A Dependency Modelling Approach for the Management of Ontology-Based Integration Systems

A thesis submitted to the
University of Dublin, Trinity College
for the degree of
Doctor of Philosophy

Aidan Boran
Alcatel-Lucent,
Blanchardstown Industrial Park,
Blanchardstown,
Dublin 15.

2010

Declaration

I, the undersigned, declare that this work has not been previously submitted as an exercise for a degree at this or any other University, and that, unless otherwise stated, it is entirely my own work.

_____

Aidan Boran
September 2010

Permission to lend or copy

I, the undersigned, agree that the Trinity College Library may lend or copy this thesis upon request.

_____

Aidan Boran
September 2010

ACKNOWLEDGEMENTS

"In the high country of the mind one has to become adjusted to the thinner air of uncertainty..."
— Robert M. Pirsig

I would like to thank my supervisors Declan O'Sullivan and Vincent Wade for agreeing to supervise my research, for their unflagging support and insightful contributions.

I would also like to thank Lou Manzione, Lawrence Cowsar, Sam Samuel, Ben Lowe and Julie Byrne from the wonderful Bell Labs, for their oversight, financial and technical support during this research.

My thanks to all in the KDEG research group for the great collaboration and friendship.

A special word of thanks to my wife, Audrey, for her unconditional support, her confidence and belief. I could not repay everything you have done for me. When we were getting married, she thought she was getting "Mr. Dependable" – in fact it turned out she got "Mr. Dependency"!

Finally, thanks are due to my son and daughter, Tom and Aimee, for keeping my feet on the ground and keeping me up to date with the football scores.

This thesis is dedicated to the memory of my father and mother, Gerry and Noreen Boran.

ABSTRACT

Ontology-based approaches that formally represent the meaning of information in a system, offer the hope of dealing with semantic heterogeneity when integrating heterogeneous data sources [Halevy 2001, Noy 2004, Wache et al. 2001, Doan and Halevy 2005, Pollock 2002]. While these ontology-based approaches offer significant advantages [Cruz and Xiao 2005, Noy 2004, Wache et al. 2001, Halevy 2005] over traditional approaches (e.g. ETL[1]), they tend to require semantic mappings to create loose coupling of systems to enable integration. The mappings may serve to relate ontologies to other ontologies (inter-ontology mappings) or to relate ontologies to underlying information sources (e.g. a database). However, when such semantic systems are scaled up, the semantic mappings also need to grow and evolve [Bernstein and Melnik 2007, Velegrakis et al. 2003, Yu and Popa 2005, Halevy et al. 2005, An and Topaloglou 2007]. Failure to provide methods to manage and evolve the semantic mappings can make the integration systems brittle.

There is currently little research to help identify, manage and evolve semantic mappings when the integration system is evolving [Bernstein and Melnik 2007, Haas 2007, Doan and Halevy 2005, Kondylakis et al. 2009]. The first part of the evolution problem, identifying and managing the mappings that need to evolve, is addressed by the dependency model in this thesis. The dependency model is important in the context of ontology-based data integration because it promises to enhance the scalability of integration systems by allowing them to find which elements of the integration system are impacted when a data source or ontology changes.

This thesis has developed an ontology-based domain specific dependency model, a more general dependency metamodel and a tool that can represent and analyse dependencies that occur between mappings, ontologies and databases in an ontology-based integration system.

The approach has been developed and evaluated using two industrial datasets.

[1] Extract, Transform and Load.

TABLE OF CONTENTS

LIST OF FIGURES

1 INTRODUCTION

1.1 Motivation

Today, large enterprises have deployed many information and database systems across distinct functional areas of the enterprise (e.g. logistics, sales, production, finance, human resources). The widespread adoption of these systems has created the problem of islands of heterogeneous and distributed information [Bernstein and Haas 2008, Haas 2007, Lowell Database Report 2003]. These islands make the development of integrated processes and applications difficult [Bernstein and Haas 2008, Haas 2007].

Within large enterprises, there is a business need for enterprise applications that can operate across functional areas. These applications must facilitate automated integration to allow business professionals to make informed decisions [Haas 2007, Lowell Database Report, IBM 2004, Halevy et al. 2005]. The "distribution" of information sources makes integration difficult because the databases and information models tend to be managed and evolved separately [Halevy 2005]. Similarly, the "heterogeneity" of the information sources makes integration difficult as it manifests itself on three levels namely syntactic, schematic and semantic levels [Cruz and Xiao 2005, Sheth et al 1999].

Such data integration problems have meant that enterprises spend a great deal of time and money on attempting to combine information from different sources into a unified format. Frequently cited as the biggest and most expensive challenge that information-technology organisations face, information integration is thought to consume about 40% of their IT budget [Bernstein and Haas 2008].

Existing data integration solutions (e.g. consolidation, federation and replication systems) are capable of resolving syntactic and schematic heterogeneities in the underlying sources but they are not capable of semantic integration [Cruz and Xiao 2005, Halevy 2005]. Since syntactic approaches do not encode meaning in the data or messages passed through the integration systems, it becomes necessary to hardcode this meaning in the applications themselves. Such hard coding leads to integration systems that are difficult to maintain [Halevy et al. 2005, Zhou et al. 2006].

Other approaches, that formally represent the meaning of data in a system, offer the hope of dealing with semantic heterogeneities. While these semantic (ontology) based approaches offer significant advantages, they tend to require semantic mappings to

create relationships between the ontologies and data sources of the systems to enable integration [Cruz and Xiao 2005, Noy 2004, Wache et al. 2001]. However, as the semantic systems are scaled up, semantic mappings also need to grow and evolve [Bernstein and Melnik 2007, Velegrakis et al. 2003, Yu and Popa 2005, Halevy et al. 2005, An and Topaloglou 2007].

In spite of decades of research into data integration, recent surveys indicate that a number of important challenges persist [Bernstein and Haas 2008, Haas 2007, Lowell Database Report 2003, IBM 2004, Halevy et al. 2005, Zhou et al. 2006]. Bernstein [Bernstein and Melnik 2007] described "data programmability" as the goal of making access to large shared data sources easier. However, he noted that the data programmability problem remains today due to the need for complex mappings between different representations of data. Despite decades of research into databases and data management, coping with heterogeneity remains one of the most time-consuming data management problems. Bernstein [Bernstein and Melnik 2007] indicates that anecdotal evidence suggests that it accounts for 40% of the work carried out by enterprise IT departments. Bernstein has proposed an extensive model management approach that seeks to provide lifecycle support for the mappings that are central to the resolution of the data programmability problem. As noted by Bernstein, many data integration approaches that are used in enterprise integration make use of mappings (e.g. Extract, Transform and Load and message mapping tools). Despite the broad usage of mappings across these approaches, there is little commonality in the approach to the management of the mappings [Bernstein and Melnik 2007, Doan and Halevy 2005, Halevy et al. 2005].

In [Halevy et al. 2005], scalability and metadata management are identified as two of the key challenges facing enterprise information integration. In [Zhou et al. 2006], it is pointed out that from a technical viewpoint the scalability of current integration toolsets rely on specialists having a deep understanding of the data, the underlying schema and the relationships across the various data sources.

