide variety of data from multiple domains and still be able to use it in a useful manner. For example, the data extracted as in Figure 4.2(a), is an example where the 'ASME pressure vessel data' is useless unless the reuser is aware of the process by which it can be used to Figure out the appropriate dimensions of the required product. A lot of this can be solved by having appropriate user interfaces (UI) specific to the task the data reuser is attempting to carry out, though the UI would still require the data to be available in a form that is reusable.

This situation can be termed as a 'Broad Data' challenge rather than the 'Big Data' challenge, since the variety of data required is not restricted. The term 'Broad Data' was coined by Hendler (2012). This research will, therefore, primarily focus on developing the framework necessary to process 'broad data' questions.

4.4 The Research Question

Having narrowed down the scope of the this research, the research question that this research will aim to answer is:

'What is an appropriate framework for data reuse in an organisation to support a broad data reuse activity'.

4.5 Conclusion

Having understood the various challenges in reusing data for costing, having the ability to reuse the data itself was identified to be the primary challenge. The heterogeneity in the storage of data was a key issue. The experiment in Section 4.1 showed that the extraction of tabluated data was possible with 100% recall precision of tabulated data with 100% repeatability. The subsequent challenge was to be able to recall a specific data point from a dataset. The need to apply a data model was realised. Furthermore, the scope of this research was narrowed down for the need to have a framework that can support reuse of 'broad data'.

RDF (Resource Description Framework) is a concept that is being researched for its ability to create machine readable data and therefore is a potential choice to model the data. All the data when in RDF can be treated as a data graph and there are numerous tools that are being developed to use such data. Further with the use of unique identifiers data in RDF form Linked Data which can be hosted on the World Wide Web. Linked Data is a research topic that is currently active and new standards have also been created like SPARQL for that can be potentially used for intelligently querying data.

Further with the creation of a giant graph of Linked data, it is a plausible solution to the problem of heterogeneity and sparseness of the data. The concept, creation methods and applicability of Linked Data is further discussed and presented in the next chapter.

Chapter 5

Linked Data : A Discussion

"Linked (Open) Data technologies offer a new way of data integration for the enterprise. Smooth interoperability between internal data sets can reduce costs as well as the enrichment of these data sets by external data can support new market intelligence paradigms for a better decision making."

– Martin Kaltenboeck , *Semantic Web Company*

This chapter introduces the concept of linked data. Linked data is being introduced and positioned to solve the issue of heterogeneity in the data models. The principles of linked data are then described and the method to convert data to RDF model specifically for the purpose of data mining are also then prescribed. An example case study is then presented to understand how linked data can be employed to support retrieving the data for data mining.

5.1 The Rationale for Linked Data

The ability to reuse data is the top priority for data extraction procedures, and a key factor in gauging its reusability, is the extent to which it is well structured (Heath and Bizer, 2011). Once data has been extracted with structure, the next step is to identify what data it is, to be able to recognise and collate the relevant data. Collation of the data is a difficult step considering the sparse nature and since the data would then be subjected to data mining procedures, being well structured becomes important. Linked data represents a set of procedures and guidelines in publishing data, which enables sophisticated processing and connect distributed data (Heath and Bizer, 2011).

Most of the documents such as Microsoft Word document (.doc, .docx), Microsoft PowerPoint (.ppt), email messages (.msg), and Web sites (.html) do have some degree of structure, but the programs that create such formats are oriented to structure textual documents rather than data. As observed by Heath and Bizer (2011), the data is intermingled into the surrounding text making it hard for software applications to extract the data. Microsoft Excel, Oracle, MySQL, Microsoft Access, and so on are inherently designed to handle data and ensure data is structured by making user conform to a structured input.

A generic approach to make structured data available is through an application programming interface (API). APIs can provide simple access to query structured data. They also can help bridge the incompatibilities that occur, especially between two programs that are build upon different programming languages. Some software vendors provide API to control what and how a third party can use information from their tool. Heath and Bizer (2011) observes, "the advent of Web APIs (structured data made available through web) has led to an explosion in small, specialized, applications (or mashups) that combine data from several sources, each of which is accessed through an API specific to the data provider." However, with the existence of such specialised APIs, one needs to know the methods for retrieving data from each API. This has created a situation where significant effort is required to integrate each data set into an application. This called for the need to have a consistency in the way in which APIs are used.

The linked data principles create a landscape where guidelines and standards are followed while publishing the data, thus creating a web of data rather than data islands.

5.2 Linked Data Principles

"Technically, Linked Data refers to data published on the Web in such a way that it is machine-readable, its meaning is explicitly defined, it is linked-to other external data sets, and can in turn be linked-to from external data sets" (Bizer et al., 2009). Linked data builds upon the standards of the Web technologies like HTTP and URIs, but extends them to share information in a way that data from different sources can be connected and queried (Kirchberg et al., 2011). Heath and Bizer (2011) observes that the term 'linked data' refers to a set of best practices for publishing and interlinking structured data and Tim Berners-Lee introduced these as 'linked data principles' (Berners-Lee, 2009). These principles were designed for linked data creation on the World Wide Web (WWW) (Jacobs and Walsh, 2004) and they are:

1. Use URIs as names for things.
2. Use HTTP URIs, so that people can look up those names.
3. When someone looks up a URI, provide useful information, using the standards (RDF, SPARQL).
4. Include links to other URIs, so that they can discover more things.

Heath and Bizer (2011) explains that the basic idea of linked data is to apply general architecture of WWW to the task of sharing structured data on the global scale and hence, understanding of the WWW architecture is necessary.

5.2.1 The World Wide Web (WWW)

"The World Wide Web (WWW, or simply Web) is an information space in which the items of interest, referred to as resources, are identified by global identifiers called Uniform Resource Identifiers (URI)", as noted by Jacobs and Walsh (2004). People or software acting on this information space are called as 'Web agents'. A user agent acts on behalf of a user and software agents include servers, proxies, spiders, browsers, and multimedia players. To understand the architecture of WWW, Jacobs and Walsh (2004) uses the scenario in the box below (page 41).

Scenario:
While planning a trip to Kerala, Ammu reads Ottapalam weather information: `http://weather.example.com/ottapalam` in a glossy travel magazine. Ammu has enough experience with the Web to recognize that `http://weather.example.com/ottapalam` is a URI and that she is likely to be able to retrieve associated information with her Web browser. When Ammu enters the URI into her browser:

1. The browser recognizes that what Ammu typed is a URI.
2. The browser performs an information retrieval action in accordance with its configured behavior for resources identified via the 'http' URI scheme.
3. The authority responsible for `weather.example.com` provides information in a response to the retrieval request.
4. The browser interprets the response, identified as XHTML by the server, and performs additional retrieval actions for inline graphics and other content as necessary.
5. The browser displays the retrieved information, which includes hypertext links to other information. Ammu can follow these hypertext links to retrieve additional information.

From the scenario above, Jacobs and Walsh (2004) illustrates the three architectural basis of the Web, namely:

1. *Identification:*

   To be able to share information within a community, they have to agree on a set of terms and meanings. Since the inception of the web, it makes use of single global identification system: the URI. Resources are identified by URIs and in the story above the resource was periodically updated with the weather information of Ottapalam and the URI was `http://weather.example.com/ottapalam`.

2. *Interaction:*

   Standardised protocols define the interaction between the web agents to enable exchange of messages. The Web's protocols for exchange of messages includes HTTP, FTP, SOAP, NNTP, and SMTP. A message may also include the metadata about the resource, the message data and the message itself (Jacobs and Walsh, 2004). Ammu, in the story above, entered the URI into a retrieval dialog box or clicked on a hypertext link in her browser, a web agent, and the browser performs a retrieval action for the resource identified by the URI.

3. *Formats:*

   Web architecture does not constrain which data formats the content providers can use and a data format specification (XHTML, RDF/XML, CSS, PNG, etc.) embodies an agreement on the correct interpretation of the representation data. For a data format to be useful, the content provider and the user has to be in agreement with the format and web browser usually takes care of this for the user by being able to parse most of the data formats. In the story above, 'the browser is configured and programmed to interpret the receipt of an "application/xhtml+xml" typed representation as an instruction to render the content of that representation according to the XHTML rendering model, including any subsidiary interactions (such as requests for external style sheets or in-line images) called for by the representation' (Jacobs and Walsh, 2004).

Heath and Bizer (2011) explains that the development and the use of standards enables the Web to transcend different technical architectures, with hyperlinks enabling the users to navigate between different servers. They also enable crawling of the web by the search engines. Hyperlinks as noted by Heath and Bizer (2011) are therefore crucial in connecting content from different servers into a single global information space.

5.2.2 Understanding the Linked Data Principles

An exhaustive search on the literature shows that the previous work on linked data builds directly on the Web architecture and applies this architecture to the task of sharing data on global scale. Thus, both informational and non-informational resources are identified by URIs (Berners-Lee, 2005a). A set of URIs that have been grouped together forms a vocabulary. For a vocabulary where the meaning of their URIs and its relationships with the other URIs within that vocabulary has been agreed upon by its users, is called an 'Ontology'. Guarino (1998) notes it's a logical theory accounting for the intended meaning of a formal vocabulary. From an artificial intelligence domain, ontology can be defined as a specification of conceptualisation (Grüber, 1993). In the world of linked data, an ontology can be described in RDF Schema (RDFs) (Manola and Miller, 2004) and Web Ontology Language (OWL) (Brickley and Guha, 2004). These formats can also be used to define relationships among the data that are published in RDF.

The relationship between data are represented in triples, where each triple is composed of a subject, predicate and an object. W3C recommends storage of triples in RDF format and thus many triples together forming an RDF dataset. Once the data is in RDF, SPARQL (SPARQL Protocol and RDF Query Language) described by a set of specifications from the W3C allows users to query the RDF dataset through SQL-like queries (Prud'hommeaux and Seaborne, 2008).

5.3 Linked Data Design Considerations

Since the primary purpose of transforming a dataset into linked data is to enable easier sharing, one must follow certain practices. These practices are designed based on where the data would be hosted and how it would be accessed. The naming of things with URI and describing them with RDF are the two activities that needs to be thought about. These decisions are important for the data to fit neatly into the Web and to be a useful linked dataset.

5.3.1 Using URIs as names

As the expansion of the name URI (uniform resource identifier) suggests, it is used to name a resource which can then be universally identified by referring to the URI. These resources can be real world entities like a person, a building, a dog, or more abstract ideas like a scientific concept (Heath and Bizer, 2011). This forms the first principle of linked data as laid out by Sir Tim Berners-Lee.

Following the second principle of linked data, a URI should be created using http URI scheme. This gives the universal uniqueness of a URI for a resource from the fact that there is currently only one WWW and that a http URI on the web always points to the same resource. Creating a http URI simply amounts to choosing part of a http://namespace that the data publisher controls. To ensure that the data publisher is able to host information about a URI, he/she must be owning the domain name of the http://namespace that was used to mint http://URI (The reader needs to note that the term 'mint' will be used to refer to the process of creating a URI). This allows these names (URIs) to be looked up by using a web client, such as a web browser, that can communicate using the HTTP protocol. This also helps to conform to the third principle of linked data where the data creator is able to provide more details about a URI when one looks up.

Due to the way in which domain ownership works in the WWW, where one who owns a domain controls it, one cannot reliably use a http URI based on a domain they do not own. This is because the one who owns the URI can change them and thus, minting a URI that oneself doesn't control is considered 'uncool' (Heath and Bizer, 2011).

5.3.2 Cool URIs

Since, a primary benefit of creating linked data is addition of value through creation of incoming and outgoing links, it is important to inspire confidence in third parties to consider linking to a data set. This confidence comes from the effort that is expended on minting stable and persistent URIs. As Heath and Bizer (2011) noted, "the specifics of the technical hosting environment may introduce some constraints on the precise syntax of these URIs; however, following a few simple rules can help achieve this".

A cool URI is one that is not likely to break if the site was moved to a different machine or reimplemented using different scripting language or framework (Heath and Bizer, 2011). Sir Tim Berners-Lee describes a cool URI as one that doesn't change (Berners-Lee, 1998a). The first step to minting a cool URI is to not use any namespace that you don't control. A common example is the use Internet Movie Database (IMDB) to refer to movies, directors, actors, etc. Each is described in a HTML document in the address like `http://www.imdb.com/title/tt1667889/` which is a URI of the film 'Ice Age: Continental Drift'. Heath and Bizer (2011) explains that it is not unreasonable to augment this URI with a fragment identifier that identifies the film itself such as:

`http://www.imdb.com/title/tt1667889/#film`

However, this approach is considered problematic as no one other than the owner of the domain, imdb.com, would be able to dereference this URI. As an alternative, the recommendation was to mint a URI in the domain that one own's thyself and state the equivalence between these and the corresponding URIs in other datasets.

Implementations can also vary on how one serves information about the URIs they have minted, such as, the URI could be a .php extension for a site mostly implemented in PHP (`http://www.mysite.com/title/tt1667889.php?id=ice-age-4"&format=rdf` , Note: This link is just an example and the author does not intend to say that mysite.com is implemented in PHP). Heath and Bizer (2011) strongly recommends to abstract away from using such implementation details in the URI and also to avoid including port numbers (`http://www.mysite.com:8080/title/tt1667889.rdf`); but recommends use of natural keys within the URI. An example of a cool URI for the 'ice age 4' movie in my own domain that serves an RDF would be:

1. URI for a real world object:

   `http://movie.mysite.com/ice-age-4`

2. URI for related information that describes the real world object and has an HTML page:

   `http://movie.mysite.com/ice-age-4.html`

3. URI for real related information that describes the real world object and has an RDF representation:

   `http://movie.mysite.com/ice-age-4.rdf`

5.3.3 Using RDF to describe things

Reminding the third principle of linked data, it states "When someone looks up a URI, provide useful information...". The previous sections have communicated on how to mint URIs to identify resources and the best practices to make them 'cool URIs'. The next step is to formulate how to describe the data that is associated with a resource described by a URI.

The W3C standard to describe URIs is by using the Resource Description Format (RDF). RDF is a data model in which the basic unit of information is a triple. Each triple consists of a subject, a predicate, and an object; DuCharme (2011) noted that, one can equate them to a resource identifier, an attribute or property name, and an attribute or property value, respectively. To understand the triple better, it is important to see how we can represent the following three facts:

1. The movie 'Ice Age 4' was directed by Steve Martino.
2. The movie 'Ice Age 4' has genre animation.
3. Steve Martino is American.

From table 5.1, one can see how to represent a fact as a triple. These RDF triples can then be stored in various formats like RDF/XML, N-triple, Turtle, etc. Now that we have data expressed as RDF triples, it is important to note the various types of triples that should be included into the description as best practice; Heath and Bizer (2011) enumerates:

1. Triples that describe resources with literals.
2. Triples that describe the resource by linking to other resources.
3. Triples that describe by linking from other resources (e.g., triples stating the resource creator).

Table 5.1: The table shows how to represent the facts in subject, predicate, and object format, i.e., as a triple (using respective URIs).

Fact	Subject	Predicate	Object
The movie 'Ice Age 4' was directed by Steve Martino.	`http://movie.mysite.com/ice-age-4`	`http://title.mysite.com/director`	`http://people.mysite.com/Steve-Martino`
The genre of the movie 'Ice Age 4' is animation.	`http://movie.mysite.com/ice-age-4`	`http://about-movie.mysite.com/genre`	`http://about-movie.mysite.com/genre/animation`
Steve Martino is American.	`http://people.mysite.com/Steve-Martino`	`http://country.mysite.com`	`http://country.mysite.com/usa`

4. Triples describing related resources (e.g., affiliation of the creator).
5. Triple describing the description itself (e.g., provenance, licensing terms).
6. Triples about the broader dataset of which this description is part of.

5.4 Consuming Linked Data

All the previous discussion were on how to create linked data and the guidelines were also laid out so that the data published as linked data can then be consumed for various purposes. SPARQL is the standard described by the W3C to consume linked data (Prud'hommeaux and Seaborne, 2008).

5.4.1 SPARQL

SPARQL (pronounced as 'sparkl') is a recursive acronym for 'SPARQL protocol and RDF Query Language'. The protocol part of the SPARQL refers to the rules for how a client program and a SPARQL processing server exchange SPARQL queries and results. More on the protocol can be accessed from the W3C recommendations by Grant Clark et al. (2008). Typically a SPARQL query says, "I want these pieces of information from the subset of the data that meets these conditions" (DuCharme, 2011). The SPARQL query language for RDF covers the syntax that is used to construct the query itself and more on the syntax can be read from (DuCharme, 2011).

5.4.2 Storing RDF in Databases

The SPARQL queries can get bigger, complex and the number of triples can grow so large that keeping them as Turtle or RDF/XML in a file wouldn't be the best option. Use of a database is the classical solution especially since the database management systems would be more efficient in indexing the data and to make a decision which data to load into the memory for a specific query. Considering the structure of the RDF data model a database model has been designed for RDF and is known as a 'triplestore'. Triplestores are available on both commercial and open source licenses and performance benchmarking of major triplestores can be found in (Rohloff et al., 2007).

5.4.3 Linked data publishing patterns

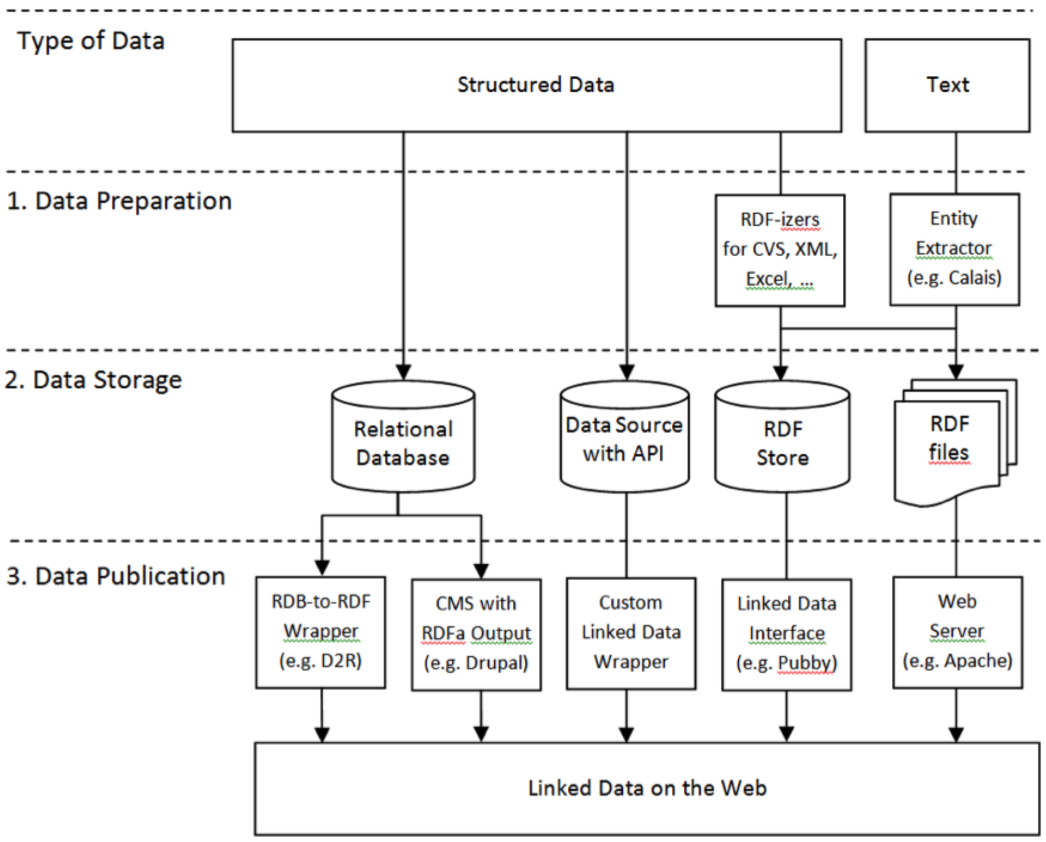

Type of Data
Structured Data
Text
1. Data Preparation
RDF-izers for CVS, XML, Excel, ...
Entity Extractor (e.g. Calais)
2. Data Storage
Relational Database
Data Source with API
RDF Store
RDF files
3. Data Publication
RDB-to-RDF Wrapper (e.g. D2R)
CMS with RDFa Output (e.g. Drupal)
Custom Linked Data Wrapper
Linked Data Interface (e.g. Pubby)
Web Server (e.g. Apache)
Linked Data on the Web

Figure 5.1: Linked data publishing options and workflows as described by Heath and Bizer (2011).

SPARQL is not limited just to querying of RDF data. Various utilities are available that treat XML, spreadsheets, etc. Likewise data stored in other types databases can also be transformed by the use of a small APIs that can act as SPARQL endpoints. Heath and Bizer (2011) points out that developers should not totally abandon "existing data management system and business applications", but add an "extra technical layer of glue to connect these into the Web of Data".

Figure 5.1 shows various linked data publishing methods based on different data and storage types. Storage of data as a RDF file is considered the most straightforward method of all, where data creators make static RDF files which can be queried by SPARQL. However for storing data which needs constant updating, this is not the best method; and so to be able to use database management tools, triplestores are used to store the RDF data. Triplestores provide a SPARQL endpoint through which the data can be queried.

A common way of publishing data is to use relational databases and these can be transformed into linked data by having a mapping between the database schema and the vocabularies. The tools that are usually used to provide this mapping are D2R server (Bizer and Cyganiak, 2006), OpenLink Virtuoso (Erling and Mikhailov, 2009), and Triplify (Auer et al., 2009). Some applications prefer to expose their data through APIs and one can write a wrapper that converts API data into RDF.

5.5 Linked Data for Data Mining

Taking the example of predicting the cost of a jet engine, for the purpose of data mining, the basic requirement from the linked data is to be able to bring together the data that describe or relate to various jet engines. This task of bringing data together is especially challenging when the data is stored away in heterogenous formats. Primarily linked data helps to standardise the data format and the data being in RDF model is phenomenally useful considering they can then be directly queried upon, using SPARQL.

5.5.1 Linked Data Creation Procedure: For Data Mining Purposes

Table 5.2: An example made-up dataset of jet engines and data related to it.

Jet Engines	**Manufacturer**	**Fan diameter** (m)	**SFC** (Kg/N/hr)	**TET** (Kelvin)	**Thrust** (lbf)	**Cost** (British Pound)
Ea1	Rolls-Royce	8	0.014	1774	78000	0.9
Ea5	GE	6	0.009	1689	54000	1.01
Ea15	Snecma	3	0.01	1976	24000	1.5
Ea101	GE	9	0.02	1873	89000	1.7

Since the focus of this section is to consume data specifically for the purpose of data mining, a set of simplified guidelines are essential to speed up the process of creating linked data. These guidelines are assuming that one has extracted all the data from heterogenous forms of storage and are now available as structured (tabulated) datasets.

To understand the guidelines, it is essential that one understands how one comprehends a dataset when its presented in a tabular form.

```
@prefix mydata: <http://example.com/> .

mydata:Ea1  a  mydata:gasturbine ;
  mydata:manufacturer    "Rolls-Royce" ;
  mydata:fan-diameter    "8"^^mydata:meter ;
  mydata:sfc     "0.014"^^<http://example.com/K/N/hr> ;
  mydata:tet     "1774"^^mydata:Kelvin ;
  mydata:thrust     "78000"^^mydata:lbf ;
  mydata:cost     "0.9"^^<http://example.com/BritishPounds> .

mydata:Ea5 a mydata:gasturbine ;
  mydata:manufacturer     "GE" ;
  mydata:fan-diameter     "6"^^mydata:meter ;
  mydata:sfc     "0.009"^^<http://example.com/K/N/hr> ;
  mydata:tet     "1689"^^mydata:Kelvin ;
  mydata:thrust     "54000"^^mydata:lbf ;
  mydata:cost     "1.01"^^<http://example.com/BritishPounds> .

mydata:Ea15 a mydata:gasturbine ;
  mydata:manufacturer     "Snecma" ;
  mydata:fan-diameter     "3"^^mydata:meter ;
  mydata:sfc     "0.01"^^<http://example.com/K/N/hr> ;
  mydata:tet     "1976"^^mydata:Kelvin ;
  mydata:thrust     "24000"^^mydata:lbf ;
  mydata:cost     "1.5"^^<http://example.com/BritishPounds> .

mydata:Ea101 a mydata:gasturbine ;
  mydata:manufacturer     "GE" ;
  mydata:fan-diameter     "9"^^mydata:meter ;
  mydata:sfc     "0.02"^^<http://example.com/K/N/hr> ;
  mydata:tet     "1873"^^mydata:Kelvin ;
  mydata:thrust     "89000"^^mydata:lbf ;
  mydata:cost     "1.7"^^<http://example.com/BritishPounds> .
```

Listing 5.1: Data from Table 5.2 in RDF data model, where URIs were minted on the domain `http://example.com/`.

Table 5.2 as an example dataset of jet engines and its associated information. To start the creation of an RDF dataset, the first question one must ask is what is this dataset about, which in this example is jet engine. The column 1 which has the names of all the engines can be called the stem or pivot, and all the other columns can now be viewed as the branches that are attached to the stem at the appropriate root (appropriate engine) on the stem. Once this model has been identified for every dataset, one can start describing them as RDF and the guidelines are:

1. The subject field of every triple should be the respective row from the pivot.

2. The column heading forms the predicate field of a triple.
3. The units that define the object field should be specified as a datatype.
4. Each and every subject (the rows from the pivot) should be backlinked.
5. Backlinking means to create a triple denoting what the subject is, for e.g., <Ea1> a <jet-engine>.

One can see the data from Table 5.2 as RDF data in listing 5.1, while following the linked data creation steps described. URIs were minted for the subjects and predicates under the domain http://example.com/.

5.5.2 Consuming Linked Data for Data Mining

Once all the data has been converted in RDF data following the previously described steps in Section 5.5.1, one can easily group all the triples that has the predicate ‘a’. These triples can be used for one to get an idea of the data that would available in the dataset. This can be especially useful in a situation when you are building a dataset to be data mined and you realise that one of the rows needs to be represented by other parameters that associate to it; one can just use that row as the new pivot and expand the dataset.

5.6 Example Case Study: Creating and Consuming Linked Data for Data Mining

This section will look at the example use case study of the consumption of linked data which was created using the guidelines listed above. The scenario is of an automotive company with various Operational Buisness Units (OBU), such as sales, performance, pollution, supply chain, etc. Each OBU has a specialized function and generate dataset that relates to their operations.

Once all the datasets from different groups have been converted to RDF, with numerous car models linked to the concept ‘Complete Car Model’ Figure 5.2 illustrates the graphical view of the linked dataset that would prevail once all the datasets have been converted into RDF, exposes the minimum information (i.e., concepts) that is required to discover the data in the dataset.

```
PREFIX coo:<http://purl.org/coo/ns#>
Select Distinct ?s1 ?p2 ?o2 {
<http://mydata.com/C+Class> ?p1 ?o1 .
?s1 ?p1 ?o1 . ## Finding other cars linked to CompleteCarModel.
```

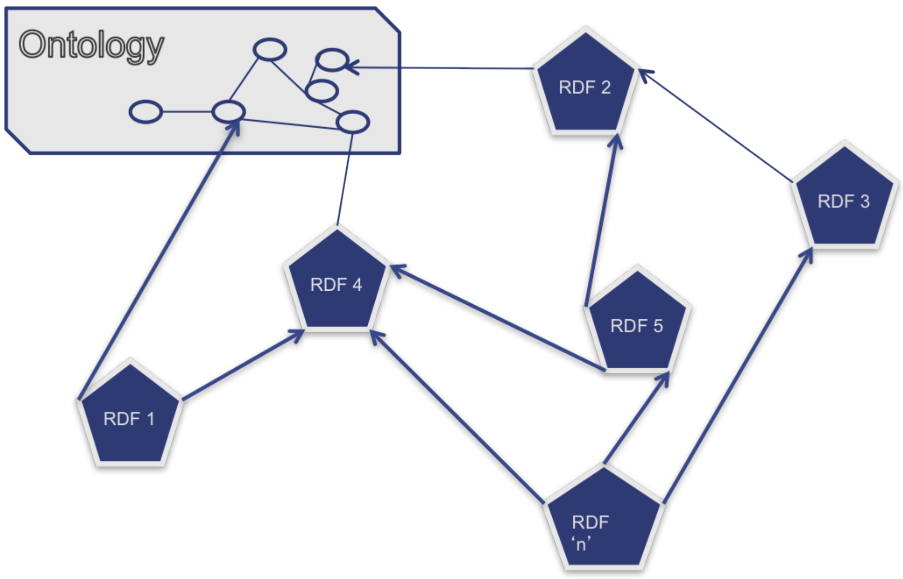


Figure 5.2: Illustration of the state of links after creation of linked datasets.

```
?s1 ?p2 ?o2 . ## Finding data related to all the car names linked to
    CompleteCarModel.
?s2 ?p2 ?o3. ## Finding other data that represent the variables ?p2.
FILTER (?o1 IN (coo:CompleteCarModel))
}
```

Listing 5.2: SPARQL query structure to retrieve the data related to 'C Class' and similar automobiles. Note comments in each line start with '##'

To query this data, a few assumptions are made:

1. There are no spelling mistakes.
2. All the employees are using the same vocabulary.
3. Finally, all the data are stored in the same triplestore, just to avoid using federated querying in this example.