This work has developed a model and tool to represent the dependencies that arise within ontology-based integration systems due to the use of mappings. The model of the mapping dependencies addresses the first step of mapping evolution i.e. understanding what parts of the integration system are affected by a proposed change in the data sources. The approach enables a deep understanding to be developed of the

dependency relations across the key parts of the integration system. The author of this thesis believes that this is a key step that will allow the integration system to evolve gracefully when the underlying data sources change.

1.2 Research Question

An important aspect for the deployment of any integration system in an industrial context is its ability to adapt to changes in the underlying data sources. In ontology-based integration systems, changes to the data sources can also impact the ontologies and mappings that comprise the integration system [Bernstein and Melnik 2007, Velegrakis et al. 2003, Yu and Popa 2005].

Thus, to ensure that the system can evolve when changes occur in the underlying data sources, it is critical to be able to identify and evolve those parts of the ontology and mappings that are impacted. This thesis asserts that a model of the dependencies that arise between the ontologies, mappings and data sources provides a potential solution to this evolution problem. The research question for this thesis is defined as:

How and to what extent can a dependency model enhance integration performance by allowing for the identification of and support for the management of the semantic mapping dependencies of an integration system?

In the context of the research in this these, a semantic mapping is defined a correspondence between elements of different schema. Schema mappings are typically used to support query rewriting and/or data transformations in data integration systems [Halevy et al. 2006, Lenzerini 2002].

Many factors influence the integration performance such as the throughput, capacity or speed (e.g. response time) of the system. The importance of a unified approach to the measurement of integration performance has been regularly identified [Lowell Database Report 2003, Halevy et al. 2005]. However, only a few unified benchmarks exist [Böhm et al 2008, Othayoth and Poess 2006, Böhme and Rahm 2001]. These approaches focus on processing performance of the integration system [Böhm et al 2008, Othayoth and Poess 2006]. The research in this thesis required a measurement of the ability of the integration system to integrate heterogeneous data source rather than a measurement of processing performance. This was required to measure how well a new approach to integration coped with semantic heterogeneity. In particular, this research has focused on the ability of ontology-based approaches for integration to

cope with semantic heterogeneity. In the industrial context, a key requirement for integration systems is the ability to cope with changes to the underlying data sources [Bernstein and Melnik 2007, Velegrakis et al. 2003, Yu and Popa 2005]. To measure these aspects of integration performance, this thesis has defined two integration performance metrics called "Integration Quality" and "Dependency Identification Performance".

"Integration Quality**"** is defined as:

- A measure of the ability of the system to carry out integrations across a range of different types of data heterogeneity.

This metric provides a qualitative measurement of the ability of the integration system to cope with different types of heterogeneity. The THALIA integration benchmark [Stonebraker 2005] provides an ideal framework to measure this aspect of integration performance since it provides a set of tests to execute based on a systematic classification of different types of syntactic and semantic heterogeneity.

"Dependency Identification Performance" is defined as:

- A measure of the ability of the system to accurately and quickly identify the mapping dependencies.

This second aspect of integration performance is focused on the ability of the integration system to evolve its mappings when new data sources are added. Dependency identification performance is important to understand because the first step of mapping evolution is to identify which mappings are impacted by the proposed change. In this thesis, dependencies are used to support the evolution of mappings and "Dependency Identification Performance" is calculated by measuring the accuracy of the dependencies found and the time taken to find the dependencies.

Four objectives were derived in order to address the research question:

1) Perform a state of the art review of approaches for semantically linking local[2] schema and aggregate or global schema[3].

[2] Local schema refers to a schema that represents the local sources to be integrated.
[3] Global schema refers to a common view of sources to be integrated.

2) Research and develop a model to define the dependencies that arise when creating semantic links between schemas to support an ontology-based integration approach between local schemas and global schemas.

3) Research and develop a prototype tool capable of supporting this dependency modelling approach.

4) Evaluate the dependency model and tool using industrial use cases.

1.3 Research Methodology

This research has been carried out in an iterative manner using the four step process from the action-based research methodology [Fisher 2004, O'Brien R. 2001].

The action based methodology was selected because it provided an adaptive, flexible and participatory approach to research. The approach involves an iterative inquiry process that leads to a refinement of the research question. Each iteration involves "plan", "action", "observe" and "refine" steps. The iterative inquiry process afforded the flexibility needed to conduct research in an environment that is subject to regular process, management and personnel changes such as the supply chains of large enterprises where the use cases in this research originated. The participatory nature of action based research was also important because it allowed business professionals from Alcatel-Lucent to influence the research by supporting use case development, to provide real industrial data sets and to participate in case studies.

The action-based research process was conducted using a series of iterations as shown in Figure 1-1. The "observe" and "reflect" steps in the action-based methodology were combined into a single step during the running of the experiments that were conducted in this research.

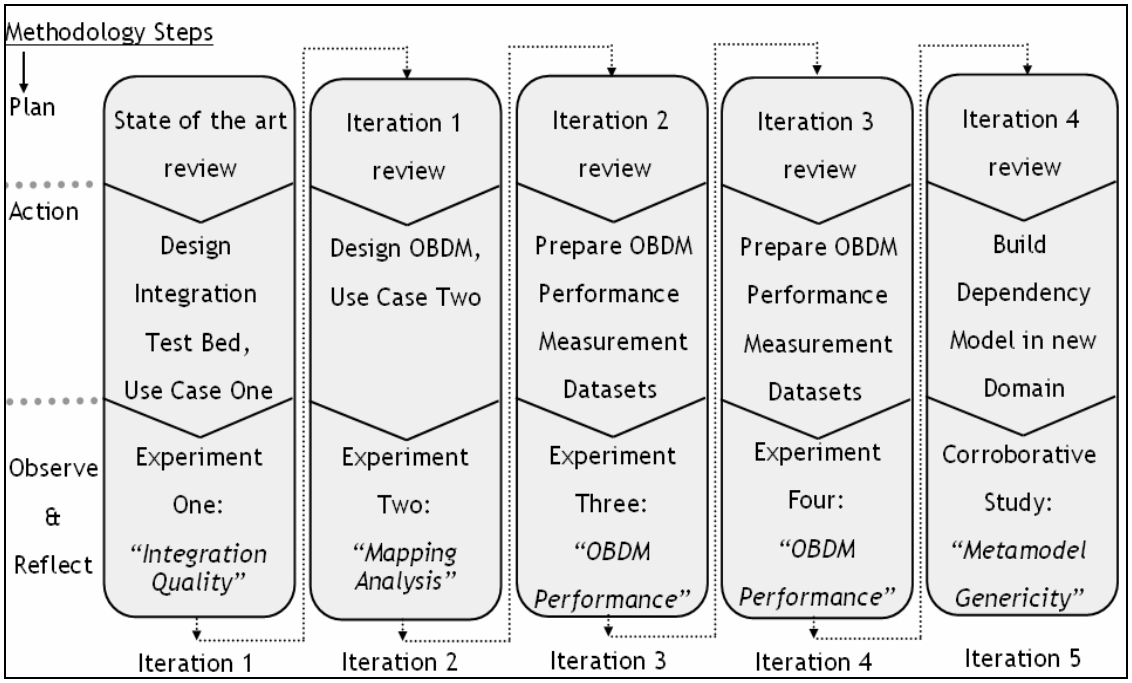

Figure 1-1: Overview of Research Methodology.