Listing 5.2 shows the SPARQL query constructed to find all the cars like 'C Class' which are defined as 'coo:CompleteCarModel' and all the data that describes the cars. This example case study shows, how one can create linked data and query them in the simplistic form. In an enterprise, the situation would be different with use of numerous triplestores (maybe, even one for each OBU) keeping in mind data privacy and protection. SPARQL 1.1 standard from W3C does include the ability to do federated queries (Gearon et al., 2012). Thus, irrespective of having numerous triplestores, one can always query the data with the same vigor, as long as one knows the location of the SPARQL endpoints of those triplestores.

5.7 Summary

The primary hindrance in collating data is, the use of heterogenous data storage mediums. One can attribute the use of numerous data storage model like Comma Separated Value (CSV), to the ease in creation of a data store for the benefit of being quick in completing the process of storage, its ability to be used independent of the Operating System (OS), and programming environment, and to the ease of sharing the data with another person by way of internet based mediums like email. While, the use of relational data models along with relational databases helps to issue required queries, ensure consistency in answers, and the ability to use database management tools. The internet infrastructure made sharing of data and remote querying of databases possible and the WWW made the discovery and delivery of the data easier. The WWW is a universally accepted standard for identifying and redirecting one to the location of a resource on the internet. W3C maintains these standards and along with, issues recommendations for the format in which the web documents should be created, like html, xhtml, and so on.

One can then contrast the extensively used tabulated data model to the W3C standard of RDF which is a part of linked data standards. RDF explicitly describes data by having the subject, predicate and object fields to describe a data element. Moreover with the use of principles suggested by Sir Tim Berners-Lee for the creation of linked data, it enables computers and humans to recognise data at the same time. With the use of SPARQL, another standard maintained by the W3C, enables one to make complex queries on the linked data. Solving the problem of heterogeneity in data storage by linked data, therefore, is an important step ahead in making collation of required data possible.

A linked data creation procedure was then explored for the specific purpose of data mining in Section 5.5. The example case study shows that the use of linked data standards of SPARQL and RDF creates an environment where it is easier to collate the data into a dataset can be useful to support data mining. The example has assumed that user knew where every dataset that was required were, and had made that possible by placing all the data that were converted into RDF in the same triplestore. It is important to note that federated queries is possible with the formalization of SPARQL 1.1 (Gearon et al., 2012) and so the issue that will be faced in a corporate or even open source scenario is that one would find it difficult to locate the required datasets in the vast expanse of the Web. The immediate solution is to document the location of the SPARQL endpoint URI and RDF location, and provide a service like Kasabi[1]. This motivates one to visit and explore the idea of semantic web for locating the data.

[1]`http://kasabi.com`

Chapter 6

Semantic Web

"A lot of people have been working on making the software smarter. In the semantic web, we're going to start making the data smarter."

– Nova Spivack, *Radar Network*

This chapter introduces the concept of semantic web. Linked data technologies has helped to overcome the problem of heterogeneous data storage medium, but the semantic web is being reviewed to solve the issue of sparse nature of data. To understand the challenges the semantic web faces, first it is important to realise the vision of semantic web.

6.1 Semantic Web: The Concept and Rationale

The use of RDF data model to describe data is one of the fundamental causes for the success in employment of linked data concept to bring together data from heterogenous sources, as discussed in the previous chapter. The issue of heterogeneity in storage mediums can, therefore, be considered solved with the use of linked data.

This research work has now been left with finding the solution to the problem that arises from the sparse nature in the location of datasets. Compared to how the WWW solved the issue of locating documents from the sparsely located computers connected by the internet, Hall and Shadbolt (2009) calls the web of linked data as the Semantic Web. This chapter will explore the concept of semantic web for a solution.

"The Semantic Web is an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation" states Berners-Lee et al. (2001). Though one can easily conclude that the afore mentioned ability of the semantic web can already be achieved by employing linked data principles, it is important to comprehend that for any successful SPARQL query all the data that is necessary needs to be available for its execution. In the scenario of a corporate enviornment with many employees, it is possible to have created numerous linked datasets and then being unable to reuse them due to the inability to track the location of all the datasets that are being created. Therefore, the rationale for visiting the concept of 'semantic web' is to be able to identify and locate the data.

6.2 The approach to Semantic Web

Having established that the rationale for having semantic web is to be able to identify and locate the data, semantic web can be equated to a large database of metadata on data. Though the semantic web would differ from just a database of metadata in the way one is able to use this metadata. URI dereferencing is one such activity which is considered as a key enabler of the semantic web. This is because, to acquire more data about a URI one should be able to access that URI for further information about itself.

When a URI is dereferenced, as per the linked data principles (see Section 5.2) a dereferencing request needs to be replied with useful information about the URI, using standards of RDF and SPARQL. A simple approach that everyone could take, is to make the SPARQL endpoints available through '`/sparql`' of any URI, i.e., `http://URI/sparql`, or with the use of a standard in HTTP header for dereferencing. This would however call for adoption of a new standard in web servers used. This could also burden the one who hosts the server because the current web infrastructure was build to identify and locate documents, which now will have the additional burden of serving the data and its metadata linked to a URI. This has to be commercially or socially beneficial to the host.

Alongside adoption of such standards it is also possible to use linked data crawlers like Ldspider (Isele et al., 2010), just like crawlers used in search engines, to identify the SPARQL endpoints and the data that it contains. Ladwig and Tran (2010) describes a method where one can employ an additional step in the execution of a SPARQL query 'to exploit the knowledge discovered at run-time, explicitly scheduled during query processing, called correct source ranking.'

6.3 Architecture of Semantic Web: The Current State

For the successful employment of Semantic Web, it is required to understand what can make it work. The original semantic web vision was published by Berners-Lee et al. (2001), where the importance of RDF and ontologies were mentioned.

In support, the first version (V1) of architecture was introduced by Berners-Lee (2000) in 2000. Antoniou and vanHarmelen (2004) described it as a layered approach and called it semantic web layer tower. Gerber et al. (2008) agrees that the proposed semantic web architecture should be called 'a stack or as 'a layered cake' as called by Hendler (2001), since it does not depict functionality. The first version was subsequently updated by Berners-Lee three times, with the latest one being the version 4 (V4) introduced in the AAAI2006 (Berners-Lee, 2006). All the four version have been depicted in Figure 6.1.

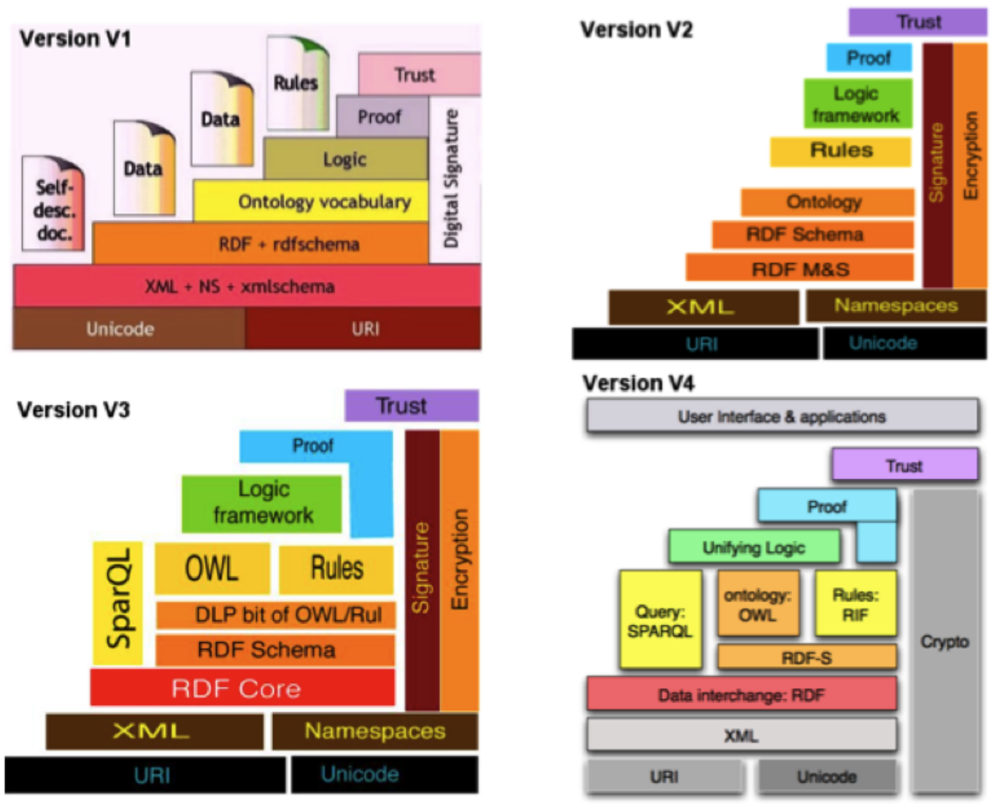


Figure 6.1: The four versions of semantic web architecture proposed by Berners-Lee (Gerber et al., 2008).(Berners-Lee, 2000)(Berners-Lee, 2003b)(Berners-Lee, 2003a)(Berners-Lee, 2003c)(Berners-Lee, 2005b)(Berners-Lee, 2006)

The latest version (V4) of semantic web architecture as suggested by Berners-Lee (2006) lists the core technologies that could make a semantic web application work, with URI and Unicode being the foundation standards. Over these, other standards of XML, RDF, RDF-S, and OWL are established for data modeling and discovery. RIF (Rule Interchange Format) forms the standard for exchanging rules in data processing and SPARQL as the standard for querying data in RDF. Unifying Logic, Proof and Trust are activities that provide support necessary to make user interaction through user interface (UI) effective and do not refer to any specific technologies themselves.

Of all the elements from the stack, trust need not necessarily be treated as a separate entity since it could be measured for many variables. Trust, as a value, for a dataset, could be measured based on its source (the provenance) or for the authentication and communication techniques used. Provenance information could be embedded as a standard in URI or by creating additional data on provenance for every source, using vocabularies like VOID (Omitola et al., 2011). Trust could also depend on the temporal properties of the data, since the validity of a data may change with time. Moreover, trust would need to be quantified for every dataset specific to the user, this is because risk involved in trusting a dataset would depend on the outcome expected from its reuse. By definition, trust is the belief in (expectation of) the behaviour of a party for some given purpose (Klyne and Matthews, 2004).

Lanella (2010) observes semantic web to be evolving into many technologies (Figure 6.1) that provide the support for semantic services, with the core technologies (e.g. RDF, OWL) playing a central role in the overall web architecture. Moreover, since semantic web services use many of these core technologies, one can assume that the architecture adapts to the needs when implemented.

6.4 Semantic Web as a Web of Services: A comparative case study

From the previous Section 6.3, one is able to deduce that semantic web is a concept that can be used with the WWW infrastructure, while the structure of the semantic web stack/architecture would depend on how it's employed. For this specific reason it is important to see an implementation example, where the semantic web concept and its need can be better understood. This Section will discuss an application, where the purpose is to answer a question using SPARQL and compare it with an imperative programming implementation.

6.4.1 The Problem scenario

This case study is set in a hypothetical situation where a person named Tom is moving to the city of 'Southampton' to start his new job at the 'University of Southampton'. He intends to send his daughter to a school which is identified by its post-code, 'SO18 2NU'. He would like to find a house to rent in Southampton with an aim of reducing the travel distance to school and his work place.

The aim of this exercise is to compare the ease of reuse of data, which has been used using a functional programming environment (like Structure Query Language, SPARQL, and Haskell[1]) to an imperative programming environment (like Python[2]). The number

[1]http://www.haskell.org
[2]http://www.python.org

of lines and the file size of the codes for reuse of data written in SPARQL and Python will be used as a measure of ease of reuse.

6.4.2 Hypothesis

The hypothesis: 'The functional programming language, SPARQL makes it easier to reuse data, when the data services are made available as SPARQL endpoints, compared to use of Python, an imperative programming language'.

The null hypothesis: 'The functional programming language, SPARQL does not make it any easier to reuse data, when the data services are made available as SPARQL endpoints, compared to use of Python, an object oriented imperative programming language'.

6.4.3 Methodology

The problem has been simplified by breaking it into simple tasks that can be repeated and they are: (a) the task of finding houses available to buy in Southampton area; (b) getting their addresses; (c) then calculating the net commuting distance to and from Tom's work and school for each house; and (d) then finding the house with the least net commute distance.

6.4.4 The Procedure

There are many house sales agents who have websites, and there are also comparison websites like Zoopla[3] who maintains a database of all the houses to rent from numerous such property agents. To get a list of all the houses available to buy in Southampton, a comparison website (Zoopla) is taken to be the data source. This comparison webpage can be accessed for a specific location (city, town) by requesting the site's webpage, through HTTP protocol, using the required location as a header. A headless-browser[4] is used to send the HTTP request to the Zoopla web server and the HTML web page returned is scraped using Document Object Model (DOM) scripting for the required information of property names, addresses, and listed price.

The other task is to calculate the travel distance to and fro from the houses to the school and Tom's workplace. For this purpose, Google Maps[5] provides a service where one can calculate the commuting distance and time between two locations on earth through an API.

[3]http://www.zoopla.co.uk
[4]http://phantomjs.org
[5]http://developers.google.com/maps/

The final task would be an arithmetic exercise of summing travel distances to calculate the net distance travelled and then finding the least value among them.

6.4.5 Implementation & Results

The backend program that does the various tasks were written in python. It was made as modular as possible to enable the reuse of any class/function that was written. The final working program ended up with three python programs:

spiderzoopla.py: This program does the job of accessing web pages of zoopla using PhantomJS, a headless browser using jQuery[6] and Qt library[7], which can be accessed through Document Object Model (DOM) API using Javascript[8].

skeleton.py : This python file is a library of various functions to access the Google Maps API and returns the required results to the master program in JSON format.

Since this experiment is to study the amount of code required to reuse the various api's and therefore, it is important to understand how the api's were created. The reuse code for python was written in *cobra.py* as the master file where the commands are written according to the logic required to complete the task of finding a house to rent for Tom.

The *cobra.py* takes in the input of name of the city that 'Tom' is moving to and also the location of his school and his work place. The *cobra.py* then calls the *spiderzoopla.py* to return the list of houses with their address and list price based on the input of his new location. For each of the house returned the *skeleton.py* is used to calculate the travel distance to and from, from the new house address to the school location and work address. The reuse python code can be seen in Listing 6.1 and the implementation codes can be seen in Appendix A.

```python
import skeleton
import spiderzoopla, json, time
## The list of functions.
def gmap_addressinput(address_in):
    #Caculating
    gmap_out = skeleton.google_geocode(address_in)# result [address_in]['
    value'])
    return gmap_out

def netdist(a,b,c,d):
    #Calculating the net distance value.
    net_dist = int(a['rows'][0]['elements'][0]['distance']['value'])+int(b[
    'rows'][0]['elements'][0]['distance']['value'])+int(c['rows'][0]['
    elements'][0]['distance']['value'])+int(d ['rows'][0]['elements'][0]['
    distance']['value'])
```

[6] http://jquery.com
[7] http://qt.nokia.com
[8] http://www.java.com/en/download/faq/java_javascript.xml

```
    print net_dist
    return net_dist

###Search inputs###
search_in_city = 'southampton'
price_max= '2100000'
price_min = '10000'
workaddress = 'Southampton univesity, SO17 1BJ'
Schooladdress = 'SO18 2NU'
## Finding the properties in zoopla.
out = spiderzoopla.crawlzoopla(search_in_city, price_max, price_min)
cleanfile = open(out,'r')
cleanjson = cleanfile.read()
results = json.loads(cleanjson)
cleanfile.close()
## Execution of commands.
addressstore= []
i=0 #Setting the inital value of the parameter to zero.
count = 0 #Setting the inital value of the parameter to zero.
shortestdist=1000000000000 #Setting a initial value for the shortest distance value.
for result in (results ['results']['bindings']): #Reading data from the json output.
    if count<4:
        pass
    else:
        sta = gmap_addressinput((result ['address']['value']))
        if sta['status'] == 'OK': #Checking if the data recieved is OK.
            to_work = skeleton.google_distancematrix((result ['address']['value']),workaddress)
            if (to_work['status'])== 'OK':#Checking if the data recieved is OK.
                from_work = skeleton.google_distancematrix(workaddress,(result ['address']['value']))
                if (from_work['status'])== 'OK':#Checking if the data recieved is OK.
                    to_school = skeleton.google_distancematrix((result ['address']['value']),Schooladdress)
                    if (to_school['status'])== 'OK':#Checking if the data recieved is OK.
                        from_school = skeleton.google_distancematrix(Schooladdress,(result ['address']['value']))
                        if (from_school['status'])== 'OK':#Checking if the data recieved is OK.
                            netdistance = netdist(to_work,from_work,to_school,from_school)#calculating the net distance.
                            if netdistance<shortestdist:
                                print 'New shortdistance updated!'
                                shortestdist= netdistance
                                shortestplace = result ['address']['value']
        else:
                print "couldn't calculate distance."
        count= count+1
        i=i+1
print "————————Final answer————————"
print shortestplace
print shortestdist
```

Listing 6.1: Python query used to find a house to rent in Southampton from where the commuting distance to places SO17 1BJ and SO18 2NU identified by their post-codes is the least.

As for the implementation in SPARQL, the spiderzoopla.py and the Google Maps API was made available as a SPARQL endpoint written in python, httpserver.py and googleapiserver.py respectively, using the BaseHTTPServer library. These endpoints were then queried through JOSEKI[9], a SPARQL 1.1 endpoint to JENA[10], which could carry out federated queries. These endpoints of API and the scraper can be considered as services in the semantic web.

To access the house data, the SPARQL endpoint was hosted at `localhost:55000/sparql` and can be used by having the location for the which the houses needs to be found as the subject, `http://www.vocab.org/zoopla/to_rent` as the predicate, and the object gets returned as the addresses of the houses available to rent in the area given as the subject. While the distance between the house and another specified location could be acquired through the SPARQL endpoint which was hosted at `localhost:54000/sparql`. This endpoint can be used by having the house address in the subject field where the predicate (for example) being `http://maps.google.co.uk/distancefrom/abcd` to calculate the distance from house to location "abcd". Likewise, `http://maps.google.co.uk/distancefrom/abcd` can be used to calculate the distance to the house, from location "abcd". The distance is returned in meters (SI unit) to the variable in the object field of SPARQL query.

One must note that the distance calculator service was implemented on the basis that the primary location information is in the subject and the secondary location ("abcd") must be entered as a part of the predicate. The predicate input is forced to follow the structure of URI `http://maps.google.co.uk/distancefrom/abcd`, so as to maintain the human readability and the RDF data model (subject-predicate-object) of that query triple (see Listing 6.2). Another possible method to implement this would have been to write a filter function just like the arithmetic filter functions specified in SPARQL 1.1 standard , but this would mean another user who wishes to use this feature would have to have this filter function available in their SPARQL endpoint.

```
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
```

[9]`http://www.joseki.org`
[10]`http://incubator.apache.org/jena/`

```
SELECT DISTINCT ?houserent (xsd:double(xsd:float(?dist1)+xsd:float(?dist2)+
    xsd:float(?dist3)+xsd:float(?dist4)) AS ?netdist)
{
SERVICE <http://localhost:55000/sparql> { "southampton" <http://www.vocab.
    org/zoopla/to_rent> ?houserent .}
SERVICE <http://localhost:54000/sparql> { ?houserent <http://maps.google.
    co.uk distancefrom/so171BJ> ?dist1 .}
SERVICE <http://localhost:54000/sparql> { ?houserent <http://maps.google.
    co.uk/distanceto/so173rq> ?dist2 .}
SERVICE <http://localhost:54000/sparql> { ?houserent <http://maps.google.
    co.uk/distancefrom/so182nu> ?dist3 .}
SERVICE <http://localhost:54000/sparql> { ?houserent <http://maps.google.
    co.uk/distanceto/so182nu> ?dist4 .}
}
ORDER BY ?netdist LIMIT 1
```

Listing 6.2: SPARQL query used to find a house to rent in Southampton from where the commuting distance to places SO17 1BJ and SO18 2NU identified by their post-codes is the least.

As for the results when the query was run on 19th April 2012 (and again on 17 April 2013), both the methods gave the same address and the net distance value. The accuracy was confirmed from the Zoopla[11] website. The results for both the methods are shown in the Table 6.1. (The actual result is not printed in this document, in order to maintain the privacy of the users of Zoopla website service). This experiment can be repeated by using the python codes provided in the Appendix A.

Table 6.1: The results of executing the query as SPARQL and Python.

Reuse method	**Number of lines of code**	**File size (bytes)**	**Run time (Average of 20 runs in seconds)**
SPARQL	10	711	24.68
Python	57	3072	54.43

6.4.6 Discussion

From the results in Table 6.1, it is clear that the SPARQL method uses fewer code in terms of number of lines and size in bytes. The SPARQL method was also seen to be faster in terms of run time (average of 20 runs) in seconds. Based on the metrics used, SPARQL can be concluded to be the easier method in comparison to python (an imperative programming language).

For another person to be able to reuse *cobra.py*, one must remember that all the associated python files also need to be transferred. Unlike in the SPARQL method, another

[11]`http://www.zoopla.co.uk`

person can run the same query just by running the SPARQL query (Listing 6.2) through a SPARQL endpoint as long as he/she has been permitted to access those endpoints.

In terms of performance even though the SPARQL method was quicker, it also important to note that the SPARQL query speeds are heavily dependant on the size of machine's random-access memory (RAM). This experiment was carried out on a computer with memory rating of 8 GB (1600 MHz DDR3) and a processor rated at 2.6 GHz (Intel Core i7).

Therefore, for the conditions that the experiments were carried out the SPARQL method have outperformed the Python method in all the three metrics and so SPARQL has been more effective with ease of reuse of data. This proves the hypothesis correct.

One could also attribute outcome to the fact that functional programming languages were inherently designed to handle data. SPARQL also has the added advantage, over SQL and Haskell, of access to various other semantic web technologies. It also must be stressed that it is the ability to make SPARQL endpoints available to other people through the internet infrastructure and the ability of SPARQL to handle data better due to its roots in functional programming, makes the reuse of data easier. With SPARQL the user can finally be able to give importance to the data rather than the programming itself.

This example also sets the stage for a semantic web approach, where the data is not necessarily in a database, because of which it could not be analysed like a normal SPARQL query before being used for the purpose of completing the query. So it is vital for one to have the metadata about the data that can come out of a SPARQL endpoint. This example also differs from the usual linked data examples reported, from the fact that the metadata of a dataset now has as equal importance as URI dereferencing in a linked data environment. Thus, showing the need for a semantic web which not just helps to identify a resource but also serves the metadata on data.

The example is also comparable to a data mining activty, where one not only needs to retrieve data, but also reuse it in an easy way. One can conclude that semantic web should not just be considered as a web of linked data, but also as a web of data services as suggested also by Fensel et al. (2011).

6.5 Linked data for Everyone : The Challenge

As a reminder the primary motive of visiting the semantic web concept was to make sparsely linked available. It is important that semantic web helps to make such an ability accessible to everyone. So far the discussion has been from the perspective of a linked data user. For a linked data user, as we have summarised previously, the semantic web is required to identify and locate data and as well be able to deference a URI to be able to find more data about that data (see Section 6.2). As an end result this research expects to have a framework where the web of data can thrive, but to make possible the creation of the web of data, it is also important to consider the needs of the data creator to enable its growth.

For a data creator to be able to contribute to the web of linked data, the person needs to be able to identify existing URIs so that they can be reused thus creating the essential links in the web of linked data. This calls for a the need to have URIs which are also human comprehensible for effective reuse. As we have seen before in the Section 5.4 on minting URIs, two people can have different standards for creating URIs and yet both may be qualify as cool URIs. This raises an issue of how could one promote the reuse of such URIs. The issue arises from the basic structure of a Uniform Resource Locator (URL). The syntax for defining a URL is:

scheme: [//authority][/path][?query][#fragid]

'where the authority is used to identify the server and path identifies a folder or directory in the server, query adds extra parameters and fragid identifies secondary resource' (Fensel et al., 2011).

To mint new URIs one could extend the path or use a new frigid or even extend the authority if the infrastructure permits. This means, if one attempts to mint new URIs by modifying the fragid or the path, the parts ahead of it need not necessarily mean anything to a reader other than helping a URI perform the functions of a URL, i.e., locating the host server that may contain more data about/related to that URI.

Therefore, when a URI owned and made dereferencable by person-A, is used to create data, one wouldn't be able to discover that data by dereferencing it; unless 'person-A' agrees to host that data on the host server or redirects the dereferencing request to the true location of the data. This creates a situation, not ideal, considering one now has to rely on semantic search engines to locate the data that they own because they reused URIs owned by others.

Therefore, one could now refer using URLs to mint URIs undemocratic as it does not support the 'Freedom of Speech' and 'Right to Information' considering the owner of the '[//authority]' get to decide relevancy of data created by others and not the one who wants to consume that data.

Moreover, URLs can be expensive and as explained by Guéret et al. (2011) not everyone would necessarily be able to afford a personal URL.

These notable needs of making linked data affordable, democratic and more importantly making URIs comprehensible in natural language, should be considered important. This is in consideration that the semantic web technologies wouldn't be of any use if the data required to process a query is not discoverable on the semantic web. This calls for the need to readdress the semantic web with a new/modified World Wide Web.

6.6 World Wide Information web (WWI): The proposal

Having established the need and the motive for a new Web, the next step is to establish the design philosophy requirements of this Web. As previously noted in Section 6.4.6, semantic web also needs to be a web of services. A URI can then be referred to as a service agent and from the case study example in Section 6.4 they would be of two types (service agents). The first type is the one that dereferences and provides data related to it or about itself. While the second type requires input of data so as to be able to output data, just like the predicate used to query the Google Maps endpoint in the Listing 6.2. So a new URI can be created by combining two or more URIs, as long as the data output of one becomes the data input of the other. Thus, effectively, one is also able to be consider the combination of URIs as a single URI, where the metadata on data that forms the input and output to a URI needs to be served by the new web.

In this new web, since anyone should be able to publish data about a URI by making their data available when that URI gets dereferenced, what makes a URI unique is the context. Since any word/phrase/sentence could be interpreted differently by different people, the identification of the person who created a new word is useful as context. A URI can then be made unique by incorporating the provenance information of the person who makes it dereferenceable. This can helps in personalisation of the web, where one can hold onto their own data and also be able to decide whom it should be made available to.

Since each URI in this new web can be considered as a service agent and therefore, it could be called the World Wide Information web (WWI). The WWI web could coexist with the WWW, and form the 3rd dimension, if one was to see the WWW as a 2 dimensional sheet of webpages and apps (from a user's point of view) (illustrated in Figure 6.2). With such a web one should not only able to ask questions about 'data' and create knowledge with the 'data', but also help create new data about 'data'.

6.7 Summary

The primary motive of visiting the concept of semantic web was to be able to locate the data that one is looking for in the vast expanses of the internet. Hall and Shadbolt (2009) described semantic web as a web of linked data.

Sir Tim Berners-Lee listed the technologies and standards that could make the vision of semantic web possible through his architecture of semantic web (Figure 6.1). The understanding gained from the discussion in Section 6.3 and 6.4 was that people would use and adopt these technologies and standards based on how they intended to apply the semantic web concept to serve their needs.