The first iteration of the action based methodology began with an analysis of an industrial use case from the Alcatel-Lucent Product Line Management supply chain. The use case required the integration of multiple data sources that contained both semantic and structural heterogeneity. To understand the best approach to tackle this problem, a review of the state of the art in information integration was undertaken. Using the outputs of this review, a generalised ontology-based integration test bed was created to support the integration use case. An experiment was then designed to apply the test bed to investigate the key issues when deploying ontology-based integration systems using the industrial use case. The integration performance of the test bed was verified by measuring its "Integration Quality" metric as defined earlier in Section 1.2. By analysing the issues that arose during the experiment it was hypothesised that the mappings that are part of the generalised ontology-based integration test bed create complex couplings between different parts of the integration system and that these couplings make the mappings difficult to evolve. This research iteration is described in detail in experiment one (Section 4.2).

The next research iteration was designed to analyse the complex coupling of the mappings in the integration system. A model of the mapping dependencies was used to show the dependency relationships that exist between mappings from the generalised ontology-based integration system. The model was developed in OWL [OWL] to

enable an ontological reasoner to automatically compute the dependencies. This is called the ontology-based dependency model (OBDM). A tool called TomE (Towards Ontology Mapping Evolution) was developed to instantiate the OBDM and to support the analysis of dependencies in the ontology-based integration system. An experiment was then developed to analyse the dependencies between the mappings from the generalised ontology-based test bed. The mappings arose from a second industrial use case from the Alcatel-Lucent logistics supply chain. Analysis of the dependencies found using the OBDM showed that approximately 30% of the mappings exhibit complex dependencies with other parts of the integration system. From the results of this experiment, a hypothesis was developed that these mapping dependencies would be difficult to identify without tool support. This research iteration is described in detail in experiment two (Section 4.4).

The next research iteration was developed to demonstrate the difficulty of mapping dependency analysis without tool support. To achieve this, a manual approach to dependency analysis was developed with the help of integration and logistics specialists. A manual approach was needed because current integration approaches provide very limited support for mapping maintenance as noted in the state of the art review [Bernstein and Melnik 2007, Haas 2007, Doan and Halevy 2005, Kondylakis et al. 2009]. The performance and accuracy of a manual approach to dependency analysis and OBDM were compared using the "Dependency Identification Performance" metric as discussed earlier (Section 1.2). To achieve this, a group of 18 users were provided with three sets of theoretical semantic mappings. The group was asked to carry out a number of timed dependency analysis tasks. The semantic mappings used in the tasks were designed to contain mappings of different complexities and represent a theoretical set of mapping evolution needs. This research iteration is described in experiment three (Sections 4.6).

The next research iteration was run to evaluate the performance of the OBDM and TomE tool when used to support the evolution of the mappings when performing a real mapping evolution task. These evolution tasks arose when a new logistics data source needed to be added to the use case described in experiment two. The new data set required both the update of existing mappings and the addition of new mappings. The OBDM and TomE tool were used to analyse which mappings were impacted by the

addition of the new logistics data. This research iteration is described in experiment four (Sections 4.8).

The final iteration carried out a corroborative study to provide an indication of the genericity of the dependency metamodel that was used to build the ontology-based dependency model. This study was carried out to assess the ability of the metamodel to be applied in other domains. The study involved the development of a dependency model to localise faults in a domestic electrical circuit. A domestic electrical circuit was selected as the application domain because it provided a different set of dependencies from the ontology-based integration system where the metamodel was previously applied. A domain expert on electrical engineering was coached through an eight-step process to build a dependency model, using the metamodel, of an electrical circuit and to carry out a dependency analysis exercise using the model. This iteration is described in the evaluation chapter (Section 4.10).

1.3.1 Summary of Industrial Use Cases and Data Sets

Throughout this work, two real integration problems and data sets from the Alcatel-Lucent supply chain were used. The integration problems and datasets provided excellent test data since they originate from multiple IT systems, multiple processes and in the case of Alcatel-Lucent multiple companies.

The first integration problem required the generation of a report that integrated financial information from the Sales, Product Lifecycle Management (PLM) and Forecasting domains. To mitigate any risk associated with lack of consistency between sales and forecasting views of the PLM, organisations attempt to balance forecasting and sales opportunities [Gilliland 2002]. In Alcatel-Lucent's supply chain, these risks are managed using a manual integration of financial information from each system. The report that is produced by this manual integration supplements the financial information with an integrated view of the customers and products. This process involves many manual steps to export data from the distributed databases and rework within a spreadsheet where the various heterogeneities are resolved manually.

The second integration problem came from Alcatel-Lucent's Reverse Logistics process. This process used a manual process to select the lowest cost shipping option. To

simplify this process, a software application (ALTO[4]) was developed to automatically generate simple routing instructions called routing guides. To simplify the database update process of this application, the ontology-based integration platform was deployed to integrate the different logistics supplier rate formats into a single common model of logistics. From the central model, the scripts to load the ALTO database could be automatically generated.

1.4 Contribution

The major contribution of this thesis is the ontology-based dependency model (OBDM) that can represent the dependencies that occur between mappings, ontologies and databases in an ontology-based integration system. The ontology-based dependency model will be beneficial to system integrators when developing approaches to improve the ability of the enterprise integration systems to evolve their mappings when data sources change.

The approach supports mapping evolution by providing three levels of the dependency graphs that enable the system integrators to manage and evolve the mappings in the integration system. This is achieved by providing dependency views that allow the user to focus in on areas of high dependence initially and then to progressively drill down to the detail to understand the impact of each dependency. The OBDM is novel because it automatically computes the dependency relationships. The automation is achieved through its instrumental usage of ontological reasoning that requires coding only to invoke the ontological reasoner. This contribution addresses, in part, the gap in the state of the art regarding the lack of tools and techniques to support the management of mappings [Bernstein and Melnik 2007, Velegrakis et al. 2003, Yu and Popa 2005, Doan and Halevy 2005, Halevy et al. 2005] because it supports the first step of mapping evolution i.e. how to identify which mappings are impacted when a data source changes. The approach of using a dependency model of mappings could be used to supplement the ontology-based integration frameworks and tools described in the state of the art review (Section 2.5.4).

The ontology-based dependency model (OBDM) was tested using industrial data from real systems from the Alcatel-Lucent supply chain. This provided a challenging set of

[4] Alcatel-Lucent Transport Optimization (ALTO) is deployed in the reverse logistics supply chain.

heterogeneous data sources for the system. The results of the evaluation of the OBDM show how the approach enables the integration specialist to quickly identify all the impacts of a complex set of changes to the data sources. By providing progressive detail of the dependencies, the integration specialist can quickly focus and assess what needs to be changed in the system. The results show that dependencies found can also be used to develop regression tests after the integration system has been updated. This analysis is useful for developers of integration systems who wish to understand the complexity involved in evolution of mappings in an industrial context.

The design of the generalised ontology-based integration test system and the setup, results and conclusions of experiment one were published in:

> Aidan Boran, Declan O'Sullivan and Vincent Wade, A Case Study of an Ontology-Driven Dynamic Data Integration in a Telecommunications Supply Chain. *Proceedings of the Workshop on the First Industrial Results of Semantic Technologies (FIRST2007) at ISWC/ASWC2007, Busan, South Korea,* 2007.