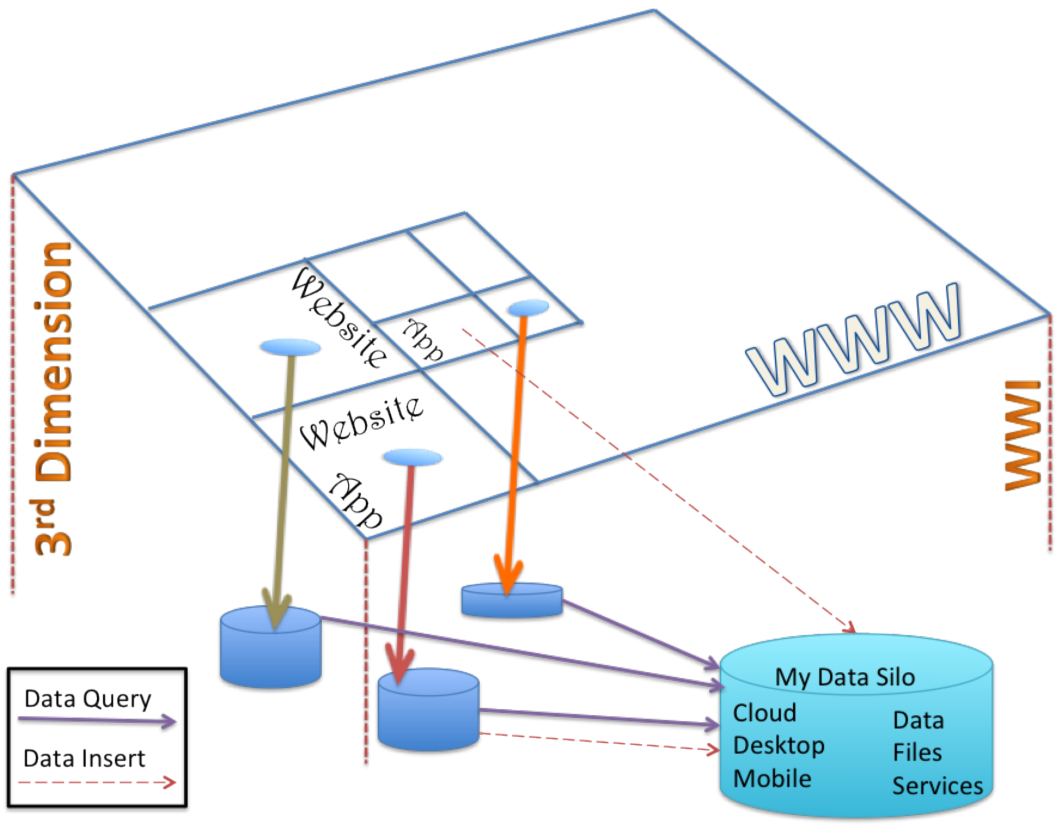


Figure 6.2: An illustration depicting the third dimension that WWI adds to the WWW.

Taking inspiration from the success of WWW in locating documents, semantic web community relied on the URL to fulfil its need to create unique URIs. The uniqueness of a URI is from the fact that the 'authority' part of a URL can be owned only by a single person/group. This empowers the owner of an authority to define a URI (minted on his 'authority') to provide data related to it when the URI is dereferenced. Though, this process also effectively hinders the ability for others, without the permission of the owner of the 'authority', to make their data about that URI available when dereferenced. As discussed in Section 6.4, this can be argued to be undemocratic and so one would have to rely on other services like semantic search to be able to discover this data. Along with the need for a more comprehensible URI and these aforementioned hinderances in Section 6.5 in making the linked data available to everyone, it was concluded that the situation the needs for a new web or modified version of WWW. Section 6.6 introduced the concept of a new Web, to be known as the World Wide Information Web (WWI) and discussed the design requirements and philosophy on how to design and develop this web.

Chapter 7

Discussion & Gap Analysis

"When companies start collaborating, customers win. That collaboration is what will bring semantics onto the web, rather than in semantic silos that have no leverage. If we're going to accelerate the semantic web, it needs to be more useful to more people."

– David Seigal, *2009*

This chapter will summarise the issues that have been discussed so far and brings them together to get a holistic view. The objective is to carry out an effective gap analysis, act as a guide to formulating a solution.

7.1 Gap in the Information Infrastructure: Data Disability

The experience of being unable to retrieve the necessary historic data from the organisation for costing was discussed in chapter 3. This inability can be called 'data disability'. The discussion in Chapter 4 and Chapter 5 attributed the situation to the following reasons:

1. Data are stored in heterogenous mediums. Example application specific data formats include Microsoft tools (Excel, Word, PowerPoint) and emails.
2. Data are stored as poorly structured tables and the lack of a formal, publicised data model makes it difficult for a computer to identify what the data is and its meaning.

Additionally, the datasets were found to be distributed in an uncontrolled manner across both individual computers and in centralised databases which also makes finding them difficult for reuse.

7.2 Solving Data Disability

An analysis of these issues (section 5), suggests that ‘data disability’ is a result of the inability to reuse data without computable instructions specific to the storage format. Therefore, the objective solution is to:

1. Mandate a dynamic data model context when creating and storing data, such that humans and computer agents are able to reuse the data easily, and
2. Use unique references such that the computers are able to recognise what the data is and about, for retrieval.

These solutions in use are evident in the World Wide Web (WWW), where hypertext is used to reference unique documents. The methods of WWW have been extended to data and the equivalent solution, proposed by Sir Tim Berners-Lee, is known as ‘Linked data’. Linked data suggests the use of Uniform Resource Locators (URL) as unique references known as Uniform Resource Identifier (URI) to represent things/concepts. The idea being, the reuse of URIs would create a web of data.

In comparison with WWW, the data retrieved as linked data needs to be reusable by a computer. Therefore, the data needs to be modeled such that the computers are able to easily reuse them irrespective of the format they are stored in. A primary requirement to be able to reuse data presented as a structured table, is to know which column has the subject and which row has the predicate (or vice versa). Linked data overcomes this challenge by adopting the use of Resource Description Framework (RDF). The RDF is a framework in which a data element is called a triple and is represented by a subject, a predicate and an object field.

Exposing the data as triples is especially useful when the need is to find a specific data element that is required from a dataset. Therefore, Linked data with data modeled as RDF is a plausible solution to create machine understandable data.

7.3 The Problem with Linked Data

Linked Data would be an instant success, if every person or software agent (hereinafter referred to as an ‘agent’) attempting to reuse a dataset recognises the vocabulary (schema and semantics) used to create it.

Most of the applications that use linked data currently are built to serve a specific purpose, so they have been able to use ontologies with predefined vocabularies to model the data as RDF. While a few other applications also have had success, as the creators themselves wrote the agents that knew how to reuse that set of linked data when the users interacted through a user interface (UI). Therefore, Linked Data as a concept would work if every person would use a common schema of accepted vocabulary. As a consequence of this practice, there are islands of data in existence with little or no interoperability amongst themselves or even with the linked data that are available openly without restriction for reuse (also known as Linked Open Data (Heath and Bizer, 2011)).

As for Linked Open Data, there is still hope as the issue can be resolved over time with the reuse of certain vocabularies more often than others thus increasing the amount of interoperable Linked Open Data. Berners-Lee (1998b) attributes this phenomenon to what he describes as the 'scale free nature of web'. The effects of the scale free nature of the web were noted by Berners-Lee and Kagal (2008) with the Zipf distribution in the ontology usage. They argued "the fractal patterns of the web systems allows a compromise between having large global ontologies and small isolated ones". Berners-Lee and Kagal (2008) depicts the tradeoff between size and effort versus reuse and interoperability of data with an illustration, as depicted in Figure 7.1.

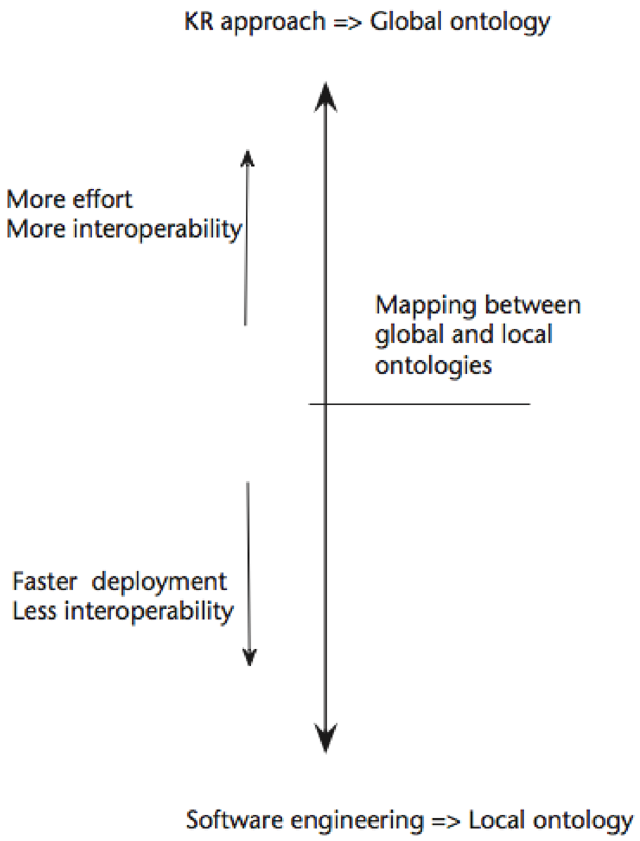

KR approach => Global ontology
More effort
More interoperability
Mapping between global and local ontologies
Faster deployment
Less interoperability
Software engineering => Local ontology

Figure 7.1: "Illustration of Global vs Local ontologies, as depicted in (Berners-Lee and Kagal, 2008) (note that KR stands for Knowledge Representation)."

7.4 The Problem with Linked Data in Typical Commercial Organisations

In a large organisation such as Rolls-Royce with over 40,000[1] employees worldwide, the first challenge is to make sure that every data created by a person/agent is 'Linked data' (in RDF) and secondly ensure that the URIs are reused for maximum interoperability of datasets.

The need to create data as RDF (or any other model chosen) can be solved using appropriate user interfaces like the Open Refine Project[2] supported by Google Inc, that serves the specific and generic needs. The second need to reuse URIs is far more profound and difficult for an engineer who can find it difficult to realise the importance of creating data as linked data that is interoperable, unless the person has attempted to reuse data.

The amount of data that will be understood and reused will depend on how many communities within the organisation have common ontologies/vocabulary. In an organisation, every data created also needs to have had reused the existing vocabulary (URIs) for increased interoperability among datasets that can improve data discovery and make reuse easier. An objective solution to this issue is to make URIs as recognisable as possible to computers and humans alike. The level of recongnisability of a URI is subjective to the person or agent reusing it; hence, (a) information on its provenance that it provides, (b) the ability to see what data has been created by its owner using it (also known as dereferencing a URI), and (c) the ability to see what data has been created by others using it could have a positive impact on its reuse. Therefore, it must be realised that the provenance of URI and the data created can be an important factor.

Considering a situation where in every employee is given a data expert, who is also aware of all the vocabulary (URIs), ensuring the reuse of newly created URIs could still pose a challenge. Any hinderance or restriction on use or dereferencing of URIs could also mean that the data is never created and stored as linked data. Berners-Lee (1998b) comprehends that this compromise between stability and diversity needs to be served by the same amount of structure at all scales.

7.5 Semantic Web and the Ability to Retrieve Data

"The Semantic Web is an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation" states Berners-Lee et al. (2001). However vague the definition of the term semantic web might be, the general conclusion drawn is that the semantic web is what you make of it.

[1] `http://www.rolls-royce.com/about/sustainability/our_people/workforce/people_data.jsp`
[2] `http://openrefine.org`

Therefore, the semantic web in the context of linked data would refer to the existence of the growing linked data graph (section 6.2).

In an organisation, even if the challenge of creating linked data were to be overcome, there is another issue in reuse. With the existence of numerous linked datasets a question arises, 'how is one person/agent to identify the location of the required data in the numerous islands of Linked Data sets?' The obvious solution that can be extended from the web is to use a search service which is aware of the location of the data and can be accessed centrally.

Assuming the use of search engines to be the solution, would be to miss the biggest benefit of having data as linked data, i.e., use of logic to find just the necessary triples instead of retrieving all the data every time. Likewise, data sources could process complex queries themselves and provide the answer using the SPARQL query language (a standard maintained by the W3C[3]). Special storage databases are also available, designed purely to store triples and process SPARQL queries. These methods and technologies have matured enough that the vision of semantic web of being able to 'pull' (Siegel, 2009) the just required or necessary data from a dataset is now possible.

7.6 Pull: Retrieving data & the Methods

Siegel (2009) describes "pulling as the first change in how we use information in over 4,000 years". The information retrieval through pull in short refers to the ability to getting when you want, what you want, where (the way) you want and Siegel (2009) explains that Pull requires a web infrastructure and that semantic web can support it. Pulling the data can be broadly categorised as:

1. *Dereferencing a URI*, is a type of 'pull' where a set of triples is returned to the person dereferencing, at discretion of the URI owner.
2. *Pull-dereferencing:* Pulling the triples that contain a URI in it as subject, predicate or object field from a specific source/graph. 'Pull-dereferencing' is a term newly coined by the author to describe the ability to dereference (or acquire triples that contain) a URI from a specific source. Pull can be useful in the reuse of URI, by finding the URIs previously used by sources that a person trusts. This can be useful in a situation where an access restriction imposed is hampering one's ability to dereference a URI.
3. *Data query:* Pulling the data which abides to the conditions asserted by a specified logic. A SPARQL query is an example.

[3]`http://www.w3.org/TR/2013/REC-sparql11-query-20130321/`

7.7 The Importance of Dereferencing : Understated and often Ignored

Dereferencing can have a significant impact on the reuse of URIs, because it gives the ability to judge a URI based on the graph that can be accessed when dereferenced. Having linked data without the ability to dereference a URI is equivalent to having the World Wide Web without the internet (i.e., not having the ability to pursue a hyperlink). Therefore, the absence of a thorough method for dereferencing can be considered the limiting factor in the chain of events that is required for the success of Linked Data.

This raises the important issue of ownership and maintenance of URIs, since the owner of the URI decides what data gets served on dereferencing. Only through the ownership of all the URIs that one creates can you control what data gets dereferenced and likewise, only through ownership can one maintain it. Another issue concerning maintenance is that, once the primary owner has left the organisation or given up their ownership of the URIs, who should and can then be held responsible for its maintenance.

Querying a dataset from a specific source and pull-derefencing are two other kinds of pull, whose reuse would depend heavily on how well trusted the source is for its quality of data. For a company such as Rolls-Royce, this is a critical issue that if any data is to be reused the consequences need to be auditable. The current Linked Open Data research community have tackled this issue by using the context URI for every triple in a dataset (called named graph) which can hold the information about the source (equivalent to provenance). This solution has led to creating a web of RDF documents and not a web of data. This only partially solved the problem, requiring subsequently a lot of research focused on creating search engines for linked data (like Sindice[4] and Swoogle[5]).

With the use of search engines one can argue that the research community might have missed the vision of pull. While the counter argument is that search engines need only be treated as agents on the network that act as a service and are useful due to the sparseness in the nature of existence of Linked Data on the internet. This is true for the Linked Open Data, but not to provide an opportunity to pull data from the Linked Data Graphs within an organisation would be to miss the whole benefit that pull can bring about in the way we communicate as described by Siegel (2009).

7.8 Summary

This objective of this chapter was to summarise the discussion held so far and create objective requirements to help formulate a solution. From the analysis of an attempt of

[4]`http://www.sindice.com/`
[5]`http://swoogle.umbc.edu/`

a broad data reuse activity for unit costing in Chapter 3, the need to solve heterogeneity and sparseness of the data were realised. The need to use a standard data model to describe all data was found to be the accepted solution in literature to make data machine reusable. Therefore, an ability to extract data from the legacy documents as loosely structured tables is not sufficient to reuse them. With the availability of provenance information of every legacy document, crowd sourcing application of data model to the tabulated data when recalled by a person doing a similar role was concluded as the solution. The need then was understood to have an environment where such data can be held and reused.

The research challenge was therefore established as to find an appropriate framework for data reuse in an organisation that can support broad data reuse activity.

Subsequent analysis in Chapter 5 has shown that the creation of the linked data graph is a plausible solution to the problem of heterogeneity. Upon discussing the challenges in creating linked data in Section 7.3, it was understood that having an infrastructure that can support the creation and growth of linked data is critical. The key to reusability of a linked dataset though further lies in having a common vocabulary of URIs, which requires reuse of existing URIs. The positive impact from dereferenceabilty of URIs and the need for provenance in creating linked data were discussed in Section 7.4. Further analysis placed the emphasis of having URIs that are comprehensible by humans and computers alike.

Meanwhile, in order to solve the challenge in creating and reusing linked data due to the sparseness in the existence of data on the web, the need to evolve the Web was discussed in Chapter 6. The current practice was found to be to use named graphs to encapsulate the information on the provenance and the location of the data. It though was found to have created just another web of documents referenced by the named graphs that contain the respective linked data. The use of search engines for data retrieval in such situation then becomes a solution and this can create a situation where dereferenceability of URI is neglected.

Derefenceability is necessary to find what concept or thing the creator intended a URI to represent and it is understood that the ability to retrieve useful data upon doing so about that URI can also aid in finding more data. In addition, the discussion in Section 7.7 found the need to incorporate into the URI provenance which also reflected its ownership as critical to control what is served when dereferenced.

The aim of this research therefore needs to focus to create a web of linked data by evolving the web of documents (Web) into a web of data. This would therefore, call for the necessary modification/addition to the ways the WWW infrastructure is used and create a framework on which Linked Data can reside upon. To remind us of the framework that WWW resides on, the hypertext dereferencing infrastructure is made of:

1. Navigation standards (URL structure)
2. URL navigation infrastructure which includes Domain Name System (DNS) servers
3. Communication protocols (like Hypertext Transfer Protocol) which retrieves the information in a readable format.

To direct the changes that are required to the WWW, Section 7.7 described the importance of creating an environment where one can pull the necessary data. This chapter also showed that the solution to the research question 'What is an appropriate framework for data reuse in an organisation to support a broad data reuse activity', does critically need to be able to aid the creation and organic growth of linked data. The creation of this 'web of data' though has a specific set of challenges that it needs to help overcome. Reiterating the list of areas that needs attention and are to be aided are:

1. URI discovery and reuse through provenance
2. URI ownership
3. URI maintenance
4. Ability to pull data by dereferencing a URI
5. Ability to discover data through pull

As noted in Section 7.4 the need to encapsulate the information on the provenance of ownership in a URI is a common factor to these. Therefore, the conclusion is that, a solution to the primary research question would be a framework that can solve all these challenges and create a web of linked data.

Chapter 8

World Wide Information Web

"The Difficulty lies not in the new ideas, but in escaping from the old ones."

– John Maynard Keynes, *Cambridge*

World Wide Information Web, boldly introduced previously as the new Web, has only been a verbal formulation of the needs for the web of data that the web is expected to fulfill. In summary, the WWI was deemed to be an environment that enables the creation of data as linked data in a corporate environment and equip the user to reuse the data. This chapter aims to present the WWI with a formal description using examples and formal methods. To understand the WWI, one needs to understand the corporate environment where the WWI would be deployed.

8.1 Security by Privacy in the Corporate Environment

In the corporate environment, the security of data was observed to be a common sentiment among the employees. The degree of data sharing was based on trust one placed on the person who is being allowed to have a copy of or access to a certain dataset. While, the trust on the accuracy of data is dependent on which department or authority it was obtained from. This approach is referred to as 'Security by Privacy'. It has its shortcoming in terms of the significant amount of time required in collating information from multiple sources; as experienced by the author while attempting to collate data for a data mining project within Rolls-Royce, where the security was primarily ensured in two steps:

- Through the use of secure communication technologies (encrypted portable disks), cryptography, etc.
- By adopting 'security by privacy' model in managing information.

The author believes this model of security can be adapted to digital medium with the adoption of methods like ‘policy based security’ which can automate the access control. This chapter will be focused on formalising WWI as a framework where creation and reuse of linked data by adopting a ‘Security by Privacy’ model of information identification and retrieval, while assuming the use of secure and up-to-date data communication and storage technologies.

8.2 WWI URI on the WWI Framework: A Solution

As a solution that can solve all the problems summarised in Chapter 7, the attempt is to include the information on the provenance of ownership of a URI within itself. The WWI framework aims to create a web of data, by creating a web of ‘things’ (people/agents which are living or non-living entities) that create and own the data. The WWI framework has a standard for (a) how URIs are created for these ‘things’, (b) how the ‘things’ can create new URIs to actively contribute to linked data growth, (c) how to locate the URIs on the internet, (d) how to ensure dereferencing would provide useful data, and (e) how pull-dereferencing can be made possible.

1. *URIs for Things:* WWI imposes a standard to identify ‘things’; every ‘thing’ needs to have had issued a unique URI for every role that it does to create data. A URI of a thing will be based on the stakeholder hierarchy for its role in the organisation. This is a necessary condition, for every ‘thing’ that intends to actively contribute (also known as ‘active thing’) to the growth of the linked data graph. Therefore, every WWI URI reflects its provenance as its stakeholder hierarchy in the organisation.

2. *‘Things’ can create URIs for ‘things’ and concepts:* Any ‘active thing’ that intends to create a new URI to represent a thing/concept would have to use its own URI as prefix and add the new thing/concept as affix. An ‘active thing’ refers to a ‘thing’ that creates and/or stores data.

3. *Locating URIs on the internet:* The WWI framework requires every URI of an ‘active thing’ to be at service upon dereferencing to provide the graph of data that it contains, and know the location of every other URI of ‘active things’ that it owns. Each ‘active entity’ therefore also acts as a subdomain DNS resolver.

4. *Pull by dereferencing:* To ensure useful data is provided upon dereferencing a URI, WWI framework has adopted the policy of ‘data as a service’. This policy aims to enable the concept of dereferencing to thrive. Every URI of an ‘active thing‘, will be dereferenced by itself, thus making it act as a portal. While it also requires every URI of things that are not ‘active’ be dereferenced by the URI of the ‘active thing’ that created or owns it.

5. *Making pull-dereferencing possible:* The policy of 'data as a service' indirectly helps make pull-dereferencing possible by ensuring every URI is dereferenceable, since this effectively makes it possible to use every URI as a named-graph URI. The WWI framework also enforces an additional policy of 'URI of things as a Data portal'; whereby the URI of every 'active thing' is required to be a portal to the data that it stores and to the data stored by the URIs that it created and owns. This policy with all the others will change the way we pull-dereference/ query from the sparse data graphs.

The World Wide Information Web will be based on this framework and standards, with an aim to enable the growth of the linked data graph in an organisation/enterprise and ensure the maintainability, discoverability, and reusability of the data. To ensure compatibility with the existing tools and methods, the existing systems will be reused as much as possible and thus, WWI will be an addition to the WWW standards.

8.2.1 The WWI URI and Organisational Structure

A URL is the fundamental standard that makes the WWW a reality. The WWI URI is same as a WWW URI with an additional requirement that the URL hierarchy should reflect the organisational stakeholder hierarchy. In this research a person or group referred to as 'X' is a stakeholder of an individual or a group referred to as 'Y', if the achievement of an objective by Y can affect X and any data/information/knowledge created by Y is owned by X. The author understands that different organisations would have different definitions for a stakeholder and wishes to leave it to the judgement of an individual organisation. The primary purpose of creating WWI URIs is to ensure that the provenance of every URI is captured in context of the organisation.

As an example, imagine a fictitious organisation 'Z' that owns the URL, `http://joshuajeeson.com`. Z has has two divisions, namely, Research and Sales. Clive Roberts is the CEO of Z. The research division is lead by division head Raj Gnanendran. Likewise, the sales division is lead by division head Ricky Algama. The WWI URIs would then look like:

Clive Roberts : `http://joshuajeeson.com/CEO/CliveRoberts`

Raj Gnanendran: `http://joshuajeeson.com/Research/DivisionHead/RajGnanendran`

Ricky Algama : `http://joshuajeeson.com/Sales/DivisionHead/RickyAlgama`

8.2.2 WWI URI for Things

Any one who wishes to contribute to the linked data graph in the WWI framework would require a URI specific to their role. When a new URI is minted, it would be based on the URI of the person that is creating it, i.e., any new thing that requires a URI will have to use URI of the creator as its prefix.

As an example, if Clive Roberts from the example in Section 8.2.1 was to create a new URI for a thing 'MXP', he would require to use his URI `http://joshuajeeson.com/CEO/CliveRoberts/` as prefix and joined by '/'. Thus, the new URI for the thing MXP would then be, `http://joshuajeeson.com/CEO/CliveRoberts/MXP`.

Likewise, it should be noted that any entity in the organisational stakeholder hierarchy is allowed to create URIs and hence, the same URI could have been created by the highest authority in organisational hierarchy, i.e, if the company had declared the new URI for the thing MXP, it would have been `http://joshuajeeson.com/MXP`.

8.2.3 Data as a Service: Maintenance and Ownership of URIs

The WWI framework is based on the paradigm of 'data as a service', due to which any WWI URI when dereferenced will retrieve a graph with the triples that has it in any of the triple field or as context. This ability has a significant impact on how the data is viewed, since every URI now represents a graph and is dereferenceable, WWI URIs are thus Cool URIs (see Section 5.3.2).

This paradigm has also direct implications on the ownership and maintenance of the WWI URI. A WWI URI can be split into the resource and prefix. The prefix stands as the provenance for any WWI URI. Therefore, the ownership, maintenance, access and storage can all be based on this provenance structure of the URI. The stakeholder hierarchy was purposely chosen as the suitable provenance structure because of the ownership rule that is assumed with it, i.e., the stakeholder owns the data or resources. Thus, any URI is owned by the creator described in its structure and the ownership is inherited along the stakeholder hierarchy to the domain of every URI upon which the WWI framework is constructed. Therefore, even though the primary ownership is with the creator, the ownership is also automatically inherited by every stakeholder in the prefix.

Likewise, the maintenance of a URI expected to be carried out by its owner, in an event of his/her absence, the maintenance responsibility can be inherited by any of its stakeholders who are also rightful owners.

8.3 WWI Linked Closed Data with Security by Privacy

Cobden et al. (2011) refers Linked Closed Data to "datasets which adhere to the principles of Linked Data publishing, but for which access to and use of the data is subject to legal or technical restrictions which go beyond attribution and share-alike obligations."

It is the general aim of this research to make easy the creation and maintenance of linked data and create an environment where the data may be distributed, discoverable, and secure. So, WWI framework aims to provide an environment where Linked Closed Data can reside and grow.

8.3.1 Data as a Service for Security by Privacy

The 'data as a service' paradigm is not restricted to WWI URIs but extends to any data that resides in the framework, i.e., each and every URI of the entity/person in the organisation is a portal where the data that they create is stored. This portal is called the Data Unit (DU) is the gateway to the data held by its owner and to the portals to which it is the stakeholder.

The direct implications from the adoption of 'Data as a service' paradigm is on how the data can be stored and accessed. On the aspect of data storage, each individual entity or person of the organisation can store their data over sparse locations. Their DU would index these data with its respective location and act as the sole point of access to all that data. The use of DU's to create the WWI web ensures that every entity or person in the organisation can store the data that they create wherever they wish to and can be accessed from their respective DU. This also helps to incorporate the 'security by privacy' approach into data seeking on WWI web, over which, further access control methods could be applied individually to each DU.

Moreover, two WWI URIs could be considered the same as long as the stakeholder hierarchy is reflected in the right order, irrepective of whether each of them is part of the authority or path of the URI. To understand this, revisiting the example used in Section in 8.2.1, the DU of the Ricky Algama could also have been queried directly by constructing the URI as `http://RickyAlgama.DivisionHead.Sales.joshuajeeson.com/`. Similarly, if the person accessing doesn't have a direct access to Ricky Algama's DU but has been given permission to access Division head's DU, the user could issue a query to Division head's DU and request it to reverse proxy on Ricky Alagama's DU. The URI would then look like `http://DivisionHead.Sales.joshuajeeson.com/RickyAlgama/`. This is an example of 'Security by Privacy' in action, where a stakeholder who own's the data can take the decision on sharing the data that he owns directly or indirectly.

8.3.2 Connecting Distributed Data through Provenance

Of the many components that DU requires, a SPARQL query processor is one. Since the context for a data is represented by a URI (also referred to as named graph (Carroll et al., 2005)), in WWI web any named graph is also a WWI URI. A named graph when dereferenced would provide all the triples with it as the context, subject, predicate or as object, stored in the DU of the owner of the named graph URI. A named graph is, as a result, equivalent to a document on the web.

Along with the policy of ‘data as a service’, the framework also enforces an additional policy of ‘URI of things as a Data portal’; whereby the URI of every ‘active thing’ is required to be a portal to the data that it stores and to the data stored by the URIs that it created and owns. This policy and the individualistic existence of DU’s as servers, provides the WWI web the additional ability where a DU can query the DUs for which it is a stakeholder as proxy for people or agents who do not have direct access. Thus effectively connecting the dataset through provenance.

8.3.3 Data as a Service for Organic Linked Data Growth

The ability to pull the necessary data from the linked data graph is one of most useful capability that the linked data graph provides and so it is necessary that WWI provide the right environment to necessitate the growth of linked data graph.

The provenance structure implied by the WWI URI has significant effect on a reuser by being able to judge the acceptance of a term. Revisiting the example from Section 8.2.1 with Clive Roberts, once MXP is a commonly approved term by every Chief Engineer in Future Programmes, the prefix of the MXP URI could be replaced with the URI of CEO thus having `http://joshuajeeson.com/CEO/MXP`. A reuser could take this as a sign of acceptance of the term MXP across the umbrella of Chief Engineer. Thus a reuser would be able to get a sense on the acceptance of the term MXP, from its associated URI. Consequently, the WWI URIs can be a key enabler in the creation of linked data without an ontology in a closed environment; where the new terms can be added to their existing ontology, as and when there is enough data being created about a thing and its properties.

This situation can be considered ideal for the growth of linked data graph, considering it as an approach somewhere between the a top down and bottom up approach to ontology creation. This claim is based on the argument by Berners-Lee and Kagal (2008), where it is argued that due to the fractal nature of the semantic web a combination of approaches would be the effective solution to induce growth of linked data as depicted in Figure 7.1.

Berners-Lee (1998b) suggested that one should expect the semantic web to demonstrate Zipf distribution in the reuse of data from various sources considering the inherent fractal nature of language and culture in human societies. Berners-Lee and Kagal (2008) observed the Zipf distribution in the ontology usage and argued that “the fractal patterns of the web systems allows a compromise between having large global ontologies and small isolated ones” (see Figure 7.1).

The author appreciates the potential power of the scale-free nature of semantic web and sees the WWI as providing the necessary framework to work somewhere in between the global and local ontology approach with the incorporation of provenance into every WWI URI. In WWI with the existence of different levels of the stakeholder hierarchy of a URI, the author anticipates that the provenance and multi level vocabulary with 'data as a service' will also provide the environment for ontology-less linked data creation and graph growth. This would be an approach catering more towards the need of engineers for faster deployment of data as linked data.

The 'data as a service' paradigm adoption by WWI ensures that every URI is dereferenceable and are COOL URIs, so one could anticipate increased rate of reuse of URIs. This is because a reuser can then see what data a URI has been previously used by its creator. This is especially beneficial when one creates a new URI when unsure of the existence of previously created URI, the use of stakeholder hierarchy in WWI URI structure can increase its reusability. In addition, the provenance structure in every WWI URI could enable one to infer the relevance and acceptance of a term, thereby aiding the growth of the linked data graph. The users would also be able to develop a sense of trust on sources, for the reason that one is able to infer the person or entity responsible for its maintenance. This could then be used to crowdsource their levels of trust or impression on the quality of data from each DU to be used as guidance to identify the nodes in the linked data graph that needs improvements.

8.4 WWI Linked Closed Data: Formal Definition

A formal definition is now required to establish the framework of WWI. It can be an important guide to the developers to remain consistent with the standards. To formulate WWI, the concepts and principles of linked data needs to be established first.

8.4.1 RDF tripleset and Named Graphs

RDF is the fundamental data model that is used in the creation of linked data . Table 8.1 lists the definition of the sets and functions that describe an RDF graph, as outlined in the W3C RDF syntax by Klyne and Carroll (2004).

Definition 8.1. (RDF triple and tripleset) An RDF triple is denoted by $(s, p, o) \in (\mathbb{U} \cup \mathbb{B}) \times \mathbb{U} \times (\mathbb{U} \cup \mathbb{B} \cup \mathbb{L})$; where s is the subject, p is the predicate and o is the object. The RDF tripleset $\mathbf{T}$ is a subset of $(\mathbb{U} \cup \mathbb{B}) \times \mathbb{U} \times (\mathbb{U} \cup \mathbb{B} \cup \mathbb{L})$.

An RDF tripleset can be described as a graph (Chein and Mugnier, 2009). Angles and Gutierrez (2008) noted that, "The application areas of graph models are those were information about the interconnectivity or the topology of the data is more important,

Table 8.1: Definition of sets and functions for RDF graph.

$\mathbb{U}$	Set of URI references
$\mathbb{B} = \{\_a, \_b, \_c, ...\}$	Set of Blank Nodes
$\mathbb{L}$	Set of Literals, including plain literals and typed literals
$\mathbb{V} = \mathbb{U} \cup \mathbb{B} \cup \mathbb{L}$	Set of Vocabulary; $\mathbb{U}$, $\mathbb{B}$ and $\mathbb{L}$ are pairwise disjoint
$\mathcal{U}(\mathbb{V}), \mathcal{B}(\mathbb{V}), \mathcal{L}(\mathbb{V})$	The set of URIs, Blank Nodes and Literal in $\mathbb{V}$, respectively
$sub(\mathbb{V}), pred(\mathbb{V}), obj(\mathbb{V})$	All the elements appear in $\mathbb{V}$ as subject, predicate and object, respectively

or as important as, the data itself. This is usually accompanied by the fact that data and relations among data are at the same level." There are other representations which include labelled directed multigraph (LDMG) (Baget, 2005), bipartitle graphs (Hayes and Gutierrez, 2004) and labelled directed hypergraphs. This research will use the labelled directed multigraph (LDMG) as its simplistic form serves the purpose, while a representation of RDF as a labelled graph would inherently be a labelled directed multigraph since there can be many arcs between two nodes and two arcs can have different labels Baget (2005). Baget (2005) and Li (2011) described an RDF tripleset as a Labelled Directed Multigraph, as follows:

Definition 8.2. (RDF tripleset as Labelled Directed Multigraph) Let $\mathbf{T}$ be an RDF tripleset, $\mathbb{V}$ be the set of terms in $\mathbf{T}$ and the LDMG representation of $\mathbf{T}$ is $\mathbf{G}$. Each triple $t = \langle s, p, o \rangle \in \mathbf{T}$ represents a distinct arc $a(t)$ and each entity $e \in ent(\mathbf{T})$ represent a distinct node. A labelled directed multigraph of over a set of terms $\mathbf{G}$ is a 4-tuple $\mathbf{G} = \langle N, A, \gamma, \epsilon \rangle$, where $N = \{n(e) | e \in ent(\mathbf{T})\}$ is a finite set of nodes, $A = \{a(t) | t \in \mathbf{T}\}$ is a finite set of arcs, $\gamma : A \to N \times N$ maps each arc to a pair of nodes $\gamma(a(t)) = \langle n(s), n(o) \rangle$ given $s \in sub(\mathbb{V})$, $o \in obj(\mathbb{V})$, and $\epsilon : N \cup A \to \mathbb{V}$ maps each node and arc to a term in $\mathbb{V}$ as $\epsilon(n(e)) = e$ and $\epsilon(a(t)) = p$ given $p \in pred(\mathbb{V})$.

Most of the modern triplestores provides an option for the storage of triples with context as the fourth column also known as Named Graph. Since contexts are necessary for the provenance of the data, a tripleset with context needs to be defined.

Definition 8.3. (RDF tripleset with context, a 4-Tuple, as tupleset) An RDF tuple is an RDF triple with context denoted by $(s, p, o, ng) \in (\mathbb{U} \cup B) \times \mathbb{U} \times (\mathbb{U} \cup \mathbb{B} \cup \mathbb{L}) \times \mathbb{NG}$, where $\mathbb{NG}$ be the set of named graphs and $\mathbb{NG} \subset \mathbb{U}$. An RDF tupleset with context $\mathbf{T}^c$ is a subset of $(\mathbb{U} \cup B) \times \mathbb{U} \times (\mathbb{U} \cup \mathbb{B} \cup \mathbb{L}) \times \mathbb{NG}$.

8.4.2 Mathematical Description of Linked Data Principles

The linked data principles were listed and briefly discussed in Section 5.2. The first principle states that all things be identified by URIs. According to the architecture of the World Wide Web (Jacobs and Walsh, 2004), a URI can either be an information resource or a non-information resource on the Web. `http://joshuajeeson.com/Users#JJD` is an example of a non-informational URI. It is considered so because when the URI with fragment part `JJD` when dereferenced retrieves the same document as another URI with the same prefix (`http://joshuajeeson.com/Users`). Fragments were initially introduced to identify a part of a HTML document and so the fragment part is never send to the host servers. Let $\mathbb{U}^I$ be the set of information resources and $\mathbb{U}^N$ the set of non-information URIs. Then $\mathbb{U}^I \subseteq \mathbb{U}$; $\mathbb{U}^N \subseteq \mathbb{U}$; and $\mathbb{U}^I$ and $\mathbb{U}^N$ are disjoint.

The second principle requires all things to be defined by HTTP URIs.

Definition 8.4. (URIs are HTTP URIs) Let $\mathbb{U}_H$ be the set of HTTP URIs and $\mathbf{G}_H$ be the RDF graph with the terms $\mathbb{V}$, where $\mathcal{U}(\mathbb{V}) \subseteq \mathbb{U}_H$. The set of information URIs and non-information URIs are defined by $\mathbb{U}^I_H$ and $\mathbb{U}^N_H$, respectively.

The third principle requires the URIs to be dereferenceable and when dereferenced provide the user with useful information. A dereferenceable HTTP URI can be broadly categorised based on the response code and for the response information that one receives (Li, 2011). The first type is when an informational resource URI is dereferenced and receives a "2XX" status code. The second type is where a non-information resource is dereferenced and the user receives a "2XX" status code, where the user then needs to identify themselves the part of the received information that is relevant to the dereferenced URI. Another possible outcome is a "3XX" response, which redirects a non-informational URI to another informational URI. Such acts where non-information URI needs to be resolved to get an informational URI, can be represented as $resolve(u^N_H) = u^I_H$. Other possible outcomes include a "4XX" or "5XX" response, which as defined by httpRange-14[1] indicates nature of that resource is unknown and are described as non-dereferenceable can be represented by $u_H \in \mathbb{U}^K_H$.

Definition 8.5. (URI is dereferenceable) Dereferencing of a URI can be defined as a partial function $deref : \mathbb{U}_H \nrightarrow \mathbf{H}$, where:

$$deref(u_H) = \begin{cases} deref(u_H) & for \quad u_H \in \mathbb{U}^I_H \\ deref(resolve(u_H)) & for \quad u_H \in \mathbb{U}^N_H \\ undefined & for \quad u_H \in \mathbb{U}^K_H \end{cases}$$

Upon dereferencing a URI the data retrieved as a graph or a document could be empty, i.e., $\mathbf{H} = \emptyset$; this is a bad practice and therefore, a person traversing the graph reaches an end.

The fourth principle to be followed while publishing linked data is to use URIs from other domains such that the datasets are traversable. This is a key activity that can increase the interoperability of datasets.

[1]`http://lists.w3.org/Archives/Public/www-tag/2005Jun/0039.html`

Definition 8.6. (Linking Datasets to Linked Data Cloud) Let $\mathbf{T}_H^C$ be the the set of all triples published in the linked data cloud and $\mathbf{T}'_H$ be the new set of triples being added to the linked data cloud, where $\mathbf{T}'_H \nsubseteq \mathbf{T}_H^C$. The URIs of $\mathbf{T}'_H$ and $\mathbf{T}_H^C$ are $\mathbb{U}'_H = \mathcal{U}(\mathbb{V}(\mathbf{T}'_H))$ and $\mathbb{U}_H^C = \mathcal{U}(\mathbb{V}(\mathbf{T}_H^C))$, respectively. By fourth principle of linked data the dataset $\mathbf{T}'_H$ can be said to be interlinked to the cloud, if the $\mathbb{U}'_H \cap \mathbb{U}_H^C \neq \emptyset$.

8.4.3 WWI URI: Formal Definition

A URI is a fundamentally important entity in the creation of Linked Data and its reuse imminently connects graphs together. The WWI framework has a formal requirement to create URIs based on the stakeholder hirearchal structure of the organisation. Thus, to formally establish the definition of a WWI URI, an organisational stakeholder structure needs to be defined which could be represented as a Labeled Directed Rooted Tree (LDRT).

Definition 8.7. (Labeled Directed Rooted Tree Representation of organisational stakeholder hierarchy) Let $\mathbb{X}$ be the set of elements in the organisational stakeholder graph, r being the root of the tree, where $r \in \mathbb{X}$. Then the LDRT is a graph $\mathbb{W} = (\mathbb{X}, \mathbb{E})$, where (x, y) is the edge, with $(x, y) \in \mathbb{E}$ or $(y, x) \in \mathbb{E}$; $x \in \mathbb{X}$; $y \in \mathbb{X}$; and $x \neq y$.

The structure of an URL is described in Section 8.4.3. A WWW URI created based on the URL would have the basic components 'scheme', 'authority', and 'path'. The use of the fragment has been purposely avoided, because the fragment is never part of a HTTP request and this means the server that is dereferencing the URI is unaware of the thing that is being dereferenced. This can lead to an increased unnecessary usage of network bandwidth, because an entire graph/document which has information related and unrelated to the URI being dereferenced gets downloaded every time. The author considers the use of non-information URIs not conferring to the 'security by privacy' as a potential risk the security and privacy. Therefore, to serve the purpose of the WWI framework the structure of a WWI URI can be represented as:

scheme: [//authority][/path]

Definition 8.8. (WWI URIs are information URIs) Let $\mathbb{U}_W$ be the set of WWI URIs, then $\mathbb{U}_W \subseteq \mathbb{U}^I$, $\mathbb{U}_W$ and $\mathbb{U}^N$ are disjoint sets.

The first principle in the WWI framework is to use the organisational stakeholder hierarchy to create new URIs for every person or entity that wishes to contribute to the organisations linked data graph. Therefore, a URI of any entity or person in the organisation can be created from the path that connects the root to self in the organisational stakeholder hierarchy LDRT.

Definition 8.9. (WWI URI of an entity/person as a path in the organisational stakeholder hierarchy LDRT) The elements from the LDRT that forms the WWI URI $u_L \in$

$\mathbb{U}_W$ for a entity/person is a set represented by $elements(u_t) = \{m, x_L, \mathbb{K}\}|\mathbb{K} = \{x_1, x_2, ..., x_{L-1}\}$, $\mathbb{K} \subseteq \mathbb{X}$, $m \in \mathbb{M}$ with $\mathbb{M}$ representing the set of scheme's, $x_1 = r$ and $\{x_L, x_{L-1}\} \in \mathbb{E}$.

Since the aim of WWI is to promote the storage of data with 'security by privacy', the data from the perspective of a user is effectively locally distributed. The applications therefore can be considered to be centrally distributed, which means different applications could have different standards of communication over the internet. As an example, a HTTPS protocol might be preferred over a HTTP by certain applications to fulfill the needs of security, while for applications with data inherently encrypted at source the use of HTTP might be preferred for increased data transfer rates (e.g., MEGA[2]). The WWI framework recognises this need to use different schemes to dereference a URI and so would assume two URIs to be the same if and only if all their elements are same and in same order, except the scheme.

Definition 8.10. (WWI URIs can use any scheme) Let $u_a \in \mathbb{U}_W$ and $u_b \in \mathbb{U}_W$ two URIs. u_a and u_b can be treated as the same URI, if and only if, $(elements(u_a) \cup elements(u_b)) - (elements(u_a) \cap elements(u_b)) \subseteq \mathbb{M}$, where $\mathbb{M}$ is the set of schemes and $\mathbb{K}(u_a) = \mathbb{K}(u_b)$.

8.4.4 Mathematical Description of WWI Principles: Data as a Service

In WWI web, all the WWI URIs are informational resources (U_W^I) and the DU is expected to process a dereferencing request received as a HTTP request or SPARQL request alike. The DU server can be made to handle multimodal access by the use of pre-defined headers which could identify the method that it should use to process the request.

As in Section 8.4.2, the WWI URIs can be categories based on the outcome. The first type is where a WWI URI dereferenced receives a "2XX" status code as reply. Second type is where the the reply is "3XX" status code with a redirection to another URI, such a response could be based by a DU on who is attempting to dereference the URI. In such a situation the URIs are represented by $\mathbb{U}_W^R$ and they need to be resolved before dereferencing, represented as $resolve(u_W^R) = u_W^I$. The final type is where the reply code is either "4XX" or "5XX", in which case the resource is unknown and is described as non-dereferenceable can be represented by $u_W \in \mathbb{U}_W^K$.

Definition 8.11. (WWI URIs as Named Graphs: Data as a service) Dereferencing of a WWI URI can be defined as a partial function $deref : \mathbb{U}_W \nrightarrow \mathbf{H}$, where:

$$deref(u_W) = \begin{cases} deref(u_W) & for \quad u_W \in \mathbb{U}_W^I \\ deref(resolve(u_W)) & for \quad u_W \in \mathbb{U}_W^R \\ undefined & for \quad u_W \in \mathbb{U}_W^K \end{cases}$$

[2] `http://mega.co.nz`

Since every WWI URI can be dereferenced to get a graph of data, they can be treated as a namedgraph to the data that it serves. Depending on the request method and request format of the data, the retrieved information can then be reused as a document or graph.

8.5 Discussion

The previous chapter had discussed the need for a URI to reflect its provenance and ownership to support URI discovery and regular maintenance. Taking this into consideration the WWI framework is proposed where every WWI URI is constructed from the stakeholder hierarchy and every data contributor is required to have their/its own URI. Any new WWI URI then can be created by its owner just by adding their/its WWI URI as prefix to a term.

To support data discovery by dereferencing, WWI framework adopts the policy of 'data as a service' whereby every WWI URI when dereferenced leads to a data unit which serves the dereferenced data. With the provenance as stakeholder hierarchy, the owner has the control over what data is served and so the ability to maintain it. Thus, the WWI inherently supports security by privacy.

The provenance can also help a reuser to judge what a URI represents and can also act as a reference to the quality of the data that it serves. Moreover, the provenance can also enhance URI reuse with better able to recognise the URIs that have been used by one's colleagues or someone else in a similar role before. This existence of implied structure in a WWI URI aims to provide an environment for organic growth of linked data where centralised ontologies are not a necessity.

As for the needs stated and summarised in Section 7.8, the WWI has tackled the need for URI discovery and reuse through provenance by embedding the stakeholder hierarchy of an organisation into the URI thereby inherently letting anyone creating a URI assume its ownership which is necessary for its maintenance. This has also helped to satisfy the need to make every URI cool to help grow the web of linked data. Yet there is still a need to understand how WWI can help overcome the specific challenges of reuse due to the sparseness in the location of data and support the ability to pull the necessary data. Hence before WWI can be deemed as the answer to the research question, this research needs to understand the challenges from the sparseness and then evaluate how WWI can help overcome them.

Chapter 9

Data Reuse with WWI

"I foresee two things happening with Linked Data. One is from the web development perspective (the so-called Web 2.0 developers) and the other is from the enterprise perspective. The web development community will sooner than later realize that Linked Data will enable easy integration of data and therefore will ease the pain of consuming data from different data sources. From the enterprise perspective, Linked Data is another data integration solution. ... Imagine being able to query 'get all my clients from cities whose population is greater than 1 million' even though I don't have the data about population of cities in my database."

– Juan Sequeda, *Executive Director at Semantic Web Austin*

This research is now left to see how well the WWI framework helps to solve the challenge of sparseness in location of data. This is an important requirement in being able to assess the conformance of WWI as the answer to the research question. With the creation of web of linked data, one must first understand the inherent challenges in reusing it.

9.1 Data Reuse with Web of Linked Data

With the creation of a web of linked data, the tools and practices would need adoption of newer techniques and methods in addition to the existing ones, be it either to take benefit of the situation or to solve existing challenges. The author does understand the significance of user interfaces (UI) in data reuse especially since many users are more familiar with data in tabulated form due to the extensive usage of Microsoft Excel. Though UI design will be out of scope of this research, the review of methods of reuse is necessary.

9.1.1 Review of the Choices in Exposing Data

Primarily, linked data can be stored as a data dump in a specific storage format or in a dedicated triple (or quad) store. The choice of a method has notable difference on how the data can be accessed.

A data dump would require a data reuser to load and read all the data from it before an attempt can be made to use the required data from it. This will have a significant impact on the bandwidth of network connecting various data sources and as well as on the random-access memory (RAM) required by the user. For the data owner, a data dump could be cheaper in terms of providing access to the data, since processing queries can be computationally expensive. While for the data reuser, a data dump would mean a greater use of computational and RAM resource. For every new data set that needs to be reused, there is a need to process all the data first.

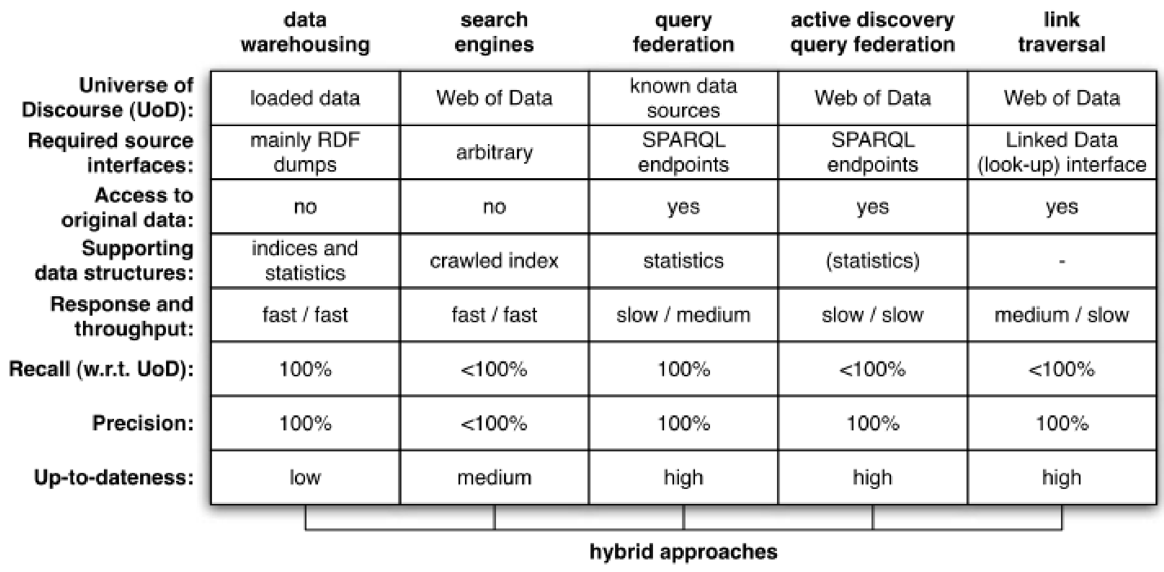
In comparison, access to a triple (or quad) store from an endpoint provides the user with the benefit of being able to query just the necessary data. In such a situation the computational cost gets distributed to the endpoint providers. This is useful for the user because of the the fact that they need not invest in a machine with high RAM and processing capabilities which can be expensive. In a cost comparison, the data dump requires to move large chunks of data all the time and so the owner would have to bear the cost of network bandwidth. Therefore, having a mechanism to compensate each data owner for the cost of data processing or network usage can be a useful practice to transfer the costs to the reuser.

However, providing a SPARQL endpoint has the additional benefit, to the data owner, of having control over their data in terms of what is being accessed and in certain situations (from the query received) what they are being used for. Therefore in comparison to the data dump method, for the benefit of security and privacy a query (SPARQL) endpoint would be the choice.

9.1.2 Review of Reuse Methods

From the perspective of a data reuser after having given access to the various datasets, there is still the challenge of finding the required data and then reusing them. Hartig and Langegger (2010) lists the following methods to find and use data, considering the sparse nature of their existence:

1. *Data Warehousing* is an approach where data is collected and then stored in central database (Chaudhuri and Dayal, 1997). Data warehousing can be specifically useful to reduce the amount of network communication that is required. This is specifically true when a lot of data is available as data dumps. The query



	data warehousing	search engines	query federation	active discovery query federation	link traversal
Universe of Discourse (UoD):	loaded data	Web of Data	known data sources	Web of Data	Web of Data
Required source interfaces:	mainly RDF dumps	arbitrary	SPARQL endpoints	SPARQL endpoints	Linked Data (look-up) interface
Access to original data:	no	no	yes	yes	yes
Supporting data structures:	indices and statistics	crawled index	statistics	(statistics)	-
Response and throughput:	fast / fast	fast / fast	slow / medium	slow / slow	medium / slow
Recall (w.r.t. UoD):	100%	<100%	100%	<100%	<100%
Precision:	100%	<100%	100%	100%	100%
Up-to-dateness:	low	medium	high	high	high

hybrid approaches

Figure 9.1: Query processing approaches and some of their properties as depicted by Hartig and Langegger (2010).

processing by a data warehouse can be termed efficient by its use of central indices and statistics as shown by Abadi et al. (2009), Neumann and Weikum (2008), and Weiss et al. (2008). Though it has its shortcomings as noted by Umbrich et al. (2010). The key issue being that the data would not necessarily be up-to-date since it can be difficult to keep the central repository in sync with the original sources.

2. *Search Engines* An alternative approach is to use search engines like Sindice (Oren et al., 2008) which crawl the Web by following the links and then provides a query interface to the index of discovered data. (Hartig and Langegger, 2010) notes that this method has to overcome the challenge of being able to process complete SPARQL queries.

3. *Query Federation* is an approach based on distributing the processing of queries to multiple, autonomous sources (Sheth and Larson, 1990). These autonomous sources then executes the sub queries and returns the results. The benefit of this approach being that there is no need to synchronise the copied data, there is no need for additional storage, and the computational cost can be distributed among the autonomous sources.

4. *Active Discovery Query Federation* is an approach based on the assumption that the location of the required data is not necessarily known when the query is executed. Therefore, this process inducts a method of active source discovery which is used once a query is exectued. Hartig and Langegger (2010) notes that this method can be useful as an addition to the federated approach since it assumes the complete Web as the universe of discourse. Though finding all the sources

of linked data on the Web can be considered impractical. Also when a new data source is found their characteristics need to be known to make them useful and this might require an additional description for each source with a specific vocabulary.

5. *Link Traversal* is another method that assumes that the location of the required dataset is unknown. This method applies a technique for traversing the links automatically of the things the query is about. Hartig and Langegger (2010) notes this is an approach inspired from the Tabulated Linked Data browser by Berners-Lee et al. (2006).

To have a direct comparison Hartig and Langegger (2010) tabulated the properties of each the methods as depicted in Figure 9.1. The properties of universe of discourse, the required source interface, on whether there is access to the original data, the data structures that support this activity, the speed of response and throughput, recall rate with respect to the universe of discourse, precision of query, and the up to dateness of the retrieved data are the factors upon which they have been contrasted.

Though search engines and query federation are two completely different approaches, with the former using crawled index, the notable difference is the access to original data cannot be guaranteed by the search engines and that the precision is not necessarily a 100%. The query federation also has a higher chance of being able to use up-to-date data, but at the cost of the universe of discourse being the known sources.

On the other hand, data warehousing using indices and statistics and can match the query federation technique in terms of recall and precision of 100%, but the query federation can more likely guarantee access to original and up to date data. Data warehouse though is a great technique to provide useful access to data that have been exposed as data dumps through a query interface. The final two techniques of active query federation and link travel can be treated more as a useful add-on to the other techniques to be able to expand their universe of discourse to the entire web of data.

It is therefore not necessarily apt to choose one method from either of these since the purposes they can fulfil are different and as concluded by Hartig and Langegger (2010) a hybrid approach is also an option.

9.2 The Challenge of Sparseness

As discussed the choice of a data storage environment and the query technique will need to be tailored to the needs of the situation. It is therefore necessary to explore the challenges faced in completing a query on the web of data. The linked data principles primarily connect sparsely located data through the reuse of identifiers (URIs) to represent data irrespective of the source. However, knowing the location of the required data in order to complete a specific data query is still a challenge . This in a nutshell is the challenge of sparseness of data.

9.2.1 Effect of Resource Constraints on Sparseness

The primary difficulty to reuse data that are distributed would be in terms of time and effort required to find the data from the various sources. The use of search engines is an obvious choice in such a situation, but as noted previously search engines aren't yet capable of processing an entire query by itself and so cannot be used on its own.

While, the option of accumulating all the data into a warehouse is not a choice either since in terms of the time and computational resource required to process a query can be high and certain queries that are expensive could also lead to a server connection time out.

Another choice is to federate a query to various SPARQL endpoints, however, Acosta et al. (2011) notes that the query executions may frequently be unsuccessful because of the lack of adaptivity of SPARQL queries. Acosta et al. (2011) identified that such an issue arises primarily because many of the endpoints are developed for very lightweight use and so if a query which requires more than 3 minutes to complete execution is posted against such endpoints (like linkedCT endpoint[1]), the endpoint will timeout without producing an answer. The solution to this problem would be to decompose a SPARQL query into simple sub-queries which can then be executed in a reasonable time. Neumann and Weikum (2009) and Vidal et al. (2010) uses query optimisation techniques and efficient physical operators to speed up the execution time. Weiss et al. (2008) and Neumann and Weikum (2009) attempts to tackle this challenge by having defined structures to store and efficiently access RDF data. To speed up subsequent queries Atre et al. (2010), Neumann and Weikum (2009), and Idreos et al. (2009) makes use of a technique to reuse data previously stored in cache.

However, a critical requirement to be able to process a query from federated sources is to join triples from them. This requires the knowledge of where the required individual triples are located in the federation. When one is aware of the sources and the location of the required triples, techniques like ANAPSID by Acosta et al. (2011) and FedX by Schwarte et al. (2011) could be used to gather linked data by decomposing queries into subqueries which can then be executed by selected endpoints.

[1] `http://www.w3.org/DesignIssues/LinkedData.html`

9.2.2 Effect of ‘Security by Privacy’ on Sparseness

The current solutions to solving sparseness is a two step process of first having a catalog of the location of the various data sources and second step is mapping what data is available from them.

From the perspective of an organisation the primary requirement is to be able to control the access to data. Thus the use of a central search engine is not necessarily a secure method from fact that a single agent is being given access to index all data. Besides the knowledge of what data is available, the agent would also know where it is located. Therefore, an attempt to breach security to control this single agent will be of high value.