The design of the ontology-based dependency model and the result of experiment two were published in:

> Aidan Boran, Declan O'Sullivan and Vincent Wade, Managing Ontology Based Integration Systems using Dependencies. *Proceedings of the Workshop on the Managing Ubiquitous Communications and Services Workshop (MUCS) at PerCom 2010, Mannheim, Germany,* 2010.

A minor contribution is the ontology-based dependency metamodel from which the domain specific dependency model was created. The ontology-based dependency metamodel could be beneficial to other management systems (e.g. service and fault management) which need to model dependencies between parts of the system. The genericity of the metamodel has been tested across two large industrial datasets that originated from a dynamic industrial environment with multiple IT systems and multiple processes. A corroborative study was carried out to demonstrate the application of the metamodel in an entirely different domain (i.e. dependency analysis in a domestic electrical circuit). The compact nature of the metamodel facilitates design flexibility, behaviour reuse and scalability. This enabled a simple process to be

defined, in Section 3.2.4, to create domain specific models from the dependency metamodel. To the authors knowledge, an ontology-based dependency metamodel has not been published before that has support for both behavioural and descriptive attributes and that can enable reasoning over the dependency relationships in the model to enable automatic computation of dependencies.

The design of the ontology-based dependency metamodel, model and toolset was published in a short paper at Network Operations and Management Symposium 2010:

> Aidan Boran, Declan O'Sullivan and Vincent Wade, A Dependency Modelling Approach for the Management of Ontology Based Integration systems. Network Operations and Management Symposium (NOMS), Osaka, Japan, *2010*.

1.5 Overview of thesis

The remainder of the thesis is structured as described below.

Chapter 2 contains a review of the state of the art in ontology-based integration system and dependency modelling. The chapter gives a brief overview of the data integration space, a detailed description of the ontology-based integration research, mapping management and dependency analysis.

Chapter 3 describes the design of the dependency metamodel, a dependency model derived from the metamodel that is specialised to the ontology-based integration domain and a tool called TomE that was created to instantiate and reason over the dependency model. The chapter concludes with a worked example of the dependency model as applied to ontology-based integration systems.

Chapter 4 describes the four experiments and a corroborative study that were conducted to evaluate the metamodelling approach. The first experiment created an environment in which the performance of a generalised ontology-based integration system was measured using data from product line management systems in an industrial context. The second experiment developed the theoretic basis to allow the evolution of mappings in an ontology-based integration system. The third experiment evaluated the performance of the dependency modelling approach by measuring the accuracy of and time taken to complete a dependency analysis exercise using the

OBDM and a manual approach to dependency analysis. The fourth experiment demonstrated the utility of the dependency modelling approach when it is applied to an ontology-based integration system that needed to incorporate a new dataset into its integration. The corroborative study applied the ontology-based dependency metamodel in a new domain to test the genericity of the metamodel when applied in other domains.

Chapter 5 describes the conclusion, contributions and future work of this research.

Appendix I provides the OWL code for the ontology-based dependency metamodel and the ontology-based dependency model (OBDM). Appendix II provides the data associated with the experiments carried out in this thesis. Appendix III provides a simple worked example of the inputs and outputs for the TomE tool. Appendix IV provides the overview of the directory structure for the code for HotFusion and TomE tools that is supplied on DVD with this thesis.

2 STATE OF THE ART

2.1 Introduction

This chapter reviews the state of the art in dependency modelling and analysis, schema and ontology evolution and ontology-based approaches to information integration. The reasons for selecting these areas are outlined below in Sections 2.1.1, 2.1.2, 2.1.3 and 2.1.4.

Before reviewing the state of the art in these three areas, this chapter begins with a review of the background and context for other (non ontology-based) approaches to information integration in Section 2.2. This is important because it describes the current approaches to information integration and provides context for the review and comparison of dependency modelling approach for the maintenance of mappings taken in this research. The chapter concludes (Section 2.7) with a description of the choices made for the background technologies used to support the ontology-based dependency modelling approach taken in this thesis.

The state of art is provided in four parts as described below.

2.1.1 Overview of Information Integration Approaches

Section 2.2 provides an overview of the approaches and technologies used to support information integration. Information integration is a complex space with many fields of endeavour spanning both the business and research communities [Bernstein and Haas 2008, Halevy et al. 2005, Zhou et al. 2006]. The review presented here provides the overall context for the ontology-based approaches discussed in detail later in this chapter.

2.1.2 State of the Art in Dependency Modelling and Dependency Analysis

Section 2.3 provides a review of the prior art in dependency modelling and dependency analysis (Section 2.3). Approaches to dependency modelling are important to consider because experiment one, in this thesis, developed the hypothesis that the complex nature of the mappings makes it difficult to quickly and accurately find the mappings that are impacted when a data source changes. Experiment two evaluated the hypothesis that this difficultly in managing changes to the data sources and mappings could be improved by modelling the dependencies that exist between the parts of the

ontology-based integration system. This review compares the dependency modelling approach taken in this research with the prior art.

2.1.3 State of the Art in the Usage of Mappings in Schema and Ontology Evolution

Mappings are a fundamental part of the approaches taken to schema and ontology evolution [Velegrakis et al. 2003, Noy and Klein 2002]. In the context of information integration systems, the evolution of schemas is important to allow the integration system to evolve as the schemas change. There are similarities between the usage of schema mappings and ontology mappings i.e. in the context of information integration the mappings are used to provide transformations between schemas or representation of data sources [Kondylakis et al. 2009, Lenzerini 2002]. Therefore, the state of art continues in Section 2.4 with a review of the prior art in schema and ontology evolution to understand the relevance of the techniques in the ontology integration domain. The approaches to schema and ontology mapping management are discussed in the context of the ontology-based mapping dependency management approach taken in this thesis. Note that, while there are similarities between these areas, there are also many differences between the areas as noted in [Kondylakis et al. 2009, Noy and Klein 2002].

2.1.4 State of the Art in Ontology-based Integration

Section 2.5 reviews the state of the art on approaches to information integration that use ontologies (ontology-based approaches). The research in this thesis developed a generalised ontology-based integration test bed. A number of integration systems that use ontology are described and compared to the approach used to create the test bed in this thesis.

2.2 Overview of the Information Integration Approaches

In [Bernstein and Haas 2008], Bernstein describes five architectural approaches for information integration that are summarised below:

- **Data Warehouses:** A data warehouse is a database that consolidates data from multiple sources and integrates it into a single source. This requires the creation of a single database schema for the warehouse and the loading of individual data sources into the warehouse. Regular synchronisation between the data warehouse

and the data sources is required to ensure that the information in the warehouse is up to date.

- **Extract, Transform and Load (ETL):** ETL approaches are typically used to simplify the loading of data into data warehouses. ETL technologies are realised as tool suites that provide loading, cleansing and querying functionality for the data warehouse.
- **Virtual Data Integration (VDI):** Data warehouses materialise the individual data sources in an integrated database. Virtual data integration offers users a mediated database schema to support the execution of queries. Queries are run against the mediated schema and the VDI software transforms the user query to queries over the individual data sources. (VDI is often called data federation. Data federation provides a single virtual view of one or more data sources. Typically, queries are issued against these virtual views and the federation system resolves these queries using either global-as-view or local-as-view approaches to access the data sources.)
- **Message Mapping:** Independently developed applications can be integrated using message oriented middleware that perform information integration functions for the enterprise. The integration functions can occur at the protocol level or at the data level (e.g. transform a sales order from one format to another).
- **Object-to-Relational Mappers:** This type of technology is used to mediate between the relational database schema and the object-oriented design approaches taken when designing software applications. Many development environments (e.g. NetBeans[5]) provide automated support to create Java classes for a relational schema using this technology.