Likewise, the current techniques that help optimise the triple join operations like DEFENDER by Montoya et al. (2012), Anapsid by Acosta et al. (2011), and FedX by Schwarte et al. (2011), all requires access to a catalog of the available sources of information.

Many organisations and in particular Rolls-Royce which work on military projects with operations in more than one country follows the strict security policy that, if a person or agent is not authorised to access a server, then the server shall not reply or even acknowledge its existence in any manner. This policy thus directs one to enforce ‘security by privacy’. Another direct implication is that, it would be best for the organisation to avoid creation and maintenance of a directory of the endpoints that are in existence. This situation is contradictory to the needs of current data discovery and reuse techniques and therefore, calls for an additional step where various data sources can opt to participate in a completing a query and let a reuser know how and where it can be accessed without the need for a global endpoint catalog.

9.3 Data Reuse with WWI

In a corporate enviornment, the trust on the source of data is a key factor which can play a part in deciding its reusability. The impact of an incorrect data can be profound especially for an activity like unit costing of gas turbine in Rolls-Royce, where the predicted cost value is used to make further decisions on procurement of materials, factory planning, and many other activities that could affect the lifecycle of a product.

Information quality, therefore, in general, is a challenge. Hartig and Langegger (2010) notes that every RDF triple published can only be considered as a claim from the data provider. The WWI framework has taken this into account and hence every data creator is required to have a URI build upon their position in the organisations stakeholder hierarchy. These data creators were referred to as ‘active contributors’ with each of them having a data unit (DU) located by their URI.

The universe of discourse in a WWI web, therefore, can be limited to the active contributors within the company. Trusted suppliers, partners, and other external sources can also be part of the WWI web once they have also been issued a WWI URI.

In order to choose a specific method in making data available, the requirements of application area needs to be understood. As a reuser in a corporate scenario like Rolls-Royce, having access to the *original* data that is *up-to-date* data is critical. The queries when issued also needs to have a precision of 100% and the recall rate of the data also needs to be a 100%. Looking at the Figure 9.1, query federation is the only approach that fits in with these requirements that an organisation would need, though it has the down side that the response and throughput rates can be significantly lower compared to warehousing and search techniques.

9.3.1 Locally Distributed Data & Centrally Distributed Applications

A DU can be considered to be a data warehouse and the WWI web as a federated data warehouse environment connected by their stakeholder hierarchy. Jindal and Acharya (2004) noted that the iterative nature of a Federated Date Warehouse reduces the extended delays normally associated with data warehousing solutions. They concluded that a Federated Data Warehouse, with its lower costs and higher reliability, provides an excellent value proposition to organisations and a very practical solution for implementing and delivering value added solution to a business.

Since each DU is independent and is taken to be the sole gateway to all the data that its owner has, the owner is responsible for the data that leaves his or her DU. Therefore, if one has multiple identities, the addition of provenance information into the data needs to be done only when it leaves one’s DU. This enables one to store all their data anywhere on the web sparsely and yet have any application attain the data that it needs through the DU. This situation can be referred to as the paradigm of *‘locally distributed data and centrally distributed applications’* from the perspective of the data owner.

If applications accepted a common vocabulary, the owner can easily switch between applications based on its merits of its UI, since the data is be stored by its rightful owner where ever they wish. Even in situations where different people or agents who may know one by different identities can still be served the same original data with different provenance information. This is another direct benefit from the way WWI framework inherently adopts ‘security by privacy’.

Moreover, since the federated DU environment in the WWI web have a SPARQL end-point for every DU, it also has the benefit from the fact that it is effectively a federated query environment with access to up-to-date and original data.

9.3.2 Provenance Driven Querying

The new opportunities that WWI creates are primarily due to the way in which the DU's are connected with its structured URIs reflecting the embedded provenance. In contrast this would be the primary difference between DU's on WWI to its unstructured existence on the web with non explicit provenance. This opens up novel ways in which a SPARQL query can be constructed and executed.

The current method of SPARQL 1.1 allows to place three variables per query line, i.e., at subject, predicate, and object position. SPARQL 1.1 also allows for the prefixes to be predefined and make them fixed with only the suffix as variable. The syntax to define a variable is to use a '?' as prefix (e.g., $?subject$ being a variable) or have the prefix fixed and joined to the variable with ':' and '?' (e.g., $prefixa$:$?subject$ where $prefixa$ is predefined).

With WWI there is an additional need to be able to use a variable at suffix as well as prefix. It is important to note that the suffix variable would only represent the term after the final '/' for WWI URIs and '#' for non-informational URIs. This addition can mainly cover two situations in addition to the usual SPARQL query construct:

1. where the user is unaware of the exact provenance information. The variable space would then have a variable each for the prefix and another for the suffix, i.e., $?prefix$:$?subject$.
2. where the user is only aware of the source authority and not the exact DU that owns the URI. This would require an ability to fix the initial part of the prefix, while the suffix and the unknown part of the prefix is declared as a variable, i.e., $prefixa$:$?prefix$:$?suffix$, where $prefixa$ is pre-defined.

9.3.3 'Domino Request' on WWI

The WWI creates a unique situation where there is not a requirement to know or catalog of all DU's that are in existence. A DU only requires to know the DU's to which it is a stakeholder. Therefore, one only ever needs to know the primary stakeholder. Since the universe of discourse is limited to the sources that have a WWI URI and along with the policy of 'URIs of things as a Data portal', the situation enables one to issue what could be called as the 'domino request'.

A 'domino request' initiates a process where by a DU that has recieved the request will process it and then would forward the 'domino request' to all the URIs to which it is a stakeholder, which are then expected to repeat the same. With a single domino request it is therefore possible to pull-dereference any data that has ever been created in an organisation, just by issuing a 'domino pull-dereferencing request' to the 'root active thing'. A 'root active thing' of an organisation is identified by the URI of the organisation itself, upon which all the other URIs are based.

A domino request to the 'root active thing' effectively enables a reuser to treat all the graphs that is held by the organisation as though it is a single graph. It can also be issued as a SPARQL query, thus enabling one to issue complex queries to all the data in the organisation as if it were in a single database. However, as the reuser is unaware of what the total number of DU's that are in existance at the time of issuing a domino request, the reuser can never be sure how long it would take for a domino request to reach all of them and so defining a time limit to a domino query can be a useful way to let the domino request to stop propagating.

Since a domino request involves getting reply from multiple DU's, the way in which the resultant replies are collated needs to be standardised. One technique is where the reply to a domino request is collated and replied back by the portal to which the request was primarily send. Another method could be to have the information on the availability of accessible data be send as reply to the issuer directly by each of the portals that receives the request, which can then be used by the issuer themselves to optimise the query into a federated one. These methods need to be developed and adapted as per the requirements and would form part of future work. The concept of domino request further opens up the research questions on what the best method would be to execute it when the constraint is speed, time, accuracy, efficiency or another metric of a query.

The method of sending a domino request also does need to be standardised as a HTTP (or HTTPS) request that can be tagged as a domino request by declaring a predefined Header or as a new application layer protocol. A new standard in SPARQL is also required where one can issue a domino request like the SERVICE request used for federated querying. In addition, the DNS system would need adaption to receive and process user credentials before processing any request.

9.4 Solving the Sparseness of Data : Demo

Since with domino request there is no more a need to maintain a catalog or directory of endpoints, the only need is to know the primary endpoint that is the root active thing. The WWI framework and the domino request together has therefore solved the final challenge in solving the challenge of reusing sparse data without the exact knowledge of where the endpoints are. This subsection will demonstrate this ability with a test case scenario.

9.4.1 Scenario & Setup

The example scenario selected is the UK Police Crime Data obtainable from `http://police.uk`. The available data contains information on the total number of crimes listed categorically into types (like burglary, anti social behaviour, total crime etc.) at Neighbourhood level identified by an area code and individual crime information (that includes crime id, crime type, latitude and longitude of crime location, outcome etc.) from the respective police force which registered the crime.

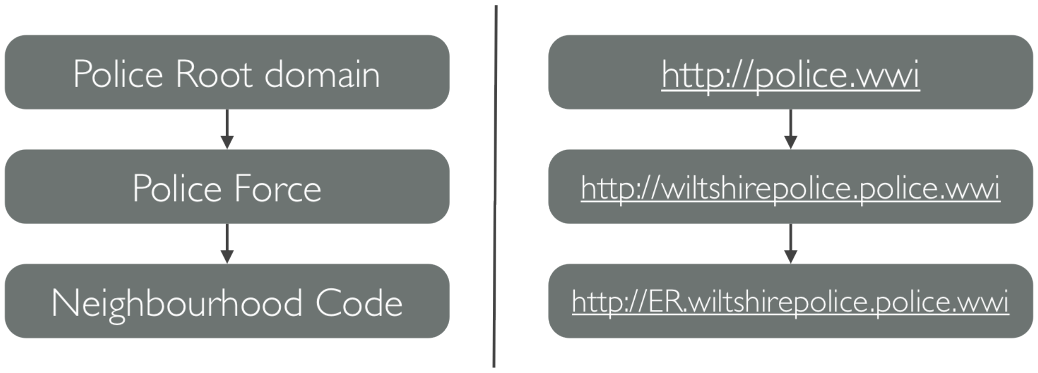

Police Root domain
Police Force
Neighbourhood Code
http://police.wwi
http://wiltshirepolice.police.wwi
http://ER.wiltshirepolice.police.wwi

Figure 9.2: Illustration of creation of active URIs for every neighbourhood identified by neighbourhood code (e.g. ER), based on the URI for its respective police force (e.g. Wiltshire Police).

```
PREFIX rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX xsd:<http://www.w3.org/2001/XMLSchema#>
PREFIX police: <http://police.wwi/>
<http://police.wwi/WiltshirePolice/ER>  rdf:type  police:Neighbourhood ;
  police:isUnderForce      police:SuffolkConstabulary ;
  police:TotalAllCrime   140^^xsd:int ;
  police:TotalAntiSocialBehaviour 57^^xsd:int ;
  police:TotalBurglary   10^^xsd:int ;
  police:TotalCriminalDamageAndArson 14^^xsd:int ;
  police:TotalDrugs 2^^xsd:int ;
  police:TotalOtherTheft 11^^xsd:int ;
  police:TotalPublicDisorderAndWeapons 3^^xsd:int ;
  police:TotalRobbery 0^^xsd:int ;
  police:TotalShopLifting 11^^xsd:int ;
  police:TotalVehicleCrime 8^^xsd:int ;
  police:TotalViolentCrime 21^^xsd:int ;
  police:TotalOtherCrime 3^^xsd:int .
<http://police.wwi/Wiltshire+Police/ER/2013-01-wiltshire-neighbourhood.
    source> rdf:type police:fileurl ;
  police:PublishedOn    "2013-01-01"^^<http://www.w3.org/2001/XMLSchema#date
    > ;
    police:ExternalDownloadLink    <http://policeuk.s3.amazonaws.com/frontend/
      crime-data/2013-01/2013-01-wiltshire-neighbourhood.zip>
```

Listing 9.1: Crime statistics by type stored in triplestore of DU of neighbourhood identified by the code 'ER' belonging to the Wiltshire police stored under the named graph `<http://police.wwi/WiltshirePolice/ER/2013-01-wiltshire-neighbourhood.source>`.

The WWI framework for this scenario has been created based on the root domain `http://police.wwi` which has child URIs for every police force (e.g., `http://wiltshirepolice.police.wwi`). Each police force URI would further have child URIs for every neighbourhood (e.g., `http://ER.wiltshirepolice.police.wwi`) as illustrated in Figure 9.2. All the force and neighbourhood URIs lead to their DU. The crime statistics are then stored in individual triplestores of their DU's which are to be identified by the area code as illustrated in Listing 9.1.

9.4.2 Aim

The objective is to collate the average total burglary reported for every police force. The data would be available as represented in Listing 9.1. The required data could be retrieved using the query in Listing 9.2, if and only if all the data were in the same triplestore. The aim of this demo is to be able to collate all the required data from federated triplestores without having access to a catalog containing the location of all the endpoints.

```
PREFIX police: <http://police.wwi/>
SELECT DISTINCT ?force (AVG(?value) as ?AvgValue) WHERE {
    ?neighbourhood   police:TotalBurglary     ?value ;
       a      police:Neighbourhood ;
       police:isUnderForce     ?force .
} GROUPBY ?force
```

Listing 9.2: The SPARQL query to collate the average total burglary reported for every police force, if all the data were in the same triplestore.

9.4.3 Implementation

The URIs of the police forces, neighbourhoods and the police root URI are all active URIs with each one of them leading to a unique data unit (DU). Every DU has a triple store and can be queried directly if one is aware of its location. The DUs report their location to their respective parent DU, with its URI being based on the hierarchy as represented in Figure 9.2.

A domino request as discussed in subsection 9.3.3 can be executed in many ways and this demo will take an approach where every DU that receives the request will acquire data for every SPARQL query line and send back the data without performing the join. Listing 9.3 represents the SPARQL query that will be processed by child URIs when it receives a request like in Listing 9.2. The join action is then carried out only by the DU that originally received the domino request and this type of domino request can be referred to as 'Domino-All'. All the domino request implementations would be based on the assumption that the required data could be anywhere within the WWI.

```
PREFIX police: <http://police.wwi/>
SELECT DISTINCT * WHERE {
OPTIONAL {
    ?neighbourhood  police:TotalBurglary    ?value . }
OPTIONAL {
    ?neighbourhood  a    police:Neighbourhood . }
OPTIONAL {
    ?neighbourhood  police:isUnderForce    ?force . }
} GROUPBY ?force
```

Listing 9.3: The SPARQL operation carried out by child DU's that recieve a domino request.

The access to domino request can also be extended such that a domino request can be send to an endpoint just like a SPARQL federated SERVICE request, using the terminology of DOMINO-ALL followed by the URL of the endpoint with the query enclosed in closed curly brackets '{ }' as in Listing 9.4.

```
PREFIX police: <http://police.wwi/>
SELECT DISTINCT ?force (AVG(?value) as ?AvgValue) WHERE {
  DOMINO-ALL <http://police.wwi> {
    ?neighbourhood  police:TotalBurglary    ?value ;
      a    police:Neighbourhood ;
      police:isUnderForce    ?force .
  }
} GROUPBY ?force
```

Listing 9.4: The SPARQL query rewritten with 'DOMINO-ALL'.

9.4.4 Results

The data is available for 44 individual police forces and a total of 4577 neighbourhood codes. The total number of servers that would need to be created will be 4622. Therefore, the scope of this demo will be limited to 2 forces with 2 neighbouring codes each.

The neighbourhoods chosen were ER (`http://police.wwi/WiltshirePolice/ER`) and ES (`http://police.wwi/WiltshirePolice/EP`) that belong to Wiltshire Police (`http://police.wwi/WiltshirePolice`) and J11 (`http://police.wwi/SuffolkConstabulary/`

J11) and H41 (`http://police.wwi/SuffolkConstabulary/H41`) of Suffolk Constabulary (`http://police.wwi/SuffolkConstabulary`).

The query in Listing 9.2 was send to the primary root URI of the Police at `http://police.wwi` as a domino request. Upon receiving the request the domino request were forwarded by it to its children DU's of Wiltshire Police and Suffolk Constabulary which forwarded them further onto their respective children. All the DU's reconstructed the query as shown in Listing 9.3 and their respective triplestores were queried.The DU's then combined and send their results along with the results received from their children DU, if any. The Police root DU which originally recieved the domino request upon receiving all the results, queried the results with the original query and the results were send back.

The total burglary neighbourhood wise were 10 in ER, 22 in EP, 21 in J11 and 11 in H41. The result obtained from the demo was that the average total burglary force wise were 16 for Wiltshire Police and 16 for Suffolk Constabulary. The results were also confirmed manually.

9.4.5 Discussion

With this demo it has been demonstrated that with the domino request on WWI it is now possible to create and store data as linked data with 'security by privacy' and yet be discovered and reused without the need for a catalog of the endpoints to be maintained. Therefore, the WWI framework with domino request can be deemed as a solution to the research question "What is an appropriate framework for data reuse in an organisation to support a broad data reuse activity".

The demonstration though has shown light on some of the practical problems. The execution time for the demo was noted to be 1.44 seconds (average of 5 runs) and the SPARQL query produced the result for the two forces of Wiltshire and Suffolk. In comparison, executing the same SPARQL query with the data from a single triplestore for all the 44 individual police forces and 4577 individual neighbourhoods produced the result in 0.27 seconds. This is clearly due to the total amount of request and response times involved in completing the domino request. Rakhmawati and Hausenblas (2012) concluded that the more dataset that is involved in query federation, worse the performance gets.

However, to speed up the process a DU can be allowed to maintain an index or statistics of data held by all the DU's to which it is a stakeholder and yet only have the need to know which of the DUs to which it is a parent should a domino request further be send to. This can be replicated at every parent DU and reduce the number of DUs that receives a domino request. Another way is where the DUs which receives the domino request could send back the statistics of the available data that can fulfil the SPARQL query, along with the URL at which it can be contacted. This information can then be used to optimise the SPARQL query for the sequence in which join operations needs to be carried out. While, in cases where the department or the parent URI is a trusted source, the query can be narrowed down by reuser themselves. The domino request could be executed in numerous such ways and the exploration of faster methods will need to be done and would be part of future work.

In addition, with domino on WWI, since the data does pass through the stakeholders, any of the DUs can hide the provenance part of the URIs to prevent letting the reuser know identity of the exact sources. The URIs could also be replaced with temporary URIs for the purpose hiding the identity of the items themselves and the data, and if numerical they could also be normalised if the purpose of reuse was to data mine. In addition, a reuser who is unaware of where the data is residing could send out a request for access to the data if any available to every DU, which can be processed depending on the requesters identity and the purpose of reuse.

In a large organisation being able to ask for permission to use the required data with a single request from unknown sources with every stakeholder having a say on what the data is being used for can be an important capability. This ability can be vital especially when the need is to to be able to build data mining models, as was the case with this research project with its initial aim to be able to build a model to predict the unit cost of a gas turbine.

9.5 Application Case Study - Airline Industry

While defining WWI for this research, the need was to have an environment where it would be easier to create a variety of data as linked data yet be discoverable for reuse. The stakeholder hirearchy was thus embedded as provenance into every WWI URI keeping in consideration the security and privacy needs of the data. This creation of inherent structure into the way in which data is stored also opened new ways to retrieve and query the data.

The discussions and examples so far have been based on the scenario where a data unit has had a single identity. This Section would like to explore the benefits of WWI when the same data has multiple stakeholders. Since all of one's data is stored in their own DU, in addition to being able to provide access to original and up-to-date data, this Section would like to demonstrate the benefit from adopting multiple identities to promote trust as well as to maintain privacy.

The scenario that this case study would investigate is the commercial aircraft industry. With the availability of satellite based high speed internet, form providers like Avanti

Plc[2] and Eutelsat[3], based on KA-band spectrum for high speed internet communication (Hu and Li, 2001), the aircrafts now have the ability to stream various parameters live to a system located outside itself while in flight. There is a possibility of adoption of this technology in the future, especially since the airline industry now functions with a variety of business models.

In recent years there has been a notable change in the ownership model of the inventory by airlines. The new leasing model provided by the manufactures of aircraft engines like the TotalCare from Rolls-Royce [4] assures a working engine that conforms to a certain performance standard when required by the leaser. Such contracts would also stipulate the conditions and limits in the flight envelope that the airline can subject the aircraft to. Therefore, an ability to communicate in real time about the various operational conditions and associated parameters by engines could help the leaser plan the servicing routine much ahead of its arrival at the hanger. A live communication feed would also mean similar data could be used by the airports to optimise the landing order of aircrafts to reduce emissions or other operational activities in busy airports. Therefore, this is a scenario with multiple stakeholders that require access to the same data, but before we can explore the application of WWI, we must first identify the major stakeholders that prevails in the commercial aircraft industry.

9.5.1 Identifying Stakeholders

For the purpose of this case study, only two types of active contributors will be considered and they are the aircrafts and its engines. Within the operational & commercial framework of an operational aircraft, four major stakeholders have been identified to the data that the aircraft or the engine creates. They are the aircraft manufacturer, the engine manufacturer, the airports federation and the inventory owner (airline or leaser).

All the data generated by the aircraft and the engine are primarily to be stored within its own computers from which control systems operate. This case study will generalise the owner of the individual engine or aircraft as the lessor. Therefore a lessor owns the data that their inventory produces. Most manufacturers are also granted access of some operational data by lessor such that they can discover common issues that occur across a product and inturn help to prevent them from occurring in other products. Figure 9.3 depicts the organisational stakeholder hierarchy to represent how the aircraft and engine data would fit into the satekholders of manufacturers and lessors.

From an operational point of view, aircrafts operate between two airports and so maintenance usually would be carried out at the local airport from where its operates. This

[2]`http://www.avantiplc.com`

[3]`http://www.eutelsat.com`

[4]`http://www.rolls-royce.com/civil/services/totalcare/`

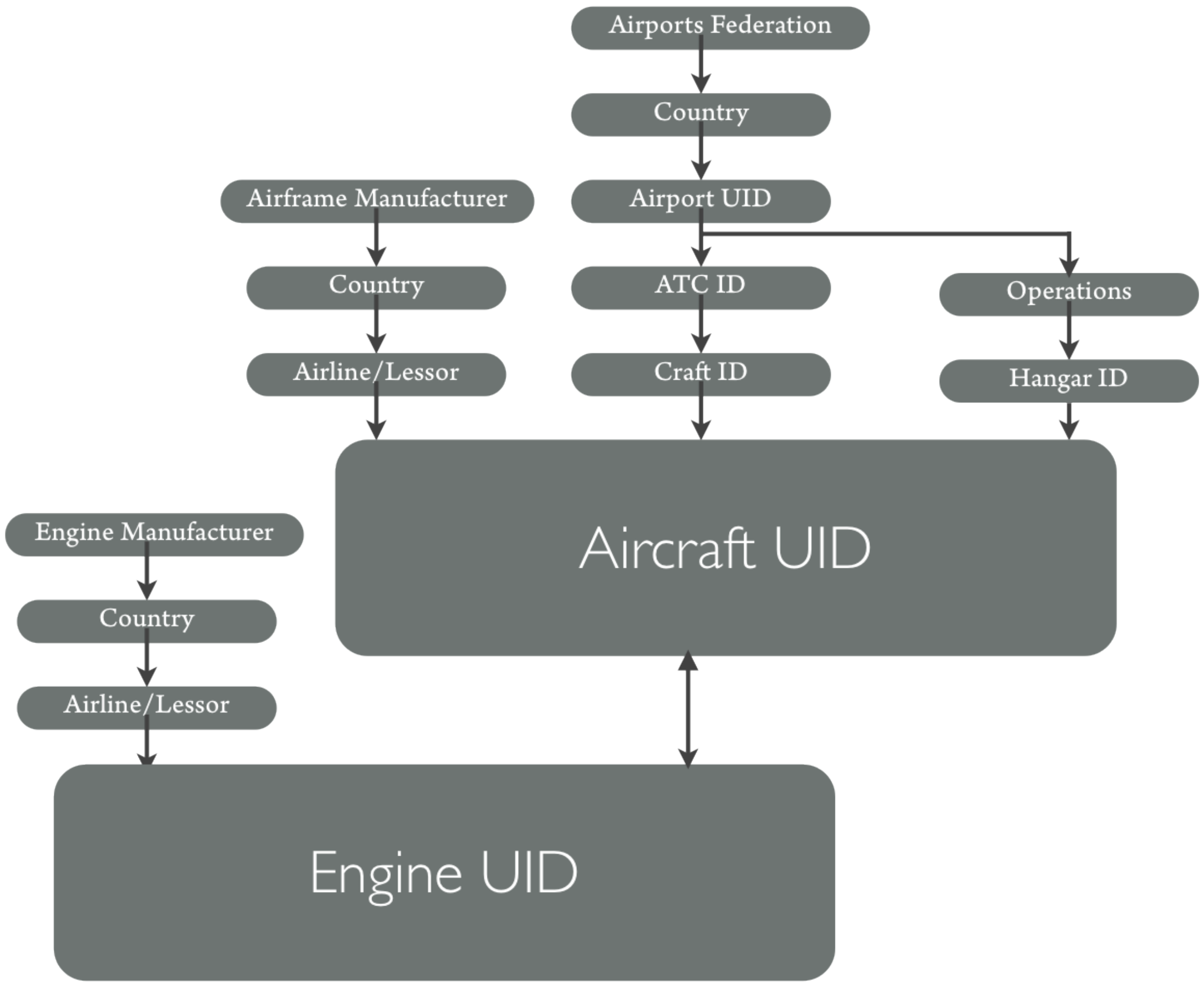


Figure 9.3: Depiction of the stakeholders to the data generated by an aircraft and its engines.

would require sharing data with the hangar where the maintenance is carried out. Likewise, another stakeholder is the air traffic control which would be keeping track of aircraft movements in the local airspace.

9.5.2 Multiple Identities of the Aircraft and the Engine

Imagine a fictitious aircraft lessor known as AeroBrit based in the United Kingdom. Let an aircraft of their's be identified by the id number *ab0948* and the URI adopted by the data unit of that aircraft will be `http://ab0948.aerobrit.wwi`, made from the the primary URI of the lessor is `http://aerobrit.wwi`. The aircraft manufacturer for example Boeing (`http://boeing.wwi`) would be able to identify the same aircraft with its URI issued based on the lessor's URI, i.e., `http://ab0948.aerobrit.unitedkingdom.boeing.wwi`.

Similarly, let the engine *rrxwb457* be manufactured by Rolls-Royce (`http://rolls-royce.wwi`), with the lessor of the engine being RRVentures (`http://rrventures.wwi`) based in United Kingdom and lessee being AeroBrit. The URI of the engine by the Lessor will then be `http://rrxwb457.rrventures.wwi` and meanwhile the engine manufacturer will be able to access the data from that engine by using the URI `http://rrxwb457.rrventures.unitedkingdom.rolls-royce.wwi`.

In this situation based on the agreement by airline with the engine lessor, some of the data from the aircraft would also need to be shared with the latter. While the aircraft lessor may not want to be identified or involved in that contract, they could rather have the aircraft DU adopt another identity of `http://aircraft.rrxwb457.rrventures.wwi`, where the aircraft is just another child DU of that engine. This would allow the engine lessor to access the aircraft data irrespective of the aircraft that it being used on, while at the same time could also give access to the engine manufacturer using the URI `http://aircraft.rrxwb457.rrventures.unitedkingdom.rolls-royce.wwi`. In addition the engine manufacturers can be restricted from a direct access to the AeroVentures DU and instead be allowed to use the URI `http://rrventures.unitedkingdom.rolls-royce.wwi/rrxwb457/aircraft/` to access the aircraft data. Likewise, the engine can adopt a generic URI for an engine at position A of the aircraft, `http://engineA.ab0948.aerobrit.unitedkingdom.boeing.wwi`. Such adoption of multiple identities provides privacy and thereby at the same time also allow the respective stakeholders to apply their own security policies.

9.5.3 Scenario 1: Aircraft Fuel Consumption and Emissions Data

This case study can now explore a possibility that some of the busy airports may want to optimise and orchestrate the landing order of aircrafts and subsequently even suggest a flight path to optimise an aircrafts arrival based on the their emissions to reduce the impact on the local environment. This means the aircraft would need to communicate its fuel consumption and emissions to the local air traffic control (ATC), since these are variables dependent on the weather conditions, altitude of flight as well as on the weight of the sum of cargo and passengers that an aircraft is carrying.

```
PREFIX heath1: <http://heath1.heathrow.unitedkingdom.airportsfederation.wwi
    />
SELECT DISTINCT * WHERE {
SERVICE heath1:aerobrit5409 {
  heath1:aerobrit5409 a  ?unknown:aircraft ;
    ?unknown:co2emmision   ?co2value ;
    ?unknown:fuelconsumption   ?fuelvalue .
} }
```

Listing 9.5: The example SPARQL query used by heath1 ATC to query the required data from an aircraft.

The ATC with this added responsibility now need to retrieve the individual aircraft parameters that would be required to optimise the landing order and the flight path of the aircrafts. Lets take an example ATC *Heath1* of Heathrow Airport (`http://heathrow.unitedkingdom.airportsfederation.wwi`), identified by its URI `http://heath1.heathrow.unitedkingdom.airportsfederation.wwi`. Each ATC can then identify the aircraft that has requested permission with a URI based on its own, e.g., the URI `http://aerobrit5409.heath1.heathrow.unitedkingdom.airportsfederation.wwi`. The ATC would then be able to query the required data from the respective DU's of the aircrafts by using a query as in Listing 9.5.

According the news report published on 16 May 2013 by The New York Times[5], the European Union was issuing fines on certain foreign carriers for not complying with a new norm that requires airlines to report their emissions when flying to any airport in the EU. The countries of the respective airlines have meanwhile asked those airlines not to share the data[6]. Such a situation can be detrimental to the attempt by individual ATCs since it does require the same data to optimise the landing order.