Bernstein concludes the review of architectural approaches to integration with a discussion on "Document" and "Portal" management approaches. Today, enterprises tend to store a wide variety of information in document formats that can be easily accessed and distributed to the desktop of users. In this context, integration is focused on providing a single document store with indexing to enable search over the document store.

Bernstein described the enabling technologies that lie at the heart of these tools as:

[5] NetBeans is an integrated development environment (IDE) for developing a wide variety of applications.

- **Extensible Markup Language (XML):** XML is a mark up language which is used to mark up with user defined tags the content in a document. XML supports information integration by providing a common representation of the data.
- **Schema Standards:** In the review by Bernstein, schema standards are discussed at the most general level and include database, XML and ontologies. Schema standards support integration because data is easier to integrate if the data sources use the same schema.
- **Schema Mapping & Matching:** Bernstein describes schema mapping and matching as fundamental technologies for integration in the review. Schema mapping tools enable transformations (mappings) to be created between individual data sources and a mediated schema. Because large schemas can have many thousands of schema elements, schema matching algorithms have been an important research area. A schema matching algorithm uses a variety of techniques (e.g. heuristics or machine learning) to find candidate matches between schema elements and thus support the user in schema management.

In [Zhou et al. 2006], data integration was classified into two categories, application centric integration (ACI) and data centric integration (DCI).

ACI approaches refer to enterprise application integration (EAI) techniques that integrate applications through the use of message brokers. EAI is defined as approaches (software and architectures) to integrate a set of computer applications. Two basic EAI patterns exist, the mediation pattern where the EAI system acts as a broker between communication systems and the Federation pattern where the EAI system acts as a global proxy for all incoming requests.

DCI approaches refer to both data warehousing and data federation approaches as discussed earlier. DCI includes Enterprise Information Integration (EII) that is a more recent term and is defined as the integration of data from multiple systems into a unified and consistent view for the end user. It is closely related to data federation because EII is focused toward the end user and not an application as in EAI. EII requires the use of an information model to represent the domain of interest whereas federation tends to use a global schema.

As noted in [Zhou and Wang 2006], most current enterprise information integration approaches are based on principles of loosely coupled federated systems (e.g. IBM Information Integrator[6], BEA Liquid Data[7]).

In [Halevy et al. 2005], Halevy et al. noted the inability of these information integration approaches to cope with semantic heterogeneity. Ontology-based approaches provide expressive description languages (e.g. OWL [OWL]) that can potentially support the resolution of semantic heterogeneity between schemas and enable automated reasoning over the schema. The expressivity of the semantic description languages offers considerable advantages over XML or relational schema when creating conceptualisation of the information in any enterprise (e.g. OWL supports of classes, subsumption and object properties). OWL also enabled format reasoning over the model. This is a significant advantage that arises from the formal semantics of the OWL language.

The scalability of current EII approaches is also discussed in [Halevy et al. 2005, Zhou and Wang 2006], where it is noted that efficient scaling of the approaches is complex due to the difficulty in constructing and maintaining a shared schema for a large number of evolving data sources.

In this thesis, an ontology-based approach was taken to construct a generalised integration test bed that used the expressive power and reasoning capability of OWL to support the development of domain and data source ontologies. The domain and data source ontologies are analogous to the mediated and local schema used in non-semantic approaches. The ontology-based test bed was used in experiments one and two to resolve semantic heterogeneities in a selection of data sources from the Alcatel-Lucent supply chain.

[6] IBM Information Integration Suite. http://www-01.ibm.com/software/data/integration/
[7] BEA LiquidData Suite. http://download.oracle.com/docs/cd/E13190_01/liquiddata/docs81/index.html

2.3 State of the Art - Dependency

Models of dependency of varying formalism and complexity have been used widely across a range of application areas. This state of the art review of dependency has focused on the applications of dependency and formalisms used in those applications. These areas were selected to enable an understanding of the breadth of application opportunity for dependencies and the approaches in these applications to formalise dependencies.

The first section reviews research on the applications of dependencies and dependency analysis across several areas of application (e.g. service management, software configuration management).

The second section reviews research on efforts to describe and classify dependencies.

2.3.1 Application of Dependencies and Dependency Analysis

This section reviews several applications which use dependencies to carry out a range of management functions (e.g. service management, fault analysis). For each application both the role that dependency analysis plays and the types of analysis that are carried out are discussed. This review enables us to develop an understanding of the importance and breadth of dependency analysis. The review starts by looking at how dependencies are used in service management [Ensel and Keller 2002, Keller et al. 2000, Cox et al. 2001, Wang and Capretz 2009, Ensel 2001], continues with a review of the application of dependency to test management [Borner and Paech 2009], workflow analysis [Varol and Bayrak 2010], software dependency management [Luo and Diao 2009, Sangal et al. 2005] and concludes with a review of application in network management [Gruschke 1998, Kar et al. 2000, Brown et al. 2001].

Keller and Ensel address the role of dependencies in distributed service management [Ensel and Keller 2002, Keller et al. 2000]. Keller notes the importance of dependency analysis in today's networked environment where applications and services depend on many other supporting services. Dependencies are formed between various components of a distributed system. The dependency relationship exists between components if one component requires another component to carry out its tasks. Two models of the dependencies in the service management domain were created. One model, called the Functional model by Keller, defined generic service dependencies (e.g. name service, database service). The other model, called the Structural model by

Keller, contains detailed descriptions of the dependencies between the components that realise the broad services defined in the Functional model.

In [Cox et al. 2001], Cox and Delugach apply a more formal dependency model to two simple examples. One example defines twelve unidirectional dependencies between components of a computer system (i.e. Browser, Email, Network, and Word Processing Package). Another example defines six dependencies between departments (i.e. Contracts, Proposal and Engineering Departments) in an enterprise. The importance of the type attributes are discussed in the context of the second example where it is noted that adding attributes to the dependency relationship enabled different types of dependency relationship to be distinguished.

Wang and Capretz [Wang and Capretz 2009] propose a model of service dependencies to support the evolution of web services. Four types of service dependency are identified that are needed to describe the types of relationships that exist between services. The semantics of each dependency relation are clearly defined however the relationships are specific to the domain. Collections of dependencies are represented as directed graphs. Service dependency matrices can be constructed from the graphs to support impact analysis.

Ensel presents an approach to automatic discovery of dependencies [Ensel 2001]. Dependencies between IT services in a heterogeneous network are constructed using a neural network and data collected during specially prepared data collection agents distributed in the network. The dependency model contains a simple ‘depends on’ relationship between two services and the work is predominately focused on the collection and automatic detection of the simple dependencies.

A model of dependencies is used by Borner and Paech to support the selection of test cases for the integration test process [Borner and Paech 2009]. The approach taken is domain specific and applies a simple domain specific dependency model in that domain. A dependency is defined as a simple unidirectional relationship between two components in a software system. Dependency attributes are defined to represent the important characteristics of the domain (e.g., dependencies exist because of class inheritance). A bespoke tool is used to analyse source code files and extract information that is loaded into an SQL database. Once the dependencies in the system have been defined, statistical correlations between the dependent components and the

errors found (as reported in a software bug tracking system) were identified. These correlations enabled the identification of the dependencies that had a higher probability of containing errors (in the underlying components) and thus provides input to the selection of integration test cases.