With WWI though, ATC can overcome this challenge by starting to issue random identities which then can be taken up by individual aircrafts every time they come to land. For example an aircraft previously identified by `http://heath1.heathrow.unitedkingdom.airportsfederation.wwi/aerobrit5409`, can instead be issued with a random id like `http://heath1.heathrow.unitedkingdom.airportsfederation.wwi/random4598`. This way the ATC can receive original and up-to-date data from individual aircrafts.

Meanwhile, the local authorities at the same time can be allowed to calculate the net emissions and net fuel consumption by using a domino query, like in the Listing 9.6, without knowing the actual identity of the individual aircrafts themselves. Such an ability can then be used to create a different strategy by local authorities to attain the real goal of reducing the net emissions like for example taxing the airports for their net emissions instead of individual airlines.

```
PREFIX heath1: <http://heath1.heathrow.unitedkingdom.airportsfederation.wwi
    />
SELECT DISTINCT (COUNT (?co2value) AS ?Netco2) (COUNT (?fuelvalue) AS ?
    Netfuel) WHERE {
DOMINO heath1:?aircrafts {
  heath1:?aircrafts a  ?unknown:aircraft ;
    ?unknown:co2emmision    ?co2value ;
    ?unknown:fuelconsumption    ?fuelvalue .
} }
```

[5]`http://www.nytimes.com/2013/05/17/business/global/17iht-emit17.html?pagewanted=all&_r=2&`

[6]`http://www.thehindubusinessline.com/industry-and-economy/logistics/india-remains-opposed-to-eus-emission-norms-ajit-singh/article4791709.ece`

Listing 9.6: The example SPARQL query used by local government authorities to calculate the net fuel consumption and emissions from original and up-to-date data by aircrafts landing in an airport in their jurisdiction by issuing a domino query.

9.5.4 Scenario 2: Correlating Airport Operations & Flight Performance

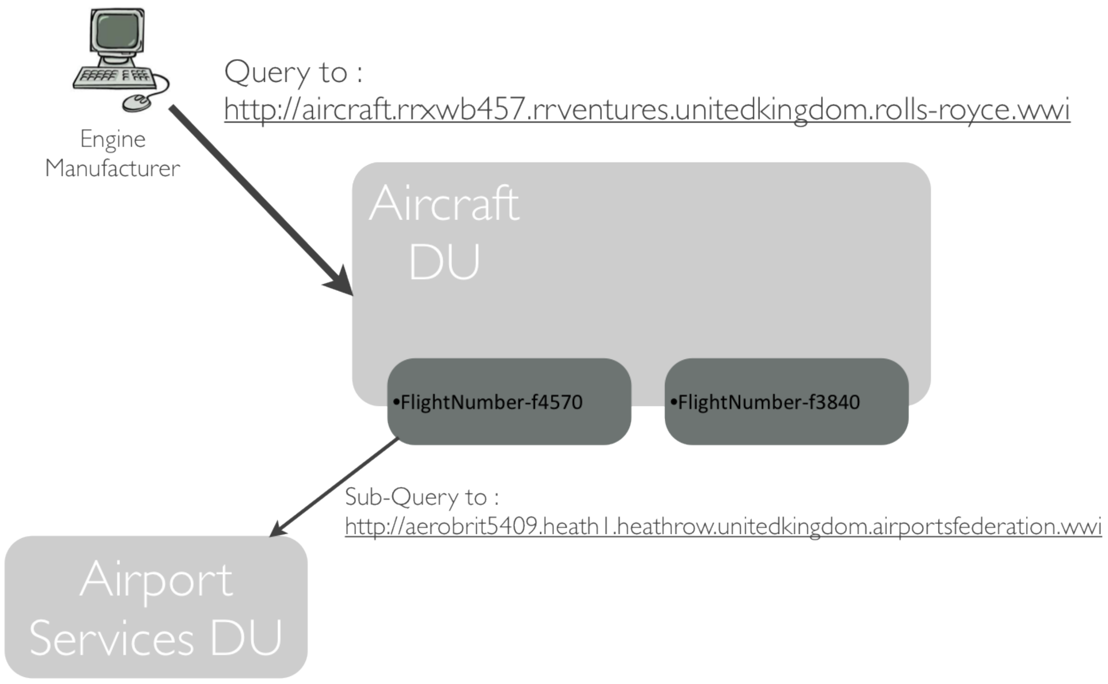

Engine Manufacturer
Query to : http://aircraft.rrxwb457.rrventures.unitedkingdom.rolls-royce.wwi
Aircraft DU
•FlightNumber-f4570
•FlightNumber-f3840
Sub-Query to : http://aerobrit5409.heath1.heathrow.unitedkingdom.airportsfederation.wwi
Airport Services DU

Figure 9.4: Depiction of an aircraft DU using its multiple identities to retrieve airport services data related to a particular flight and then sharing with the engine manufacturer to support engine diagnostics.

From a casual conversation with an engineer, the author was told about a situation where the engine manufacturer received reports of underperformance of an engine model by an airline where the same engine model operated by a different airline in the same region was not facing similar issues. Upon further and detailed investigation it was found that some airlines didn't face the problem from their precautionary measure of making the airport sweep the runway before every takeoff. That airport region being surrounded by sand, the ingestion of sand by the engine was found to be the problem. This situation is an example where airport operations and associated activities that help prepare an aircraft can have an affect on its performance and life cycle.

Since each aircraft with its id from the stakeholder airport would have access to the data on aircraft operations activity before each of its flight from an airport, the engine manufacturer can request the aircraft's DU under each of its engines (e.g., at `http://aircraft.rrxwb457.rrventures.unitedkingdom.rolls-royce.wwi`) for information on airport activities that occurred ahead of a particular take-off, e.g. flight number f4570 as depicted in Figure 9.4. This could help find patterns among similar engines that are in operation in similar areas. This is an example where use of multiple identities can help to compile broad data from multiple stakeholders for reuse without compromising the identity.

Likewise, an airport could also datamine for patterns in aircraft operations and local weather, emissions and other flight characteristics. With the anonymity that can be provided by WWI with the use of domino requests, the airports federation could have allowed access to original and up-to-date access to to airport operations data as well as aircraft performance data to the manufacturers or leasers, who can then datamine to find any common practices that help or hinder achieving the expected aircraft performance. This would be an useful capability to help prepare airports for better operational standards.

9.6 Summary

The objective of this chapter was to overcome the challenges from sparseness in location of data. The distributed nature of data storage required choices that needed to be made in the way a dataset is exposed which consequently affected how it can be reused.

In a corporate environment the primary requirement is an access to original and up-to-date data. Due to the requirement that the data be from trusted sources the universe of discourse could be limited in this situation and so a combination of federation and data warehousing was concluded to be the need in Section 9.1.1. For each of the warehouse a SPARQL endpoint was also recommended as the choice of data exposure technique, since it provides an ability to control data reuse.

The web of data that resides on a federated data warehouse SPARQL endpoint environment was though found to impose a challenge in terms of the time and computational resource required to find and process the required data. Section 9.2.1 discussed the possibility of querying strategies to overcome some of the challenges like Anapsid by Acosta et al. (2011).

From further discussion on specific security policy within Rolls-Royce, it was realised that no server is allowed to respond to any request or even acknowledge even its existence in any form, if the requester doesn't have the required access credentials. Along with the need for security by privacy, the creation and maintenance of an endpoints catalog would also be considered unacceptable. This added an additional barrier to the existing techniques, since all the current techniques does need to know the location of the endpoints to query it.

The unique way of embedding provenance structure that is used to create a WWI URI though opened new possibilities to finding data. The stakeholder hierarchal structure meant every DU was required to know only the location of its children DUs. Therefore, a query send to a particular DU can be propagated to child DUs, thus covering the entire WWI network. This method was named 'domino request' and was realised to be the solution to the unique challenge of finding data without maintaining an endpoints catalog. A demonstration of this technique in Section 9.4 though noted that the domino request can be time consuming compared to having all the data in the same repository. It was concluded that speeding it up would require further research and would be a matter of future work. Another need is to adapt the current DNS infrastructure to have an authentication mechanism to control its use based on users access credentials.

The noted benefits from domino request were the ability to retrieve original and up-to-date data while maintaining the privacy of the data creators. A case study on the airline industry further showed on how WWI can be applied to organise data in a world of internet of things. The ability of any thing to use multiple identities for every stakeholder showcased the benefit of being able to serve original data to all of them from the same source. Meanwhile it was also noted that with the domino request there is an additional benefit of collating data to find patterns and link between airport operations and aircraft performance without exposing the identity of any of the sources, which can be helpful to make better informed decisions.

Chapter 10

Conclusions & Future Work

"I think and think for months and years. Ninety-nine times, the conclusion is false. The hundredth time I am right."

– Albert Einstein

To draw some conclusions, it is first necessary to summarise the journey this research has undergone. This journey can be summed up into three stages that involved (a) finding the research question, (b) review of the concepts of linked data and semantic web to carry out a gap analysis of the requirements, and (c) formulation of a solution. At the end of each of these stages decisions were made that narrowed the research focus and so the following structure of conclusion is also aimed to reveal as much of its limitations as a consequence of the assumptions that have been made.

10.1 The Research Question: Prioritising the Problems

This research project was funded as a part of Strategic Investment in Low Carbon Emission Technologies (SILOET) by Rolls-Royce. The initial objective was to attempt to develop a quick costing method for a unit gas turbine, to support the company in the initial stages of a bidding process, from the wealth of data being generated while designing and manufacturing. The first task was to understand the process of information reuse, while at the same time the author was informed of a significant risk of being unable to collate the necessary data within the time scale of this research.

10.1.1 Summary: Chapters 2-4

In Chapter 2, a review of the existing literature was carried out to understand the various design reuse systems and the tools that have been used previously. Cost was then put into the perspective of the designer with the design spiral by Keane and Nair (2005). The list of the various ways in which cost can be classified as described by Tammineni (2007) were explored in Section 2.2 and concluded that it is important to define what the term cost meant to this research. Cost was then defined as the net monetary value of resources consumed to make a product or achieve a state.

Various cost estimation methodologies and tools were explored and presented in Figure 2.5 (Niazi et al., 2006) and Figure 2.6 (Scanlan et al., 2005), respectively. Due to the need to predict the unit cost as quick as possible, a parametric approach known as data mining was pinned down as the preferable choice. In Section 2.4 the concept of data mining was introduced, where data is processed to extract valuable correlations between various parameters and can then be used as a mathematical model to predict and infer an outcome dependent on those parameters. The general process of data mining was then described and the various challenges that are generally faced were noted in Section 2.4.2. The literature review in Chapter 2 concluded with the need to explore the importance of cost in a real life situation of a corporate environment like Rolls-Royce.

Chapter 3 reviews the bidding process of a civil gas turbine engine in Rolls-Royce and explores the relevance of 'the ability to quickly predict the unit cost of an engine'. Inspired from Kipling and Kliros (1993), a 'six W' representation of a cost engineer's mind was depicted to understand the importance of cost in the procedure of capturing customer requirements. This also pointed to the need to use a parametric approach (data mining) to predict the cost as quick as possible with a measurable level of confidence, in order to be able to reply quickly to a Request For Information (RFI).

Section 3.2 then looked into the challenges of reusing data for predicting cost and categorically classified them as issues of 'data accessibility' and 'data availability'. It was concluded that the primary focus needs to be to make data accessible, since the situation of not having enough (engine) data points would change with time considering more engines will be designed in the future and thus preparing the company for the future was deemed important. An insight into the state of data existence is shown in Figure 3.7, where the condition of data availability was noted as sparse with heterogenous mediums of storage. Taking the situation into consideration, a costing framework was suggested as illustrated in Figure 3.8 which involves extracting and compiling data from the existing documents and databases.

Consequently, Chapter 4 explored the ability to extract data from heterogeneous documentation formats. The aim was to be able to recall data that were stored as tables in the documents. The experiment concluded that the conditions of 100% recall precision and repeatability of extracting tabulated data can be achieved. Upon further discussion, it was realised that the lack of a data model when in tabulated form impairs the ability to extract just the required data point from a table. Moreover, if one were to use data from a document like in Figure 4.2 to choose a design, a mechanism to be able to be absolutely certain what every word means is also a necessity.

10.1.2 Conclusion on the Research Question

The discussion so far has shown the need for an ability to quickly predict unit cost of a gas turbine and how the process of data mining could be the choice. The noted lack of enough data points to datamine and inaccessibility of the required data meant the following tasks to be out of scope of this research within the stipulated time frame:

1. Understanding the effects on cost not just by design parameters, but also by the choice of manufacturing processes, material procurement and lifecycle of an engine.
2. Development and fine tuning of a mathematical model for unit cost prediction.

Moreover, from the understanding of the steps involved in reusing data in Section 3.4.2, it was shown that the process of data collation will have to be repeated to attain more suitable dataset if the question that one was attempting to answer is altered. In fact, the attempt to predict the unit cost is just an example of the vast amount of capabilities that can be developed with an ability to datamine (previously referred to as broad data reuse activities). This justifies that solving the challenge of data accessibility is the priority since it could also have a profound impact in other areas. Such an ability to make all data accessible in Rolls-Royce means to make all the data machine readable such that an engineer will then be able to easily collate the necessary data.

Meanwhile, in solving the data accessibility issues the heterogeneity and sparseness of data were the two key two challenges to be overcome. The realisation that the tabulated data extracted from heterogenous sources is not machine readable, meant all the tables still need to have had a data model applied such that just the required data can be recalled for reuse. Since applying the model would require an understanding of what data a table represents, the situation calls for the formulation of the strategy. A possible strategy suggested was to use the provenance information from the metadata of every document to find people of who are carrying out similar tasks to convert the data. However, this requires an environment where such modelled data can be stored and recalled from and is based on the provenance. Therefore, the scope of this research was narrowed and the research question was then concluded to be: "What is an appropriate framework for data reuse in an organisation to support a broad data reuse activity ?".

10.2 Gap Analysis: Finding the Bottlenecks

Consequently, the next step in solving the issue of data accessibility was to make those data machine readable. Due to availability of mature tools and methods to reuse linked data, this concept is further explored. The Chapters 6 and 7 that follows, aims to understand and list the specific set of challenges that would need to be overcome to enable creation and reuse of linked data in Rolls-Royce.

10.2.1 Summary: Chapters 5-7

In Chapter 5 the discussion on creation of linked data showed how the RDF data model is instrumental in making the data recognisable to a machine. This also opened up the opportunity to use the powerful SPARQL language designed to query RDF data. Section 5.4 explored the various ways to consume RDF from multiple sources like RDF files, databases and APIs. Further in Section 5.5 an example case study showed how the SPARQL query can be used to consume linked data in RDF format for the purpose of data mining. It was concluded that employment of linked data is a plausible solution to help recognise data that are available over the network. This left the want of a capability to collate the required data automatically, with the final challenge of creating 'an ability to find the required data' over the internet or intranet.

The need to be able to locate data over the internet inspired to explore the concept called Semantic Web. Berners-Lee et al. (2001) had proposed the concept of the Semantic Web as an extension of the current Web better enabling computers and people to work in cooperation. Hall and Shadbolt (2009) described the Semantic Web as the web of Linked Data. Sir Tim Berners-Lee listed the technologies and standards that could make the vision of semantic web possible through his architectures of semantic web (Figure 6.1). The understanding gained from the discussion in Section 6.3 made the author realise that the people adopt the technologies and standards listed in the semantic web architecture/stack depending on how they intended to apply them. It was also noted that how ever these technologies were applied, the vision of Semantic Web were the same. Therefore, semantic web instead of being a solution as hoped for, the author realised it is a vision of the state of the web.

Section 6.4 did a comparative case study, where providing the data was seen as a service. This opened up the new world where data gets created on the fly based on an input to a URI. Such data aren't indexed but need to be found to help participate in the linked data graph. This poised the need for Semantic web also to hold metadata of data to be able to discover non existent data. Further challenges in making linked data available to everyone were discussed and in Section 6.5 along with the need to have human comprehensible URI's, the derefereceability of a URI was deemed important.

In Chapter 7, the challenge of heterogeneity and sparseness of data were summed up with the need to mandate a data model along with the use of URIs as a necessary part of the solution. The situation was referred to as 'data disability'. The need therefore was understood to have the right infrastructure to support the growth and reuse of linked data as first noted in Section 6.6. An important requirement when creating new linked data is to reuse as many URIs as possible that already exist, this act helps to increase the interoperability among datasets since only then one would be able to recognise the vocabulary of URIs used to describe the data. Meanwhile, derefereceability of a URI was understood to be a useful way to help judge what thing or concept a URI represents and so is helpful in making the decision to reuse a URI. As a consequence the ownership rights to a URI is also necessary to maintain the data that gets served when dereferenced. In addition, in a corporate scenario like that of Rolls-Royce the data access restrictions that would be in existence can inhibit one from dereferencing a URI. The importance of dereferencing was further noted in Section 7.7 due to the part it can play it making the vision of pull by Siegel (2009) from semantic web a reality. Moreover, the absence of derefereceability could mean a reliance on semantic search engines to discover data. Besides the use of a crawler to index all the linked data is not necessarily a situation that would be allowed in a high security corporate environment considering the risk of data loss or theft by external agents.

10.2.2 Conclusion on the Challenges

The precondition of RDF as the data model is an assumption that has been clearly made. This is primarily from the fact that the subject, predicate, and object structure of representing data is the minimum information which is required to model it. The choice is further justified from the benefits of matured tools that are publicly available and the benefits seen from the query language of SPARQL. Nevertheless, the need to use unique references to identify the vocabulary used, the combination of use of URIs and RDF forms the widely researched and adopted concept of linked data.

The biggest challenge in adopting the concept of linked data was identified to be its creation. The adoption of linked data has so far been mainly for data that can be left in the public domain. The use of search engines are inevitable in this situation described as Linked Open Data (LOD) since there is not necessarily a structure to the sources of data. The realisation that this is not necessarily the situation in an organisation indicated that it should be possible to take benefit of the structured provenance information. The inability to assure dereferenceabilty of every URI to everyone for security reasons meant embedding provenance into the URI would be a necessary addition to help one identify and judge what a URI stands for. In addition an ability to see what URIs one's colleagues have used so far would also be another helpful guide to choosing a URI for reuse. The lack of any existing URIs to represent a thing or a concept would mean one has to create a new URI, in such situations URI discovery based on the provenance of the person who created it can be another useful capability. Therefore, it was concluded that in Linked Closed Data (i.e., linked data in a corporate scenario) requires a third dimension in the web infrastructure that is based on provenance of ownership.

This instilled the need to rethink what the Web is. Consequently a proposal to have a new web was suggested and it was called the World Wide Information Web (WWI). The objectives of WWI were concluded to be to support:

1. URI discovery and reuse through provenance
2. URI ownership
3. URI maintenance
4. Ability to pull data by dereferencing a URI
5. Ability to discover data through pull

10.3 Formulating a Solution: WWI

Having found the challenges that the WWI needs to overcome, the next step was to formally define a solution that also adheres to the specific information technology (IT) security requirements of an organisation.

10.3.1 Summary: Chapters 8-9

From the personal experience of the author in trying to collate data from employees of Rolls-Royce, a common sense for the need for security being prevalent was noticed. As a practice no one would share the data that they had, without confirming who the other person claimed to be. This method was called 'security by privacy' and adopting this method of protecting data by maintaing privacy in WWI was considered essential.

To enable 'security by privacy' and the need to incorporate the provenance into the URI as previously noted, the solution found was to create the URIs based on the stakeholder hierarchy of a person's position in an organisation. This inherently lets every stakeholder also assume ownership of the data to which it is a stakeholder. A WWI URI is based on the structure of a WWW URI with an additional requirement that the hierarchy in authority and path must reflect the stakeholder hierarchy. Moreover, since dereferencing any WWW URI requires participation from every part of the authority and path in sequence of its hierarchy, the existing DNS process can be evolved to have them authenticate the requesters access credentials before processing any request. This would help to ensure interoperability with the existing Web.

In addition the WWI considers two WWI URIs the same as long as the combined hierarchy represented by their authority and path parts are the same, irrespective of what proportion of the stakeholder hierarchy is represented in the 'authority' and 'path' of a URI as formalised in Section 8.4.3. This type of specification is possible in WWI because, to ensure security by privacy WWI adopts the internet of things concept where any person or thing that wishes to contribute to the linked data graph in an organisation would need to have had issued a WWI URI that identifies them. Meanwhile, in order to ensure data discovery by provenance the WWI further adopts the policy of 'data as a service' which requires the URI of every data contributor to lead to a data unit that can dereference itself as well as serve the data that it owns. These concepts were then formalised in Section 8.4 along with the principles of linked data.

The final test that remained to have WWI concluded as the research question was to see if WWI can also satisfy the needs of an organisation in reusing data. The IT security policy of Rolls-Royce presented a unique challenge, where by no server (i.e., data unit (DU) in WWI) is supposed to respond or even acknowledge its existence in any way if the one who is sending the query doesn't have the required access credentials. In addition, to maintain the security by privacy, the need was to avoid creation and maintenance of a catalog of the endpoints.

This created a unique challenge, since all the current methods of querying needs to know the location of the endpoint to query it. Moreover, to optimise a query an access to catalogue of endpoints is necessary. The need, therefore, was to have a mechanism whereby a endpoint could have had a choice to opt in to participate in a query. The solution was found by making use of the fact that every data unit identified by its URI would know the location of its child DU and thereby a new method of sending a request called as 'domino request' was formulated. Section 9.3.3 defined a 'domino request' as a process where by a DU that has received the request will process it and would then forward that 'domino request' to all the URIs to which it is a stakeholder, which are then expected to repeat the same. Thereby a reuser is able to treat the entire WWI of an organisation as a single graph. At the same time the structure of how a WWI URI is, also enables what was described in Section 9.3.2 as provenance driven querying.

The domino request on WWI was then demonstrated with an scenario of managing the UK police crime data and concluded that WWI with domino request is a solution to the research question. However, the demo also noticed that a domino request can be very slow depending on the total number of requests made and their response times.

10.3.2 Conclusion

The creation and growth of linked data in an organisation requires to have had the vocabulary of URIs reused to increase the interoperability among datasets. This called for an openness in way data is contained and emphasised on the importance being able to dereference any URI and as well see what URI others have used to represent a thing or concept. This was found to be in contradiction with the needs when seen from the perspective of reusing this data, which called for creating a more closed system with a requirement of privacy for security. The resolution in overcoming these conflicting interests was understood as to incorporate the provenance as stakeholder hierarchy into every URI, thus providing an inherent structure to the web of linked data.

Security by privacy as a data management style was embraced, the use of provenance meanwhile presented the issue that any reuse of data would then require the knowledge of provenance, which was contradictory to the IT security policy. This was overcome with the formulation of the 'domino request' on WWI, where the traditional requirement for a catalog of endpoints are no longer necessary. WWI therefore is a vision where privacy is at its core, while the incorporation of provenance on the other hand provides the organisation a scope for the much required accountability, hence an ability to audit the information usage.

An issue though was noted in terms of the query processing time required by a domino request, which is dependent on the number of number of requests required and their response times. Nevertheless, since WWI can enable automated collation of the required data from a vast organisation like that of Rolls-Royce, it therefore is indeed a better solution than not having it at all, since WWI with domino request can provide the necessary infrastructure to overcome the data disability to support broad data reuse activities. Newer techniques of processing a domino request is necessary and hence is an area that needs to be worked on and would require further attention in the future work. The capabilities of WWI was further showcased in the case study where it was shown how WWI can fit into the needs of the airline industry. The inherent use of internet of things concept within WWI along with ability to ensure privacy with domino requests, can enable the stakeholders to adopt newer techniques in managing independent resources using a more holistic approach. The case study also showcased that with the independence that 'internet of things' provides an ability to adopt multiple identities which in turn help to preserve privacy while still being able to provide access to original and up-to-date data.

10.4 Research Contribution

The major contributions of this research are:

1. Demonstrated how the usage of functions as linked data web services can help reuse them just like other linked data in a SPARQL query.

2. Demonstrated how corporate data can be converted to linked data by minting URIs embedded with stakeholder hierarchy as provenance to support the organic growth of linked data without the need for a central ontology. The incorporation of provenance into every URI was demonstrated to create inherent structure to linked data thus enabling provenance driven linked data creation and reuse.

3. Demonstrated how the concept of domino request can give a reuser an ability to discover data from various graphs connected by WWI web, as though all the data were located in the same triplestore.

4. Demonstrated that the domino request can help a data owner to maintain their privacy and yet have their data reused without being held accountable.

5. Demonstrated how WWI framework can be used to evolve a part of the existing Web into an internet of things built with a priority to ensure privacy that can help create linked closed data. This is unlike the semantic web stack upon which the linked open data (LOD) is based on where privacy is taken into account in the last step.

6. Demonstrated how WWI helps to create the paradigm of ‘locally distributed data and centrally distributed applications’ with respect to the data owner in an internet of things environment for privacy while being able to apply the respective access control policies from multiple stakeholders.

10.5 Future Work

Further work in the area of creating the WWI web includes the following

10.5.1 Making Deployable WWI DU

The first task that is necessary is to prepare a deployable WWI DU prototype and this would require adoption of a DNS protocol that supports access control. Adoption of communication protocols that are faster can be another addition to make communication among DU’s quicker. A notable change that WWI brings about is the way in which provenance driven querying can be done and this requires adopting new standards to SPARQL and implementing them to a triplestore. One of the key aims of the WWI was to support organic growth of linked data. Therefore, developing a metric to measure the interoperability of datasets that can be evaluated at every stakeholder level can be another useful guide to help identify new URIs that needs incorporation to an organisation’s formal ontology. The overall objective of the activities in this section is to identify the necessary tasks required to make a deployable DU to create a WWI web.

10.5.2 Formulation of Standards in Security Policies

As has been discussed in this document previously, security and access control is of prime importance. Therefore, understanding and formulation of a policy based security environment that allows role based security would an important requirement. These security policies need to prepare not just for DNS usage control but also for data reuse control and at the same time be prepared for an significant amount of communication requests. In addition, a distributed security access management system can be a useful addition for the internet of things to instil resilience for things working together autonomously.

10.5.3 Development of Appropriate Generic UI

As for any technology the ability to be inclusive of all user types is important. Linked data is one such area that needs support from a helpful user interface to find data, reuse URIs, and create data. An important requirement that would need to be achieved in this area would be to help a user incorporate functions written in multiple programming languages and data from multiple sources, since different people are more adept at different programming languages. Moreover, since reuse of linked data does involve dealing with data as RDF an appropriate generic UI would be a necessity.

10.5.4 Evolve WWI to Create a True Web of Services

For the need to enhance and bring to life the semantic web capabilities as envisioned previously, the WWI needs to be evolved to become a true semantic web of services. The need is to be able to incorporate models and data into the same web and being able to to combine multiple URIs (i.e., a compound URI), which can give an an output based on an input. Due to the inherent structure of the WWI URI, this can be carried out by adoption of a standard to combine URIs and would be a area that would require rigorous discussion before adoption.

Another notable area in data reuse that has not been covered in this thesis is temporal accuracy of data. Temporal accuracy can be significant, as the author understands that an RDF triple is a statement or a fact said by a particular person or entity. The validity of facts or statements can vary with time and do need regular maintainance. A solution that the author had in mind was to expand the storage model of triples in triplestore or quadstore with an extra column that would store the time at which a particular triple was created. By this method, when reusing data the reuser can try to measure the validity of an RDF triple based on the time at which it was created. This is another matter that requires further discussion since an extra column would mean a requirement for more computational resources.

10.5.5 Adoption of Payment Mechanism

As noted in Chapter 9, processing SPARQL queries can be expensive both in terms of computational resource and bandwidth consumption. Therefore, development of a payment model to compensate the data owners would be crucial to make the business model for data privacy work. In addition statistical techniques or data indices can be used to precompute and estimate the cost of a query before it is executed. The payment model can also be a useful tool for organisations to control and allocate the computational resources among various projects and employees.

10.5.6 Extending Application of WWI

Since the creation of a WWI web only requires deployment of DU, the WWI concept can be extended to create other networks. High trust networks like social networks can be transformed by WWI style identity and resource management for privacy. A notable benefit from localised management of a network could also mean better connectivity among people who live in areas with reduced network bandwidth to major datacenters. Likewise the application of WWI needs to be explored to identify areas which place prime importance on privacy along with access to original and up-to-date data.

10.6 Concluding Remarks

WWI with domino has aimed to strike a balance between privacy & accountability and open access (for interoperability) & closed access (for security) to create the linked data without having to compromise on any of the needs of the organisation. The next few challenges that WWI has to face can be summed up as to find a solution with the balance between the need for control on data against the performance needs of an organisation.

References

Abadi, D. J., Marcus, A., Madden, S. R., and Hollenbach, K. (2009). SW-Store: a vertically partitioned DBMS for Semantic Web data management. *The VLDB Journal*, 18(2):385–406.