Varol and Bayrak [Varol and Bayrak 2010] use a simple notion of dependency between operators of a workflow is used to generate workflows. The dependency relations are used to support an algorithm that selects the best placement of operators in a workflow. The approach is focused mainly on the workflow generation and thus makes little comment on the dependency graphs illustrated in the work.

Luo and Diao define four types of feature dependencies (global, local, operational and impact dependencies) in [Luo and Diao 2009] that are used to build a domain dependency model of the features in a software product. The semantics of each dependency relation is defined clearly but the relationships are specific to the domain. This approach proposes to investigate feature transitivity and deduction from the transitivity in the future.

Dependency models have also been used for some time for modelling of complex software architectures [Sangal et al. 2005]. In this approach, dependencies are extracted from the code by a conventional static analysis and shown in a tabular form known as the 'Dependency Structure Matrix' (DSM). A variety of algorithms are available to help organise the matrix in a form that reflects the architecture and highlights patterns and problematic dependencies.

An ontology-based approach is taken to the analysis of dependencies by Drabble et al in [Drabble et al. 2009]. Node and Event/Action ontologies are defined. The approach used Protégé to build the ontologies, however it is not clear what ontology language is used (e.g. OWL-DL). A node that exhibits a dependency is represented by a "Dependency" Class and "dependentUpon" and "hasDependency" relations. The event/action ontology provides an interesting and valuable addition to the domain model because it appears to enable a bridge between events occurring in the domain and the description of the dependencies in the domain. The architecture mentions the use of reasoning over dependency relationships (e.g. transitivity) using a reasoning tool called "Athena"; however no details on the reasoning carried out are given. The authors claim that the approach enabled an information bridging service that allowed

information from different and disparate sources to be brought together based on the dependencies implicit in the system.

In [Maddox and Shin 2009], Maddox and Shin propose a computational framework in which dependencies between geo-spatial referencing variables are automatically examined. The framework proceeds in four steps. The first and second steps are responsible for the gathering and reformatting of the geo-spatial data into a common relational database format. The third and fourth steps define and use the concepts of homogeneity, selectivity and exclusivity between elements of the relational database table. A set of heuristics rules are then applied to identify potential dependencies in the data. While the notion of dependency is secondary in this work to the definition of the data mining approaches taken, the value that the dependency analysis provides in helping end users understand the data dependencies is noted by the authors.

In [Deng et al. 2004], Deng et al describe an approach to managing both simple and complex mappings between ontologies representing loosely coupled domains. OWL is extended to allow the specification of virtual properties whose values are derived functionally and not stored. These virtual properties can be used to express complex mappings between ontology terms.

In network management, dependency models have been used to support the correlation of events and alarms to an underlying root cause [Gruschke 1998, Kar et al. 2000, Brown et al. 2001]. In [Brown et al. 2001], a dynamic method to collect dependencies in a distributed system is described. The method requires active perturbation of the system and as such requires significant preparation to construct the dependency model. In [Kar et al. 2000], an approach for managing application services is described that enhanced existing network management infrastructure to cater for application service management. In this case a simple list of dependent resources is maintained. In each of the cases above, the dependency models, while simple, provide useful information to localise faults. Because the dependency models are now explicitly represented in a modelling language (but are part of management infrastructure), the potential for reasoning over and transformation of the model is reduced.

2.3.1.1 Analysis

From the review above, it can be seen that dependencies play an important role across a wide range of application domains. While each domain application above makes

specific use of dependencies, a number of common features appear with respect to what dependency is used for, how they are visualised and what level of formalism is used to represent the dependency model and relationships.

Usage of dependencies

The most common usage of dependency is to represent simple antecedent/dependant relationship between elements in a domain under study [Sangal et al. 2005, Ensel 2001]. In [Wang and Capretz 2009], it is proposed to reason over the ontology-based transitive dependencies relations. In [Deng et al. 2004, Bernstein and Melnik 2007], chains of dependent elements are constructed. The creation of chains of dependencies is also hinted at by Keller [Keller et al. 2000] as an advantage of the dependency analysis approach but the model does not provide ability to automatically build using chains other than using bespoke coding.

In [Drabble et al. 2009], the Event/Action concepts enable an innovative link between the dependency relationships of the domain and the event/actions that trigger those dependencies.

Visualisation of dependencies

A number of different forms are used to visualise dependencies. Dependencies are often represented in tabular form as seen in [Sangal et al. 2005, Borner and Paech 2009, Varol and Bayrak 2010, Wang and Capretz 2009, Maddox and Shin 2009]. Graphs are a common presentation format for dependency as seen in [Ensel 2001, Gruschke 1998, Ensel and Keller 2002, Varol and Bayrak 2010, Luo and Diao 2009, Drabble et al. 2009, Wang and Capretz 2009].

Formalisms

The formalisms used to represent dependencies vary greatly. Most approaches provide only simple representations for the dependency relationship [Sangal et al. 2005, Ensel 2001, Borner and Paech 2009, Varol and Bayrak 2010]. In most cases, especially [Luo and Diao 2009, Wang and Capretz 2009, Maddox and Shin 2009], the representation of dependency is very domain specific and it is difficult to see how it could be applied in the domain under study.

In [Ensel and Keller 2002], an RDF description of a simple dependency is described and uses the XML path language, XPath [XML Path Language], to carry out query on

the RDF documents. In [Drabble et al. 2009] an OWL model is provided to represent the domain and dependency model. These approaches provide the potential to carry out reasoning over the dependency relationships however this is only hinted at in this work and not discussed in detail.

In [Deng et al. 2004], the approach appears to support only dependency chains between properties of classes in the context of ontology to ontology mappings. It does this at the expense of adding extra semantics to the source ontologies and thus couples the dependency model and domain explication in one source.

The approach taken in this thesis provides an ontology-based dependency metamodel that provides formal semantics in OWL for the constructs related to dependency. The dependencies in the ontology-based dependency model in this work are used to model the dependency relationships between mappings, ontologies and data sources in an integration system. The dependency model is used to carry out an impact analysis of the mappings affected by a changing data source. The dependencies are represented using three graphical views that allow the user to examine increasing detail of the dependencies by navigating between the three views. The separation of the dependency metamodel from the domain model enables independent evolution of the metamodel and domain models. The compact nature of the metamodel and process (Design Chapter, Section 3.2.4) for building domain specific models enables its application in other domains. The ontological basis of the metamodel provides the formal semantic for the dependency relationships over which automated reasoning can be carried out (using ontological reasoners).

2.3.2 Models of Dependency

Keller [Keller et al. 2000] and Cox [Cox et al. 2001] attempt to define the fundamental parts of dependency so that they are not tied to any specific domain.

In [Keller et al. 2000], dependencies are formed between various components of a distributed system. The dependency relationship exists between components if one component requires another component to carry out its tasks.

To support the model of dependencies in this domain, a multidimensional space of dependency attributes were defined. As shown in Figure 2-1, six dimensions are defined by Keller that represent the characteristics of dependencies between components in the distributed system under analysis.