Acosta, M., Vidal, M.-E., Lampo, T., Castillo, J., and Ruckhaus, E. (2011). ANAPSID: An Adaptive Query Processing Engine for SPARQL Endpoints. In *link.springer.com*, pages 18–34. Springer Berlin Heidelberg, Berlin, Heidelberg.

Altshuller, G. (1984). *CREATIVITY AS AN EXACT SCIENCE: The Theory of the Solution of Inventive Problems* . Gordon and Breach Science Publishers.

Anderson, J. R., Boyle, C. F., and Reiser, B. J. (1985). Intelligent Tutoring Systems. *Science*, 228(4698):456–462.

Anderson, J. R., Michalski, R. S., Michalski, R. S., Carbonell, J. G., and Mitchell, T. M. (1986). *Machine learning: an artificial intelligence approach*, volume 2. Morgan Kaufmann.

Angles, R. and Gutierrez, C. (2008). Survey of graph database models. *ACM Computing Surveys (CSUR)*, 40(1).

Antoniou, G. and vanHarmelen, F. (2004). *A Semantic Web Primer.* MIT Press.

Atre, M., Chaoji, V., Zaki, M. J., and Hendler, J. A. (2010). Matrix ”Bit” loaded: A Scalable Lightweight Join Query Processor for RDF Data. In *the 19th international conference*, page 41, New York, New York, USA. ACM Press.

Auer, S., Dietzold, S., Lehmann, J., Hellmann, S., and Aumueller, D. (2009). Triplify: light-weight linked data publication from relational databases. In *Proceedings of the 18th international conference on World wide web*, pages 621–630, New York, New York, USA. ACM Press.

Baader, A., Baessler, M., Doerre, J., Götz, T., Hamp-Bahnmüller, T., and Lang, A. (2006). Improving text search quality by exploiting organizational information.

Baget, J.-F. (2005). *RDF Entailment as a Graph Homomorphism*, volume 3729 of *4th International Semantic Web Conference, ISWC 2005, Galway, Ireland, November 6-10, 2005. Proceedings.* Springer Berlin Heidelberg.

Bergadano, F., Matwin, S., Michalski, R., and Zhang, J. (1992). Learning two-tiered descriptions of flexible concepts: The POSEIDON system. *Machine Learning*, 8(1):5–43.

Berners-Lee, T. (1998a). Hypertext Style: Cool URIs don't change. `http://www.w3.org/Provider/Style/URI`. Accessed April 18, 2013.

Berners-Lee, T. (1998b). The Scale-free nature of the Web. `http://www.w3.org/DesignIssues/Fractal.html`. Accessed April 18, 2013.

Berners-Lee, T. (2000). Semantic Web - XML2000. `http://www.w3.org/2000/Talks/1206-xml2k-tbl/slide10-0.html`. Accessed April 18, 2013.

Berners-Lee, T. (2003a). Standards, Semantics and Survival. `http://www.w3.org/2003/Talks/01-siia-tbl/`. Accessed April 18, 2013.

Berners-Lee, T. (2003b). The Semantic Web and Challenges. `http://www.w3.org/2003/Talks/01-sweb-tbl/Overview.html`. Accessed April 18, 2013.

Berners-Lee, T. (2003c). WWW Past & Future. `http://www.w3.org/2003/Talks/0922-rsoc-tbl/slide30-0.html`. Accessed April 15, 2013.

Berners-Lee, T. (2005a). Uniform Resource Identifier (URI): Generic Syntax. `http://tools.ietf.org/html/rfc3986`. Accessed April 15, 2013.

Berners-Lee, T. (2005b). Web for real people. `http://www.w3.org/2005/Talks/0511-keynote-tbl/`. Accessed April 15, 2013.

Berners-Lee, T. (2006). Artificial Intelligence and the Semantic Web. `http://www.w3.org/2006/Talks/0718-aaai-tbl/Overview.html`. Accessed April 18, 2013.

Berners-Lee, T. (2009). Linked Data - Design Issues. `http://www.w3.org/DesignIssues/LinkedData.htm`. Accessed April 18, 2013.

Berners-Lee, T., Chen, Y., Chilton, L., Connolly, D., Dhanaraj, R., Hollenbach, J., Lerer, A., and Sheets, D. (2006). Tabulator: Exploring and analyzing linked data on the semantic web. In *Proceedings of the 3rd International Semantic Web User Interaction Workshop*.

Berners-Lee, T., Hendler, J., and Lassila, O. (2001). The Semantic Web: Scientific American. `http://www.scientificamerican.com/article.cfm?id=the-semantic-web`. Accessed April 15, 2013.

Berners-Lee, T. and Kagal, L. (2008). The Fractal Nature of the Semantic Web. *AI Magazine*, 29(3):29.

Bizer, C. and Cyganiak, R. (2006). D2r server-publishing relational databases on the semantic web. In *Proceedings of the 5th International Semantic Web Conference (2006)*.

Bizer, C., Heath, T., and Berners-Lee, T. (2009). Linked Data - The Story So Far. *International Journal on Semantic Web and Information Systems*, 5(3):1–22.

Bolton, L. and Clegg, C. (2011). *Analysis of "As-Is" unit cost bidding process and associated organisational and social issues.* Rolls-Royce plc. Technical Report.

Braha, D. (2001). *Data mining for design and manufacturing: methods and applications.* Springer Netherlands.

Brickley, D. and Guha, R. V. (2004). RDF Vocabulary Description Language 1.0: RDF Schema .

Brinke, E. (2002). *Costing support and cost control in manufacturing: A cost estimation tool applied in the sheet metal domain.* PhD thesis, University of Twente.

Brooks, F. J. (2011). GE Gas Turbine Performance Characteristics. `http://www.muellerenvironmental.com/documents/GER3567H.pdf`. Accessed April 30, 2013.

Carroll, J. J., Bizer, C., Hayes, P., and Stickler, P. (2005). Proceedings of the 14th international conference on World Wide Web - WWW '05. In *the 14th international conference*, page 613, New York, New York, USA. ACM Press.

Chaudhuri, S. and Dayal, U. (1997). An overview of data warehousing and OLAP technology. *ACM SIGMOD Record*, 26(1):65–74.

Chein, M. and Mugnier, M. L. (2009). Graph-based knowledge representation: computational foundations of conceptual graphs.

Cobden, M., Black, J., Gibbins, N., Carr, L., and Shadbolt, N. (2011). A research agenda for linked closed data. In *Second International Workshop on Consuming Linked Data*, Bonn, Germany.

Colin, D. (2005). *Management Accounting for Buisness.* London Thomson Learning.

Cooper, R. and Kaplan, R. S. (1991). *The Design of Cost Management Systems: Text, Cases and Readings.* Prentice-Hall Inc., New York.

Crowder, R., Bracewell, R., Hughes, G., Kerr, M., Knott, D., Moss, M., Clegg, C., Hall, W., Wallace, K., and Waterson, P. (2003). A Future Vision For The Engineering Design Environment: A Future Sociotechnical Scenario. *Multiple values selected.*

Curan, R., Raghunathan, R., and Price, M. (2004). Review of aerospace engineering cost modelling: The genetic casual approach. *Progress in Aerospace Sciences.*

Daniel, J. J. and Marsh, R. (2010). *Data-mining in Rolls-Royce.* Rolls-Royce Plc. Technical report.

Dean, E. B. (1995). Design for Competitive advantage. *Cost Technology.*

Dontas, K. and Zemankova, M. (1990). APPLAUSE: An implementation of the Collins-Michalski theory of plausible reasoning. *Information Sciences*, 52(2):111–139.

Dubois, D., Prade, H., and Yager, R. R. (1993). *Readings in Fuzzy Sets and Intelligent Systems*. The Morgan Kaufmann Series in Representation & Reasoning. Morgan Kaufmann.

DuCharme, B. (2011). *Learning SPARQL*. Querying and Updating with SPARQL 1.1. O'Reilly.

Duffy, S. M., Duffy, A. H. B., and MacCallum, K. J. (1995). A Design Reuse Process Model. In *International Conference on Engineering Design*.

Erling, O. and Mikhailov, I. (2009). *RDF Support in the Virtuoso DBMS*, volume 221. Springer Berlin Heidelberg, Berlin, Heidelberg.

Falkenhainer, B. and Michalski, R. (1990). Integrating quantitative and qualitative discovery in the ABACUS system. *Machine learning: an artificial intelligence approach*, 3:153–190.

Fayyad, U., Piatesky-Shapiro, G., and Smyth, P. (1996). From Data Mining to Knowledge Discovery in Databases. *AI Magazine*, 17(3).

Feelders, A. (1996). Learning from biased data using mixture models. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, pages 102–107.

Fensel, D., Facca, F. M., Simperl, E., and Toma, I. (2011). *Semantic Web Services*. Springer Berlin / Heidelberg.

Gatnar, E. and Rozmus, D. (2005). Data Mining : The Polish Experience. In *Innovations in Classification, Data Science, and Information Systems*, pages 217–223. Springer.

Gearon, P., Passant, A., and Polleres, A. (2012). SPARQL 1.1 Update. `http://www.w3.org/TR/sparql11-update/`. Accessed April 15, 2013.

Gerber, A., Van der Merwe, A., and barnard, A. (2008). A Functional Semantic Web Architecture. In *The Semantic Web: Research and Applications*, pages 273–287. Springer Berlin / Heidelberg.

Grant Clark, K., Feigenbaum, L., and Torres, E. (2008). SPARQL Protocol for RDF. `http://www.w3.org/TR/rdf-sparql-protocol/`. Accessed April 15, 2013.

Grüber, T. R. (1993). A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2):199–220.

Guarino, N. (1998). Formal Ontology and Information Systems. In *Proceedings of FOIS'98*, Trento, Italy.

Guéret, C., Schlobach, S., de Boer, V., Bon, A., and Akkermans, H. (2011). ï£ijIs data sharing the privilege of a few? Bringing Linked Data to those without the Web. In *Extended Semantic Web Conference 2011*, pages 1–21, Heraklion.

Hall, W. and Shadbolt, N. (2009). The Semantic Web Revolution - Unleashing the World's most Valuable Information. In *Online Information Conference 2009*, Olympia Conference Centre, London.

Hartig, O. and Langegger, A. (2010). A Database Perspective on Consuming Linked Data on the Web. *Datenbank-Spektrum*, 10(2):57–66.

Hayes, J. and Gutierrez, C. (2004). *Bipartite Graphs as Intermediate Model for RDF.* Springer Berlin Heidelberg.

Heath, T. and Bizer, C. (2011). *Linked Data: Evolving the Web into a Global Data Space* . Synthesis Lectures on the Semantic Web: Theory and Technology. Morgan & Claypool, 1 edition.

Hendler, J. (2001). Agents and the Semantic Web. *IEEE Intelligent Systems*, 16(2):30–37.

Hendler, J. (2012). Increasing access to the web of "broad data". In *the International Cross-Disciplinary Conference*, pages 1–2, New York, New York, USA. ACM Press.

Hernandez, J. H. (2008). *Proximity Engineering in Rolls-Royce.* Rolls-Royce plc. Technical report.

Hu, Y. and Li, V. O. K. (2001). Satellite-based Internet: a tutorial. *Communications Magazine, IEEE*, 39(3):154–162.

Idreos, S., Kersten, M. L., and Manegold, S. (2009). Self-organizing tuple reconstruction in column-stores. In *the 35th SIGMOD international conference*, page 297, New York, New York, USA. ACM Press.

Isele, R., Umbrich, J., Bizer, C., and Harth, A. (2010). LDSpider: An open-source crawling framework for the Web of Linked Data . In *International Semantic Web Conference*, Shanghai.

Jacobs, I. and Walsh, N. (2004). Architecture of the World Wide Web, Volume One. `http://www.w3.org/TR/webarch/`. Accessed April 23, 2013.

Jindal, R. and Acharya, A. (2004). *Federated data warehouse architecture.* Wipro Technologies white paper. Technical report.

Kaufman, K. and Michalski, R. (1996). A method for reasoning with structured and continuous attributes in the INLEN-2 multistrategy knowledge discovery system. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, pages 232–237.

Keane, A. J. and Nair, P. B. (2005). *Computational Approaches for Aerospace Design: The Pursuit of Excellence.* John Wiley & Sons Ltd.

King, S. (2006). *Introduction to Data Mining.* Rolls Royce Power. Power Point Presentation.

Kipling, R. and Kliros, T. (1993). *The Elephant's Child and Other Just So Stories.* Dover Children's Thrift Classics. Dover Publications, Incorporated.

Kirchberg, M., Ko, R. K. L., and Lee, B. S. (2011). From Linked Data to Relevant Data – Time is the Essence. *arXiv.org.*

Klyne, G. and Carroll, J. J. (2004). Resource Description Framework (RDF): Concepts and Abstract Syntax. `http://www.w3.org/TR/rdf-concepts/`. Accessed April 15, 2013.

Klyne, G. and Matthews, B. (2004). SWAD-Europe: Deliverable 11.1: Framework for Security and Trust Standards. Technical Report IST-2001-34732.

Ladwig, G. and Tran, T. (2010). Linked data query processing strategies. In *Linked Data Query Processing Strategies*, pages 453–469. Springer Berlin Heidelberg.

Lakshminarayan, K., Harp, S., Goldman, R., and Samad, T. (1996). Imputation of missing data using machine learning techniques. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, pages 140–145.

Lanella, R. (2010). Semantic Web Architectures. `http://semanticidentity.com/Resources/Entries/2010/9/8_Semantic_Web_Architectures_(Whitepaper)_files/semweb-arch-wp-2010.pdf`. Accessed April 16, 2013.

Lavelli, A., Califf, M. E., Ciravegna, F., Freitag, D., Giuliano, C., Kushmerick, N., Romano, L., and Ireson, N. (2008). Evaluation of machine learning-based information extraction algorithms: criticisms and recommendations. *Language Resources and Evaluation*, 42(4):361–393.

Lee, S. and Siau, K. (2001). A review of data mining techniques. *Industrial Management* {*&* Data Systems, 101(1):41–46.

Li, Y. (2011). *Publishing Media Fragments and Annotations using Linked Data Principles.* PhD thesis, University of Southampton.

Loftus, G. (1994). Using confidence intervals in within-subject designs. *Psychonomic Bulletin & Review.*

Maloof, M. and Michalski, R. (1995). Learning evolving concepts using partial memory approach. In *Working Notes of the AAAI Fall Symposium on Active Learning*, pages 70–73. Citeseer.

Manola, F. and Miller, E. (2004). RDF Primer. `http://www.w3.org/TR/rdf-primer/`. Accessed April 15, 2013.

Michalski, R., Kaufman, K., and Wnek, J. (1991). The AQ family of learning programs: A review of recent developments and an exemplary application. *Reports of the Machine Learning and Inference Laboratory*, 1051:91–11.

Montoya, G., Vidal, M.-E., and Acosta, M. (2012). DEFENDER: a DEcomposer for quEries against feDERations of endpoints. In *Extended Semantic Web Conference 2012*.

Neumann, T. and Weikum, G. (2008). RDF-3X: a RISC-style engine for RDF. *Proceedings of the VLDB Endowment*, 1(1):647–659.

Neumann, T. and Weikum, G. (2009). Scalable join processing on very large RDF graphs. In *the 35th SIGMOD international conference*, page 627, New York, New York, USA. ACM Press.

Niazi, A., Dai, J., Balabani, S., and Seneviratne, L. (2006). Product cost estimation: Technique classification and methodology review. *Jounral of Manufacturing Science and Engineering*, 128:563.

Omitola, T., Zuo, L., Gutteridge, C., Millard, I., Glaser, H., Gibbins, N., and Shadbolt, N. (2011). Tracing the Provenance of Linked Data using voiD. In *The International Conference on Web Intelligence, Mining and Semantics*.

Ong, S. K. (2008). *Design reuse in product development modeling, analysis and optimization*. World Scientific Publishing, Hackensack NJ, world scientific publishing edition.

Oren, E., Delbru, R., Catasta, M., Cyganiak, R., Stenzhorn, H., and Tummarello, G. (2008). Sindice.com: a document-oriented lookup index for open linked data. *International Journal of Metadata, Semantics and Ontologies*, 3(1):37–52.

Parnas, D. L. (1994). *Software aging*. IEEE Computer Society Press, ieee computer society press edition.

Patni, H. and Henson, C. (2010). Linked sensor data. *International Symposium on Collaborative Technologies and Systems*.

Prud'hommeaux, E. and Seaborne, A. (2008). SPARQL Query Language for RDF. `http://www.w3.org/TR/rdf-sparql-query/`. Accessed April 15, 2013.

Putnam, L. H. (1978). A general empirical solution to macro software sizing and estimating problem. *Transactions on Software Engineering*, pages 345–361.

Quinlan, J. (1990). Probabilistic decision trees. *Machine learning: an artificial intelligence approach*, 3:140–152.

Rakhmawati, N. A. and Hausenblas, M. (2012). On the Impact of Data Distribution in Federated SPARQL Queries. In *Semantic Computing (ICSC), 2012 IEEE Sixth International Conference on*, pages 255–260.

Rezayat, M. (2000). Knowledge-based product development using XML and KCs. *Computer-Aided Design*, 32:299–309.

Ribeiro, J., Kaufman, K., and Kerschberg, L. (1995). Knowledge discovery from multiple databases. In *Proceedings of the First International Conference on Knowledge Discovery and Data Mining*, pages 240–245.

Rohloff, K., Dean, M., Emmons, I., Ryder, D., and Sumner, J. (2007). An Evaluation of Triple-Store Technologies for Large Data Stores . In Meersman, R., Tari, Z., and Herrero, P., editors, *Proceedings of the 2007 OTM Confederated international conference on On the move to meaningful internet systems*, pages 1105–1114, Berlin, Heidelberg. Springer Berlin Heidelberg.

Roy, R., Kelvesjo, S., Forsberg, S., and Rush, C. (2001). Quantitative and qualitative cost estimating for engineering design. *Journal of Engineering Design*, pages 147–162.

Rychener, M. (1988). *Expert systems for engineering design.* Academic Press, Inc.

Savransky, S. (2000). *Engineering of creativity: introduction to TRIZ methodology of inventive problem solving.* CRC Press.

Scanlan, J., Rao, A., Bru, C., Hale, P., and Marsh, R. (2006). Datum project: Cost estimating environment for support of aerospace design decision making. *Journal of Aircraft*, 43(4):1022–1028.

Scanlan, J. P., Rao, A. R., Bru, C., Hale, P., and Marsh, R. (2005). The DATUM project: a cost estimating environment for the support of aerospace design decision making. *Journal of Aircraft*, pages 1022–1028.

Schwarte, A., Haase, P., Hose, K., Schenkel, R., and Schmidt, M. (2011). *FedX: A Federation Layer for Distributed Query Processing on Linked Open Data.* Springer Berlin Heidelberg.

Sheth, A. P. and Larson, J. A. (1990). Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Computing Surveys*, 22(3):183–236.

Shuford, R. H. (1995). *Activity-based costing and traditional cost allocation structures, Cost estimators reference manual.* Johan Wiley & Sons Inc.

Siegel, D. (2009). *Pull: The Power of the Semantic Web to Transform Your Business.* Portfolio Hardcover, 1 edition.

Sivaloganathan, S. and Shahin, T. (1999). Design reuse: an overview. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, 213(7):641–654.

Slowinski, R. (1992). *Intelligent Decision Support: Handbook of Applications and Advances of the Rough Sets Theory.* Kluwer Academic Publishers.

Stewart, R. D. (1995). *Detailed Cost Estimating, in Cost Estimators Referene Manual.* Johan Wiley & Sons Inc.

Tammineni, S. V. (2007). *Designer driven cost modelling.* PhD thesis, School of Engineering Sciences, University of Southampton, Southampton.

Ullman, D. (1997). *The mechanical design process.* McGraw-Hill, New York.

Umbrich, J., Hausenblas, M., Hogan, A., Polleres, A., and Decker, S. (2010). Towards Dataset Dynamics: Change Frequency of Linked Open Data Sources. In *CEUR*.

Vidal, M.-E., Ruckhaus, E., Lampo, T., Martínez, A., Sierra, J., and Polleres, A. (2010). *Efficiently Joining Group Patterns in SPARQL Queries.* Springer Berlin Heidelberg.

Watson, I. (1999). Case-based reasoning is a methodology not a technology. *Knowledge-Based Systems*, 12(5-6):303–308.

Weiss, C., Karras, P., and Bernstein, A. (2008). Hexastore: sextuple indexing for semantic web data management. *Proceedings of the VLDB Endowment*, 1(1):1008–1019.

Widmer, G. and Kubat, M. (1996). Learning in the presence of concept drift and hidden contexts. *Machine Learning*, 23(1):69–101.

Wight, T. P. (1936). Factors affecting the cost of airplanes. *Journal of Aeronautical Sciences*, pages 122–128.

Wiseall, S. (2010). *SILOET work package 2.5 - Cost Modelling – level 2 plan.* Rolls-Royce plc. Technical report.

Younossi, O., Arena, M. V., Moore, R. M., Lorell, M., Mason, J., and Graser, J. C. (2002). *Military Jet Engine Acquisition: Technology basics and Cost-Estimating Methodology.* RAND.

Zadeh, L. (1965). Fuzzy Sets. *Information and Control*, pages 338–353.

Ziarko, W. P. (1994). *Rough Sets, Fuzzy Sets and Knowledge Discovery.* Springer-Verlag.

Appendix A

Python Codes: Relocation Experiment

A.1 spiderzoopla.py

The following code runs a service where the Zoopla website is queried for houses available to rent for a specified location in the United Kingdom.

```

def phantomqueryin(search_city, price_max, price_min):
    phantomjsquery = '''


        var page = new WebPage();

        page.onConsoleMessage = function(msg) {
        console.log(msg);
        };

        page.open(encodeURI("http://www.zoopla.co.uk/for-sale/houses/%(a)s
    /?beds_min=3&price_max=%(max_price)s&price_min=%(min_price)s&
    property_type=houses&q=%(b)s&search_source=home&page_size=50&pn=1"),
    function (status) {
        if (status !== "success") {
        console.log("Unable to access network");
        } else {
        page.evaluate(function() {
        var list = document.querySelectorAll('a.listing-results-address'),
    address = [], i;//span.status //twitter-atreply
        var listroom = document.querySelectorAll("h2.listing-results-attr
    "),room = [], i;
        var listprice = document.querySelectorAll("a.listing-results-price
    "), price = [], i;

```

```
        console.log('{');
        console.log('  "head": {  ');
        console.log('    "vars": [ "address" , "description" ,"listprice"
    ]');
        console.log('  } ,');
        console.log('  "results": {  ');
        console.log('    "bindings": [  ');

        for (var i = 0; i < list.length; ++i) {
        console.log("      {")
        console.log('        "address" : { "type":"string", "value" : " ' +
     list[i].innerHTML.replace(/<.*?>/g, '')+'" } ,' );
        describer = listroom[i].innerHTML.replace(/<.*?>/g, '');
        describer = describer.replace(/\\n/g,'');
        console.log('        "description" :{ "type":"string", "value" : "'
     + describer+ '" } ,' );
        pricer = listprice[i].innerHTML.replace(/<.*?>/g, '');
        pricer = pricer.replace("/ /g", "");
        pricer = pricer.replace('/[^0-9]+/', '');
        pricer= pricer.replace(/\D/g,'');
        pricer = pricer.replace("/,/g", "");
        console.log('        "listprice" : { "type":"literal", "value" : "'
     + pricer + '" }');

        if ( i < ((list.length)-1) ) {console.log("      },")} else {
    console.log("      }")};
        }
        console.log("    ]")
        console.log("  }")
        console.log("}")
        });
        }
        phantom.exit();
        });


        '''%{"a":search_city, "b":search_city, "max_price":price_max, "
    min_price": price_min }
    return phantomjsquery

##
phantomjsquery1 = '''


    var page = new WebPage();

    page.onConsoleMessage = function(msg) {
    console.log(msg);
    };

    page.open(encodeURI("http://www.zoopla.co.uk/for-sale/houses/london/?beds_min=3&price_max=1250000&price_min=10000&property_type=houses&q=London&search_source=home&page_size=50&pn=1"), function (status) {
    if (status !== "success") {
    console.log("Unable to access network");
    } else {
    page.evaluate(function() {
    var list = document.querySelectorAll('a.listing-results-address'), address = [], i;//span.status //twitter-atreply
    var listroom = document.querySelectorAll("h2.listing-results-attr "), room  = [], i;
    var listprice = document.querySelectorAll("a.listing-results-price"), price  = [], i;
    for (var i = 0; i < list.length; ++i) {
    console.log("{")
    console.log('address:' + list[i].innerHTML.replace(/<.*?>/g, '') );
    console.log('description:' + listroom[i].innerHTML.replace(/<.*?>/g, ''));
    console.log('listprice:' + listprice[i].innerHTML.replace(/<.*?>/g, ''));

    }
    });
    }
    phantom.exit();
    });



    '''
def crawlzoopla(search_city, price_max, price_min, sanjay = "/Users/Home/Desktop/Phantomjsfiles/pythonjs/tempo.txt" ):
    import os, json
    ##Creating the js file.
    filename = """/Users/Home/Desktop/Phantomjsfiles/pythonjs/temp2.js"""
    fileoutput = """/Users/Home/Desktop/Phantomjsfiles/pythonjs/temp_output.txt"""
    def removetempfiles():
        if os.path.exists(filename):
            os.remove(filename)
        if os.path.exists(fileoutput):
            os.remove(fileoutput)

    removetempfiles()
    phantomquery = phantomqueryin(search_city, price_max, price_min)
    f = open(filename,"w+")
    f.write(phantomquery)
    f.close()
    ## Running the js file with phantomjs.
    output = os.system('/opt/local/bin/phantomjs %(a)s > %(b)s ' %{"a":filename, "b":fileoutput})
    ##Processing the json data and Removing the pound sign.
    #sanjay = "/Users/Home/Desktop/Phantomjsfiles/pythonjs/tempo.txt"
    if os.path.exists(fileoutput):
        #print "yes"
        f = open(fileoutput,"r")
        f2 = open(sanjay,"w")
        content=[]
        while(True):
            read_fileoutput = f.readline()
            #print read_fileoutput
            len_readline=len(read_fileoutput)
            content.append(read_fileoutput)
            if (len_readline==0):
                break
        for every in content:
            #print every
            #f2.write(every.replace(rpl,''))
            str=''
            for j in range(0,len(every)):
                #every1=every.replace('r',rpl)
                if(every[j]=='\xc2'):
                    pass
                    #print 'hola'
                elif (every[j]=='\xa3'):
                    pass
                    #print 'hola again'
                else:
                    str=str+every[j]

            every1=str
            f2.write(every1)
        f2.close()
        f.close()
        removetempfiles()
        print "\n\n Finished Spider has crawled zoopla succesfully."
        return sanjay
```