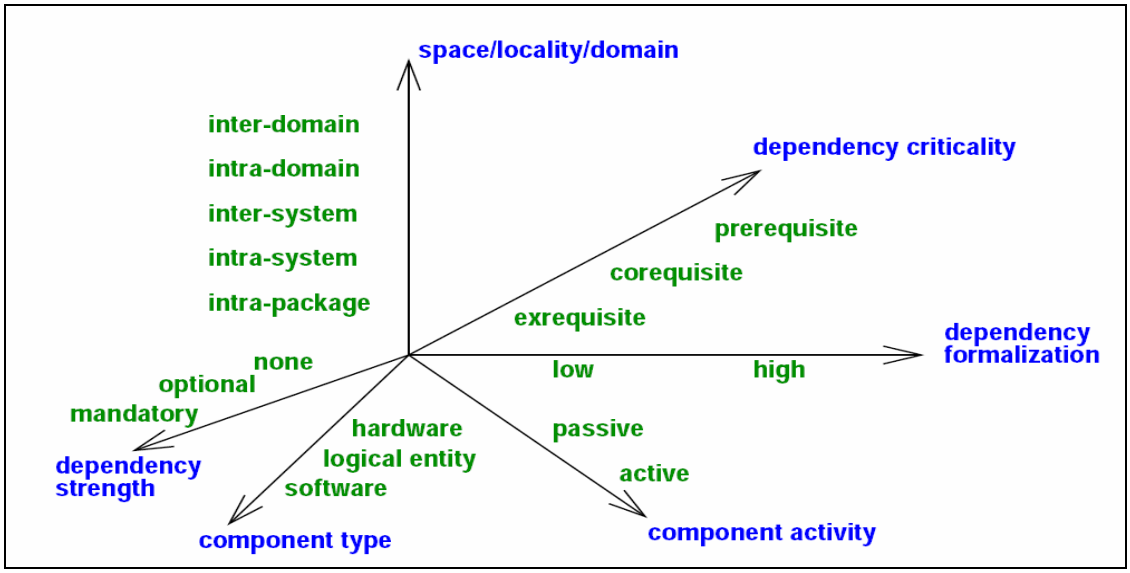

Figure 2-1: Keller's Multidimensional space of dependencies.

Using these attributes of dependency, two models of the dependencies in the service management domain were created. One model, called the Functional model by Keller, defined generic service dependencies (e.g. name service, database service). The other model, called the Structural model by Keller, contains detailed descriptions of the dependencies between the components that realise the broad services defined in the Functional model.

A technical realisation of the model, for example in UML or ontology-based is not provided.

In [Cox et al. 2001], Cox et al. attempt to formalise the definition and characterisation of dependencies in a unified approach. The approach taken is to identify and characterise the dependencies that exist between entities in a model of any domain. A dependency relation is defined by Cox and Delugach as a relation between a number of entities in the domain model, where it can be said that change to one of the entities implies a potential change to the other. Bidirectional and unidirectional dependency relations are defined. Cox and Delugach defined six dependencies attributes, selecting only two of the attributes defined by Keller, noting that six of the Keller dependency attributes are more suited as attributes of the system and not the dependency relation.

While Cox et al. illustrate their dependency model using two simple examples, it is clear again that a technical realisation of the model has not been created in any formal modelling language.

In [Drabble et al. 2009], an ontology-based approach is taken to the analysis of dependencies. A Node and Event/Action ontologies are defined. While the approach defined the ontologies in OWL, they are domain specific and focus on the domain description rather than the dependency description.

A number of high level description languages have been standardised in the IT systems management domain. The OSI General Relationship Model (GRM) [OSI GRM] offers a model for reasoning about, representing, managing and developing re-usable specifications for relationships between resources. While GRM defines a powerful generic model for defining relationships between managed objects and provides a mechanism for qualifying these relationships by means of attributes, it is tightly coupled with the OSI Structure of Management Information and CMISE and, thus, has not been used outside of TMN [ITU-T TMN] environments.

2.3.2.1 Analysis

The models proposed in [Keller et al. 2000] and [Cox et al. 2001] provide useful insight into the attributes and formalisation of dependency that are useful in the service management domain. The models have the advantages, as noted by Keller [Keller et al. 2000], that no modification of the application is needed if existing system configuration data can be used to populate the dependency model.

While both approaches provide a description of the dependency attributes, the core behaviour of the dependency relationship is not described and represents simple unidirectional or bidirectional relationships between antecedents and dependent elements. The creation of chains of dependencies is hinted at by Keller as an advantage of the dependency analysis approach but neither model provides ability to automatically build using chains other than using bespoke coding.

The ontology-based dependency modelling approach presented in this thesis (Section 3.2, Chapter 3), describes two different aspects of dependency attributes – i.e. behavioural attributes and descriptive attributes. While the descriptive attributes of the model are important, it is the behavioural attributes that enable the automatic reasoning over the ontology-based dependency model and thus provide the dependency analysis with the capability to automatically build chains of dependencies.

In [Drabble et al. 2009] a dependency analysis approach is described that uses SWRL rules [SWRL] to support “the mapping and additional deduction of information” in

collaborative environments. This is an interesting and useful addition to support the design of models of dependency, however it is unclear where and how the SWRL rules are applied.

In the approaches discussed above, the process to acquire instances to populate the dependency model is not explicitly specified and the approaches use bespoke coded solutions to acquire the instance data [Ensel and Keller 2002, Keller et al. 2000, Borner and Paech 2009, Drabble et al. 2009]. This makes any generalisation of the approaches difficult.

Derived Requirements

Based on the state of the review of dependency, the following requirements were derived for the design of a dependency model that could model and analyse dependency across more than one domain:

- Selection of the appropriate abstraction level to cater for a range of dependencies that might exist in different domains (e.g. inter system, inter domain and intra system).
- Selection of the method to support computation of dependencies (e.g. the ability to traverse the dependencies to the deepest level to enable full root cause analysis that is important for service management).
- Approach for extracting the domain or system knowledge about dependencies to inject into the dependency model.

2.4 State of the Art – Mapping Usage in Schema and Ontology Evolution

Schema mappings are used to support query rewriting and/or data transformations in data integration systems [Halevy et al. 2006, Lenzerini 2002]. Mappings also are a fundamental part of the approaches taken to schema and ontology evolution [Kondylakis et al. 2009, Noy and Klein 2002]. Therefore, it is important to understand if mappings management approaches taken in schema and ontology evolution are useful in the context of managing mapping dependencies.

In the context of information integration systems, there are similarities between the usage of schema mappings and ontology mappings, i.e. the mappings are used to support data transformations and/or query rewriting between schema or ontology representation of data sources [Kondylakis et al. 2009].

The approaches to schema and ontology mapping management are discussed in the context of the ontology-based mapping dependency management approach taken in this thesis. Note that, while there are similarities between schema and ontology evolution, there are also differences between the areas as noted in [Kondylakis et al. 2009, Noy and Klein 2002].As described by [Noy & Klein 2000] the differences arise from different usage paradigms and the presence of explicit semantics in ontologies. For example, because ontologies can be used as controlled vocabularies the results of a query over an ontology could include elements of the ontology itself (e.g. subclasses or super classes).

The first section reviews research on the management and evolution of database schema mappings (Section 2.4.1).