A.2 skeleton.py

The following python code is a set of functions that have been used to support queries.

```
import urllib
import httplib2
import json
import time
print "Running skeleton.py"

def joseki(sparqlendpoint,squery,queryoutput):


    query1 = '''
        PREFIX afn: <http://jena.hpl.hp.com/ARQ/function#>
        PREFIX rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#>
        PREFIX coo:<http://purl.org/coo/ns#>


        SELECT DISTINCT '''
    for qout in queryoutput.split(","):
        query1 = query1+" "+qout

    query2 = '''
        FROM <%(a)s>
        WHERE {
        SERVICE <%(b)s>
         '''%{"a":sparqlendpoint, "b":sparqlendpoint}

    query2 = '''
        WHERE {
        ?sn ?pn ?on
        SERVICE <%(b)s>
         '''%{"b":sparqlendpoint}

    query2 = '''
        WHERE {
        <http://example.org/book/book1> ?p ?o.
        SERVICE <%(b)s>
         '''%{"b":sparqlendpoint}
    query2 = '''
        WHERE {
        SERVICE <%(b)s>
         '''%{"b":sparqlendpoint}
    query2 = '''
        WHERE {
        <http://example.org/book/book1> ?p ?o.
        SERVICE <%(b)s>
         '''%{"b":sparqlendpoint}
    query2 = '''
        WHERE {
        SERVICE <%(b)s>
         '''%{"b":sparqlendpoint}

    query = query1 +query2+squery+ '''
        }LIMIT 10'''
    print query

    repository = 'sparql'
    repository = 'books'
    endpoint = "http://localhost:2020/%s" % (repository)

    print "POSTing SPARQL query to %s" % (endpoint)
    params = { 'query': query }
    headers = {
        'content-type': 'application/x-www-form-urlencoded',
        'accept': 'application/sparql-results+json'
    }


    (response, content) = httplib2.Http().request(endpoint, 'POST', urllib.
    urlencode(params), headers=headers)
    print "Response %s \n" % response.status
    #print content
    results = json.loads(content)
    return results

def joseki2(squery):
    repository = 'sparql'
    repository = 'books'
    endpoint = "http://localhost:2020/%s" % (repository)

    print "POSTing SPARQL query to %s" % (endpoint)
    params = { 'query': squery }
    headers = {
        'content-type': 'application/x-www-form-urlencoded',
        'accept': 'application/sparql-results+json'
    }
    (response, content) = httplib2.Http().request(endpoint, 'POST', urllib.
    urlencode(params), headers=headers)
    print "Response %s \n" % response.status
    #print content
    results = json.loads(content)
    return results


def openrdf(sparqlendpoint,squery,queryoutput):
    #openrdf is sesame rdf store.
    query1 = '''
        PREFIX afn: <http://jena.hpl.hp.com/ARQ/function#>
        PREFIX rdfs:<http://www.w3.org/2000/01/rdf-schema#>
        PREFIX foaf:<http://xmlns.com/foaf/0.1/>
        PREFIX owl:<http://www.w3.org/2002/07/owl#>
        PREFIX xsd:<http://www.w3.org/2001/XMLSchema#>
        PREFIX rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#>
        PREFIX coo:<http://purl.org/coo/ns#>
        PREFIX vso:<http://purl.org/vso/ns#>
        PREFIX gr:<http://purl.org/goodrelations/v1#>
        PREFIX vvo:<http://purl.org/vvo/ns#>

        SELECT DISTINCT '''
    for qout in queryoutput.split(","):
        query1 = query1+" "+qout

    query2 = """
        WHERE
        """
    query = query1 +query2+squery
    print query
    endpoint = sparqlendpoint


    print "POSTing SPARQL query to %s" % (endpoint)
    params = { 'query': query }
    headers = {
        'content-type': 'application/x-www-form-urlencoded',
        'accept': 'application/sparql-results+json'
    }

    (response, content) = httplib2.Http().request(endpoint, 'POST', urllib.
    urlencode(params), headers=headers)
    print "Response %s \n" % response.status

    results = json.loads(content)
    return results

def kasabi(sparqlendpoint,squery,queryoutput):

    endpoint="http://api.kasabi.com/dataset/cars/apis/sparql"



    print "POSTing query to %s" % (endpoint)

    params = { 'apikey':'9b2ca2177b9daa1522183d51b09d39b443b2f252','output'
    :'json','query' : squery
            }
    para = urllib.urlencode(params)
    headers = {
        'X_KASABI_APIKEY': '9b2ca2177b9daa1522183d51b09d39b443b2f252',
        'accept': 'application/json'
    }

    (response, content) = httplib2.Http().request(endpoint+"?"+para, 'GET')
    print "Response %s \n" % response.status
    print content
    filename = """/Users/Home/Desktop/Phantomjsfiles/pythonjs/out.json"""
    f = open(filename,"w+")
    f.write(content)
    f.close()
    results = json.loads(content)
    return results

def google_geocode(address='1600 Amphitheatre Parkway, Mountain+View, CA',
    geo_outfile='/Users/Home/Desktop/Phantomjsfiles/pythonjs/geo_outfile'):
```

```
    ##In order to avoid ip block by google, a delay of 0.7 seconds is
    applied before every query.
    time.sleep(0.7)
    endpoint= "http://maps.googleapis.com/maps/api/geocode/json"
    params = {
    'address':address,
    'sensor' : 'false'
    }
    para = urllib.urlencode(params)
    (response, content) = httplib2.Http().request(endpoint+"?"+para, 'GET')
    print "Response %s \n" % response.status
    #print content
    results = json.loads(content)
    print results ['status']
    return results

def google_distancematrix(start, end, mode='driving',units='metric'):
    endpoint = 'http://maps.googleapis.com/maps/api/distancematrix/json'
    start = start.replace(' ','+')
    end = end.replace(' ','+')
    params = {
    'origins':start,
    'destinations' : end,
    'mode':mode,
    'units':units,
    'sensor':'false'
    }
    para = urllib.urlencode(params)
    (response, content) = httplib2.Http().request(endpoint+'?'+para, 'GET')
    #print "Response %s \n" % response.status
    #print content
    results = json.loads(content)
    #print results ['status']
    return results
```

A.3 cobra.py

The following is the python code used to find a house to rent in Southampton from where the commuting distance to places SO17 3RQ and SO18 2NU identified by their post-codes is the least.

```
import skeleton
import spiderzoopla, json, time
## The list of functions.
def gmap_addressinput(address_in):
    #Caculating
    gmap_out = skeleton.google_geocode(address_in)# result [address_in]['
    value'])
    return gmap_out

def netdist(a,b,c,d):
    #Calculating the net distance value.
    net_dist = int(a['rows'][0]['elements'][0]['distance']['value'])+int(b['rows'][0]['elements'][0]['distance']['value'])+int(c['rows'][0]['elements'][0]['distance']['value'])+int(d ['rows'][0]['elements'][0]['distance']['value'])
    print net_dist
    return net_dist

###Search inputs###
search_in_city = 'southampton'
price_max= '2100000'
price_min = '10000'
workaddress = 'Southampton univesity, SO17 1BJ'
Schooladdress = 'SO18 2NU'
## Finding the properties in zoopla.
out = spiderzoopla.crawlzoopla(search_in_city, price_max, price_min)
cleanfile = open(out,'r')
cleanjson = cleanfile.read()
results = json.loads(cleanjson)
cleanfile.close()
## Execution of commands.
addressstore= []
i=0 #Setting the inital value of the parameter to zero.
count = 0 #Setting the inital value of the parameter to zero.
shortestdist=1000000000000 #Setting a initial value for the shortest distance value.
for result in (results ['results']['bindings']): #Reading data from the json output.
    if count<4:
        pass
    else:
        sta = gmap_addressinput((result ['address']['value']))
        if sta['status'] == 'OK': #Checking if the data recieved is OK.
            to_work = skeleton.google_distancematrix((result ['address']['value']),workaddress)
            if (to_work['status'])== 'OK':#Checking if the data recieved is OK.
                from_work = skeleton.google_distancematrix(workaddress,(result ['address']['value']))
                if (from_work['status'])== 'OK':#Checking if the data recieved is OK.
                    to_school = skeleton.google_distancematrix((result ['address']['value']),Schooladdress)
                    if (to_school['status'])== 'OK':#Checking if the data recieved is OK.
                        from_school = skeleton.google_distancematrix(Schooladdress,(result ['address']['value']))
                    if (from_school['status'])== 'OK':#Checking if the
    data recieved is OK.
                        netdistance = netdist(to_work,from_work,
    to_school,from_school)#calculating the net distance.
                        if netdistance<shortestdist:
                            print 'New shortdistance updated!'
                            shortestdist= netdistance
                            shortestplace = result ['address']['value']
        else:
            print "couldn't calculate distance."
    count= count+1
    i=i+1
print "----------Final answer----------"
print shortestplace
print shortestdist
```

A.4 httpserver.py

This code acts as a proxy to use the web scraping service to find houses to rent from the Zoopla site as a linked data service from a SPARQL query.

```
import time
import BaseHTTPServer
import urllib
from fyzz import parse
import spiderzoopla
import json


HOST_NAME = 'localhost' # !!!REMEMBER TO CHANGE THIS!!!
PORT_NUMBER = 55000 # Maybe set this to 9000.


def headput(varkeys):
    h_1 = """
        <sparql xmlns="http://www.w3.org/2005/sparql-results#">
        <head>
         """
    h_end = """</head>"""
    for every in varkeys:
        h_1 = h_1+'''<variable name="'''+every+'''"/>'''
    h_out = h_1+h_end
    return h_out

def sparqlworthyaddress(address_val):
    ##Assumtion: All the '_' means its a ',' ==> No more valid!
    #Note: This could potentially be solved by issuing every address with a
     token which then gets use in the processing. Or some way for triple
    stores to accept these characters.
    #newadd = address_val.replace(',','_')
    newadd = address_val.replace(' ','')
    return newadd

def tailput(varkeys_tail):
    tails = tails+'''
    <result>
    <binding name="'''+varkeys[0]+'''">
    <uri>'''+(varkeys_tail ['address']['value'])+'''</uri>
    </binding>
    </result>
    '''


def zooplastoretokasabi(result):
    ###Kasabi inputs: dataset and namespaces###
    dataset = 'mytest'
    ns = 'http://data.kasabi.com/dataset/'+dataset
    ns_is = ns+'/is'
    ns_value = ns+'/value'
    ns_search_city=ns+'/searchcity'
    ns_max_price = ns+'/'+search_in_city+'/maxprice'
    ns_min_price = ns+'/'+search_in_city+'/minprice'
    ns_sale = ns+'/forsale'
    ns_description = ns+'/description'
    ns_listprice = ns+'/listprice'
    ns_locatedin = ns+'/locatedin'
    #cityname/address in searchcity
    kapi.triple_createandupload(dataset,ns+'/'+search_in_city+'/'+
    sparqlworthyaddress(result ['address']['value']),'uri',ns_locatedin,'
    uri',ns+'/'+search_in_city, 'uri')
    #cityname/addressvalue description resultvalue
    kapi.triple_createandupload(dataset,ns+'/'+search_in_city+'/'+
    sparqlworthyaddress(result ['address']['value']),'uri',ns_description,'
    uri',(result ['description']['value']),'value')
    #cityname/addressvalue listpice resultvalue
    kapi.triple_createandupload(dataset,ns+'/'+search_in_city+'/'+
    sparqlworthyaddress(result ['address']['value']),'uri',ns_listprice,'
    uri',(result ['listprice']['value']),'value')

def queryprocess(s):
    paths = urllib.unquote(s.path)
    paths = paths.replace('+',' ')
    #paths = paths.replace('\n',' ')
    unused, queryin = paths.split('?query=')
    print '\nquery:\n', queryin
    query_parsed = parse(queryin)
    qp_selected = query_parsed.selected
    vars = query_parsed.variables
    varkeys = vars.keys()
    heading = headput(varkeys)
```

```python
    qp_where = query_parsed.where
    qp_where_individuals = qp_where[0]
    search_in_city = str(qp_where_individuals[0])
    search_in_city = search_in_city.replace("SparqlLiteral('",'')
    search_in_city = search_in_city.replace("')",'')
    if qp_where_individuals[1] == '''<http://www.vocab.org/zoopla/to_rent>''' :
        search_location = qp_where_individuals[0]
        ## Finding the properties in zoopla.
        out = spiderzoopla.crawlzoopla(search_in_city, price_max='2100000', price_min='10000') #out is the location of output.
        cleanfile = open(out,'r')
        cleanjson = cleanfile.read()
        results = json.loads(cleanjson)
        cleanfile.close()
        #print results
        # Execution of acquisition and storage commands.
        tails = """
            <results>
            """
        uploadtokasabi = True
        addressstore= []
        i=0
        count = 0
        shortestdist=1000000000000
        for result in (results['results']['bindings']):
            if count<4:
                pass
            else:
                #zooplastoretokasabi(result)
                res = (result ['address']['value'])
                res = res.replace(',','')
                res = res.replace('.','')
                tails = tails+'''
                    <result>
                    <binding name="'''+varkeys[0]+ '''">
                    <literal>'''+res+'''</literal>
                    </binding>
                    </result>
                    '''
                    #tailput(result)

            count = count+1
    tails = tails + """</results> </sparql> """
    resultant = heading+tails
    print "query processed"
    return resultant


class MyHandler(BaseHTTPServer.BaseHTTPRequestHandler):
    def do_HEAD(s):
        s.send_response(200)
        s.send_header("Content-type", "application/sparql-results+xml")
        s.end_headers()

    def do_POST(s):
        abc = queryprocess(s)
        """Respond to a POST request."""
        print "It was a POST request"
        print s
        if not abc is None:
            length = len(abc)
        else:
            length = 0
        statusCode=200
        s.send_response(statusCode)
        mimeType = 'application/sparql-results+xml'
        s.send_header('Content-Type', mimeType)
        s.send_header('Content-Length', length)
        s.end_headers()
        print abc
        s.wfile.write(abc)


    def do_GET(s):
        abc = queryprocess(s)
        """Respond to a GET request."""
        print "it was a get request"
        print s
        if not abc is None:
            length = len(abc)
        else:
            length = 0
        statusCode=200
        s.send_response(statusCode)
        mimeType = 'application/sparql-results+xml'
        s.send_header('Content-Type', mimeType)
        s.send_header('Content-Length', length)
        s.end_headers()
        s.wfile.write(abc)


        #s.wfile.write("<html><head><title>Title goes here.</title></head
    >")
        #s.wfile.write("<body><p>This is a test.</p>")
        # If someone went to "http://something.somewhere.net/foo/bar/",
        # then s.path equals "/foo/bar/".
        #s.wfile.write("<p>You accessed path: %s</p>" % s.path)
        #s.wfile.write("</body></html>")


if __name__ == '__main__':
    server_class = BaseHTTPServer.HTTPServer
    httpd = server_class((HOST_NAME, PORT_NUMBER), MyHandler)
    print time.asctime(), "Zoopla to-rent Server Starts - %s:%s" % (
HOST_NAME, PORT_NUMBER)
    try:
        httpd.serve_forever()
    except KeyboardInterrupt:
        pass
    httpd.server_close()
    print time.asctime(), "Server Stops - %s:%s" % (HOST_NAME, PORT_NUMBER)





''' Fyzz tree structure:

    self.type = None
    self.prefixes = {}
    self.selected = []
    self.variables = {}
    self.where = []
    self.distinct = False
    self.reduced = False
    self.orderby = []
    self.offset = None
    self.limit = None

    '''


'''
abc = """
    <sparql xmlns="http://www.w3.org/2005/sparql-results#">
    <head>
    <variable name="s"/>
    </head>
    <results>
    <result>
    <binding name="s">
    <uri>http://www.w3.org/2002/07/owl#Thing</uri>
    </binding>
    </result>
    </results>
    </sparql> """

def output_head_builder(tree):
    #function not used
    #heads = " {u'head': {u'vars': [ "
    heads = """{ "head": { "vars": [ """
    if tree.selected[0] == '*':
        vars = tree.variables
        varkeys = vars.keys()
        print varkeys
        counting = len(varkeys)
        for every in varkeys :
            counting = counting - 1
            print every
            if counting == 0:
                heads = heads +'"'+ str(every)+'"'
            else :
                heads = heads +'"' + str(every)+'"'+','

    else :
        varkeys = []
        counting = 0
        for every in tree.selected :
            #Cleaning the variables from the junk around it.
            tvars= tree.selected[counting]
            tvars = tvars.replace("SparqlVar(",'')
            tvars = tvars.replace(")",'')
            tvars = tvars.replace("'",'')
            varkeys[counting]= tvars
            counting = counting + 1
        counting = len (varkeys)
        for every in varkeys :
            counting = counting - 1
            if counting == 0:
                heads = heads + varkeys[counting]
            else :
                heads = heads + varkeys[counting]+','
    print "heads output : ", heads
    return heads
'''
```

A.5 googleapiserver.py

This code acts as a proxy to use the google api for calculating global distance and travel times between two points on the map.

```
import time
import BaseHTTPServer
import urllib
from fyzz import parse
import spiderzoopla
import json
import skeleton

HOST_NAME = 'localhost' # !!!REMEMBER TO CHANGE THIS!!!
```

```
PORT_NUMBER = 54000 # Maybe set this to 9000.

def headput(varkeys):
    h_1 = """
        <sparql xmlns="http://www.w3.org/2005/sparql-results#">
        <head>
        """
    h_end = """</head>"""
    for every in varkeys:
        h_1 = h_1+'''<variable name="'''+every+'''"/>'''
    h_out = h_1+h_end
    return h_out

def queryprocess(s):
    paths = urllib.unquote(s.path)
    paths = paths.replace('+',' ')
    #paths = paths.replace('\n',' ')
    unused, queryin = paths.split('?query=')
    print '\nquery:\n', queryin
    query_parsed = parse(queryin)
    qp_selected = query_parsed.selected
    vars = query_parsed.variables
    varkeys = vars.keys()
    heading = headput(varkeys)
    qp_where = query_parsed.where
    qp_where_individuals = qp_where[0]
    from_location = str(qp_where_individuals[0])
    from_location = from_location.replace("SparqlLiteral('",'')
    from_location = from_location.replace("')",'')
    googleloc = qp_where_individuals[1]
    googleloc_a = googleloc.split("<http://maps.google.co.uk/distanceto/")
    googleloc_b = googleloc.split("<http://maps.google.co.uk/distancefrom/"
    )
    if googleloc_a[0] == '':
        to_location = googleloc_a[1]
        to_location= to_location.replace('>','')
        dist = skeleton.google_distancematrix(from_location,to_location)
        if (dist['rows'][0]['elements'][0]['status'])== 'OK':
            dist_value = int(dist['rows'][0]['elements'][0]['distance']['
    value'])
        else :
            dist_value = 100000000
    elif googleloc_b[0] == '':
        to_location = from_location
        from_location = googleloc_b[1]
        from_location= from_location.replace('>','')
        dist = skeleton.google_distancematrix(from_location,to_location)
        if (dist['rows'][0]['elements'][0]['status'])== 'OK':
            dist_value = int(dist['rows'][0]['elements'][0]['distance']['
    value'])
        else :
                dist_value = 100000000
    else :
        dist_value = 100000000
    tails = '''<results>
        <result>
        <binding name="'''+varkeys[0]+'''">
        <literal datatype="http://www.w3.org/2001/XMLSchema#integer">'''+
    str(dist_value)+'''</literal>
        </binding>
        </result>
        </results> </sparql>
        '''
    resultant = heading+tails
    print "query processed"
    return resultant


class MyHandler(BaseHTTPServer.BaseHTTPRequestHandler):
    def do_HEAD(s):
        s.send_response(200)
        s.send_header("Content-type", "application/sparql-results+xml")
        s.end_headers()

    def do_POST(s):
        """Respond to a POST request."""
        abc = queryprocess(s)
        print "it was a POST request"
        if not abc is None:
            length = len(abc)
        else:
            length = 0
        statusCode=200
        s.send_response(statusCode)
        mimeType = 'application/sparql-results+xml'
        s.send_header('Content-Type', mimeType)
        s.send_header('Content-Length', length)
        s.end_headers()

        s.wfile.write(abc)


    def do_GET(s):
        abc = queryprocess(s)
        """Respond to a GET request."""
        print "it was a GET request"
        if not abc is None:
            length = len(abc)
        else:
            length = 0
        statusCode=200
        s.send_response(statusCode)
```

```
            mimeType = 'application/sparql-results+xml'
            s.send_header('Content-Type', mimeType)
            s.send_header('Content-Length', length)
            s.end_headers()
            print abc
            s.wfile.write(abc)

if __name__ == '__main__':
    server_class = BaseHTTPServer.HTTPServer
    httpd = server_class((HOST_NAME, PORT_NUMBER), MyHandler)
    print time.asctime(), "Google api Server Starts - %s:%s" % (HOST_NAME,
    PORT_NUMBER)
    try:
        httpd.serve_forever()
    except KeyboardInterrupt:
        pass
    httpd.server_close()
    print time.asctime(), "Server Stops - %s:%s" % (HOST_NAME, PORT_NUMBER)
```

Appendix B

The Data Unit

The following is the code used to simulate the data unit of the Suffolk Constabulary which governs the neighbourhoods identified by the codes J11 and H41.

```
import time
import BaseHTTPServer
import urllib
from fyzz import parse
import json
import httplib2
import types
from collections import Counter
from SPARQLWrapper import SPARQLWrapper, XML

HOST_NAME = 'localhost' # !!!REMEMBER TO CHANGE THIS!!!
PORT_NUMBER = 55005 # Maybe set this to 9000.
Root = False
THEMasterRoot = False
ownuri = "http://police.wwi/SuffolkConstabulary"
owngraph = "http://police.wwi/SuffolkConstabulary/2013
    _01_suffolk_neighbourhood.source"
childrenuri = []
#childrenuri = ["http://police.wwi/SuffolkConstabulary/J11","http://police.
    wwi/SuffolkConstabulary/H41"]
repository = 'play'
repuri = "http://localhost:8080/openrdf-workbench/repositories/%s/query" %
    repository
srepuri = "http://localhost:8080/openrdf-sesame/repositories/%s" %
    repository
crepuri = "http://localhost:8080/openrdf-workbench/repositories/%s/clear" %
    repository
lordrep = "crime"
grepo = "http://localhost:8080/openrdf-workbench/repositories/%s/query" %
    lordrep
sgrepo = "http://localhost:8080/openrdf-sesame/repositories/%s" %lordrep

```

```python


def query_rewrite(ast):
    newprefixes =""
    for every in ast.prefixes :
        theprefixes = "PREFIX "+ every+": <"+ ast.prefixes[every]+">"
        newprefixes = newprefixes+"\n"+theprefixes

    #newselected = "SELECT DISTINCT * "
    qc = "CONSTRUCT { "
    for every in ast.where :
        for newe in every:
            if str(type(newe)) == "<class 'fyzz.ast.SparqlVar'>" :
                nnewe = str(newe)
                nnewe = nnewe.replace("SparqlVar('","")
                nnewe = nnewe.replace("')","")
                newv = " ?"+nnewe
                qc = qc+" "+ newv
            elif str(type(newe)) == "<type 'str'>":
                qc = qc+" "+newe
            elif str(type(newe)) == "<type 'tuple'>":
                if newe[0] == '':
                    qc = qc+" "+ newe[1]
                else:
                    for thekey, theval in ast.prefixes.iteritems():
                        if theval == newe[0]:
                            qc = qc+" "+ thekey +":"+ newe[1]
        qc = qc +" . \n"
    qc = qc +" }"
    newselected = qc

    qline =" WHERE {\n GRAPH <%s> { " %owngraph
    for every in ast.where :
        #qline = qline +"\n GRAPH <%s> { " %owngraph
        qline = qline +"\n OPTIONAL {"
        for newe in every:
            if str(type(newe)) == "<class 'fyzz.ast.SparqlVar'>" :
                nnewe = str(newe)
                nnewe = nnewe.replace("SparqlVar('","")
                nnewe = nnewe.replace("')","")
                newv = " ?"+nnewe
                qline = qline+" "+ newv
            elif str(type(newe)) == "<type 'str'>":
                qline = qline+" "+newe
            elif str(type(newe)) == "<type 'tuple'>":
                if newe[0] == '':
                    qline = qline+" "+ newe[1]
                else:
                    for thekey, theval in ast.prefixes.iteritems():
                        if theval == newe[0]:
                            qline = qline+" "+ thekey +":"+ newe[1]
            qline = qline +" . }"
        qline = qline +"\n } \n }"
        newquery = newprefixes +'\n'+newselected +qline
        if len(ast.orderby) != 0 :
            if str(ast.orderby[0][1]) == "asc":
                nnewes =str(ast.orderby[0][0])
                nnewes = nnewes.replace("SparqlVar('","")
                nnewes = nnewes.replace("')","")
                newquery = newquery +" ORDER BY ?"+nnewes
            elif str(ast.orderby[0][1]) == "desc":
                nnewes =str(ast.orderby[0][0])
                nnewes = nnewes.replace("SparqlVar('","")
                nnewes = nnewes.replace("')","")
                newquery = newquery +" ORDER BY DESC("+nnewes+")"
        if str(ast.limit) != "None" :
            newquery = newquery + " LIMIT "+ str(ast.limit)
        if str(ast.offset) != "None":
            newquery = newquery +" OFFSET "+str(ast.offset)
        print newquery
        return newquery


def local_dump(body, repository, graph):
    print "Loading %s into %s in Sesame" % (repository, graph)
    params = { 'context': '<' + graph + '>' }
    endpoint = "http://localhost:8080/openrdf-sesame/repositories/%s/statements?%s" % (repository, urllib.urlencode(params))
    (response, content) = httplib2.Http().request(endpoint, 'PUT', body, headers={ 'content-type': 'application/rdf+xml' })
    print "local_dump Response %s" % response.status

def retrieveall_sparqlxml(n_graph):
    rquery= "CONSTRUCT {?s ?p ?o } WHERE {GRAPH <%s> {?s ?p ?o } }" %n_graph
    allparams = { 'query': rquery }
    allheaders = {
        'content-type': 'application/x-www-form-urlencoded',
        'accept': 'application/sparql-results+xml'
    }
    rresp, rcontent = httplib2.Http().request(repuri, 'POST', urllib.urlencode(allparams), headers=allheaders)
    return rcontent

def retrieveall_XML(n_graph):
    rquery= "CONSTRUCT {?s ?p ?o } WHERE {GRAPH <%s> {?s ?p ?o } }" %n_graph
    sparqled = SPARQLWrapper(srepuri)
    sparqled.setQuery(rquery)
    sparqled.setReturnFormat(XML)
    results = sparqled.query().convert()
    rcontent = results.serialize(format="xml")
    return rcontent

```

```
def retrieveforroot_XML(rquery):
  allparams = { 'query': rquery }
  allheaders = {
    'content-type': 'application/x-www-form-urlencoded',
    'accept': 'application/sparql-results+xml'
  }
  rresp, rcontent = httplib2.Http().request(repuri, 'POST', urllib.
    urlencode(allparams), headers=allheaders)
  return rcontent

def clear_graph(n_graph):
  allparams = { 'context': n_graph }
  rresp, rcontent = httplib2.Http().request(crepuri, 'POST', urllib.
    urlencode(allparams))
  print rresp

def queryprocess(s):
  paths = urllib.unquote(s.path)
  paths = paths.replace('+',' ') # Check point
  #paths = paths.replace('\n',' ')
  unused, nvqueryin = paths.split('?query=')
  print '\n not virgin query:\n', nvqueryin
  ###querying data
  #Rewrite the query with Optional and Construct
  if THEMasterRoot == True :
    virgin_path = s.path
    virgin_path = virgin_path.replace("%28AVG%20%28%3Fvalue%29%20as%20%3
    Favgvalue%29","")
    paths = urllib.unquote(virgin_path)
    paths = paths.replace('+',' ') # Check point
    #paths = paths.replace('\n',' ')
    unused, queryin = paths.split('?query=')
    print '\virgin nquery:\n', queryin
  else :
    queryin = nvqueryin
  nq = query_rewrite(parse(queryin))
  #Querying data from own repository
  if THEMasterRoot == False :
    sparqled = SPARQLWrapper(sgrepo)
    sparqled.setQuery(nq)
    sparqled.setReturnFormat(XML)
    results = sparqled.query().convert()
    contents = results.serialize(format="xml")
    #Dumping data to temp. repository.
    n_graph = ownuri
    local_dump(contents, repository, n_graph)

  #Send query to children if any
  n_graph = ownuri
  if len(childrenuri) != 0 :
    for every in childrenuri :
        #Getting data from children
        print queryin
        '''
        sparq = SPARQLWrapper(every)
        sparq.setQuery(queryin)
        sparq.setReturnFormat(XML)
        results = sparq.query().convert()
        content = results.serialize(format="xml")
        '''
        print every+"?query="+urllib.quote(queryin)
        resp, content = httplib2.Http().request(every+"?query="+urllib.quote(
    queryin))
        print content
        local_dump(content, repository, n_graph)
  #Constructing all the new data to be send back
  if Root == True :
    #Send reply as sparql result 1) Sparql domino request 2) Madeup domino
    request
    #This is 2)
    retdata = retrieveforroot_XML(nvqueryin)
    #retrieveall_sparqlxml(n_graph)
  else :
    retdata = retrieveall_XML(ownuri)
  #Delete all temporary data stored
  clear_graph (n_graph)
  return retdata

class MyHandler(BaseHTTPServer.BaseHTTPRequestHandler):
    def do_HEAD(s):
        s.send_response(200)
        s.send_header("Content-type", "application/sparql-results+xml")
        s.end_headers()

    def do_POST(s):
        print "It was a POST request"
        abc = queryprocess(s)
        """Respond to a POST request."""
        print "It was a POST request"
        if not abc is None:
            length = len(abc)
        else:
            length = 0
        statusCode=200
        s.send_response(statusCode)
        mimeType = 'application/sparql-results+xml' #Attention
        s.send_header('Content-Type', mimeType)
        s.send_header('Content-Length', length)
        s.end_headers()
        print abc
        s.wfile.write(abc)


    def do_GET(s):
      #Send the queries to children
        print "It was a GET request"
        #Process the SPARQL query with Optional
        abc = queryprocess(s)

        """Respond to a GET request."""
        print "it was a get request"
        if not abc is None:
            length = len(abc)
        else:
            length = 0
        statusCode=200
        s.send_response(statusCode)
        mimeType = 'application/sparql-results+xml' #Attention
        s.send_header('Content-Type', mimeType)
        s.send_header('Content-Length', length)
        s.end_headers()
        s.wfile.write(abc)


if __name__ == '__main__':
    server_class = BaseHTTPServer.HTTPServer
    httpd = server_class((HOST_NAME, PORT_NUMBER), MyHandler)
    print time.asctime(), "Zoopla to-rent %s Server Starts - %s:%s" % (
   ownuri, HOST_NAME, PORT_NUMBER)
    try:
        httpd.serve_forever()
    except KeyboardInterrupt:
        pass
    httpd.server_close()
    print time.asctime(), "Server Stops - %s:%s" % (HOST_NAME, PORT_NUMBER)
```

Listing B.1: This is the Python code used to create a DU to simulate the WWI enviornment.