The second section reviews research on the management of ontologies and ontology mappings (alignments) (Section 2.4.2).

Each section starts with some basic definitions and a summary of the approaches to schema and ontology evolution.

2.4.1 Database Schema Evolution

Rahm and Berstein [Rahm and Bernstein 2006] define schema evolution as “the ability to change deployed schema, i.e. metadata structures formally describing complex artefacts such as databases, messages, application programs or workflows”. Schema

mappings are used in the evolution process to "describe relationships between data sources" [Yu and Popa 2005].

Kondylakis et al [Kondylakis et al. 2009] present a detailed review of the schema and ontology evolution. Schema evolution techniques can be classified as approaches based on mapping composition and approaches based on mapping adaptation. Approaches that use mapping composition attempt to evolve schema by composing successive schema mappings. Approaches that use mapping adaptation attempt to evolve schema by updating schema mappings every time a primitive change operation occurs to the schema. They cite a number of differences between changes in schema and ontologies that mean that the approaches used for schema evolution are not appropriate for ontology evolution.

From [Curino et al. 2008, Kondylakis et al. 2009], the most relevant current approaches to schema evolution are outlined briefly below:

- In [Ra 2005], the Program Independency Schema Evolution (PISE) methodology is described. The PISE methodology uses multiple views over the sample data to ensure that as new applications are added, existing and older applications can continue to access the older views without program modification.

- In [Cleve and Hainaut 2006], an approach to maintain consistency between the software applications and the database schema they access is proposed. The approach requires the propagation of three types of schema transformation (adding a schema entity, removing a schema entity, transformation database key types) to the applications that access the schema. Only the third transformation type allowed automatic update of the software application. The first and second transformation types are used to help the database programmer locate relevant program sections using pattern searching or dependency graphs. The dependency graph approach is not elaborated upon in this work.

- In [Bernstein and Melnik 2007], Bernstein proposed an extensive model management approach that seeks to provide lifecycle support for the mappings that are central to the resolution of the data programmability problem. The model management approach defines the semantics behind a

range of operators (compose, difference, merge and inverse) that can be applied to models such as database schema and schema mappings. This can be classified as a mapping composition approach.

- In [Noy and Klein 2002], a mapping evolution technique that uses mapping adaptation approach is described. This approach focuses on incrementally adapting mappings as the schemas evolve. The approach has developed a model for representation of the schema changes and an algorithm to rewrite the mappings based on the model of the schema changes.

Two the recent tools to support schema evolution are PRISM and Clio. These were selected because provide a comprehensive set of features for schema evolution based on the current state of the art.

The PRISM workbench [Curino et al. 2008] represents the evolution step in terms of Schema Modification Operators (SMO), an operational language that naturally captures the atomic operations used to evolve an existing schema. The SMO operators represent a detailed set of the "create", "update" and "delete" operations on schema elements (e.g. table, column).

An earlier project, called The Clio project [Miller et al. 2001], is a system for managing and facilitating the complex tasks of heterogeneous data transformation and integration. Clio consists of three components, the schema engine, the correspondence engine and the mapping engine. The schema engine is responsible for loading and verifying schemas. Given a pair of schemas, the correspondence engine generates and manages a set of candidate correspondences between the two schemas. The generated correspondences can be augmented, changed or rejected by a user using a graphical user interface through which users can draw value correspondences between attributes. The mapping engine supports the creation, evolution and maintenance of mappings between pairs of schemas. In Clio, a mapping is a set of queries from a source schema to a target schema that will translate source data into the form of the target schema. The mapping creation process is inherently interactive and incremental. Clio stores the current mapping within its knowledge base and allows users to extend and refine mappings one step at a time.

2.4.1.1 Analysis

While mappings play key roles in the approaches to schema evolution described above, however the management approaches taken for the maintenance of mappings is not dealt with. Furthermore, there are also the fundamental reasons why the approaches are not easily transferable to ontology evolution as described in Noy [Noy and Klein 2002] Among these fundamental differences is that ontologies themselves are data that can be reasoned over in a way that schemas cannot (e.g. a query on a database schema will usually result in a set of instance data, while a query on an ontology can result in both instance data and elements of the ontologies itself). Furthermore ontologies themselves incorporate explicit semantics of a domain which in the case of schema based system tend to be incorporated into the application itself.

In the context of ontology-based integration systems, these approaches to evolution may not be directly applicable due to the differences in both the usage and nature of mappings in the ontology domain. The expressive nature of ontology languages compared to the relational model makes it unclear if the SMO operators defined in [Curino et al. 2008] are relevant to the ontology domain. The process to ensure the consistency of the evolved ontology is also more complex due to the higher expressive nature of ontologies.

The process for schema mappings and schema evolution tends to be coupled and the lifecycle of each is not identified or managed separately. Given the open nature of usage of ontologies on the World Wide Web, the ontology mappings may well find reuse outside the original application domain and therefore would benefit from a defined management approach.

The formal semantics of ontology-based languages allow for the use of reasoning that can be used for consistency checking of evolved ontologies that is not possible without bespoke coding in the case of schemas.

The author of this thesis believes that, based on the evidence in [Noy and Klein 2002, Lenzerini 2002], the mappings in the ontology-based integrations systems are sufficiently different from the schema approaches that the mappings would benefit from an independent management approach. The ontology-based dependency modelling approach proposed in this thesis provides an approach for the management of mappings in the ontology-based integration domain that see the mappings as

fundamental parts of the integration system that need to be evolved when the data sources change.

2.4.2 Ontology and Mapping Management

Ongoing maintenance of the ontologies is also of critical importance for any industrial deployment of an ontology-based integration solution as noted in [Wache et al. 2001, Uschold and Gruniger 2004, Hepp et al. 2008].

Work by Doan and Halevy [Doan and Halevy 2005], also identified the maintenance problem but concede that "it has received relatively little attention".

Much research work has been carried out on process and tools to support the development, evolution and alignment of ontologies [Harth et al. 2004, NeOn 2005, Zablith 2009].

A comprehensive review of the state of the art in ontology management is presented in [Hepp et al. 2008]. The review covers ontology management tools, ontology evolution and ontology alignment in detail.

The following important ontology management infrastructures are discussed across a range of functionality as shown in Figure 2-2.

Comparison Ontology development tools

	Protégé OWL	Semantic Works	TopBraid Composer™	IODT	SWOOP	OntoStudio®
Primary Ontology Language[11]	OWL	OWL	OWL	OWL	OWL	F-Logic
View	Form Text	Form Text Graph	Form Text (UML-like) Graph	(UML-like) Graph	Browser-like	Forms
Platform	Java	.NET	Eclipse	Eclipse	Browser + Java	Eclipse
Supported Reasoner	Via DIG	None	Pellet, (built-in) Via DIG	RACER, Pellet	Pellet	OntoBroker
Repository	Files, RDBMS	Files	Files, RDBMS	RDF on RDBMS	Files	Files, RDBMS

Figure 2-2: Ontology Management Infrastructures [Hepp et al. 2008]

In [Hepp et al. 2008], it is noted that the current tools available for ontology management are "limited with respect to (i) lifecycle support (ii) collaborative

development of semantic applications (iii) Web integration and (iv) the cost-effective integration of heterogeneous components in large applications".

Lifecycle support is important in the context of the research in this thesis as the dependency modelling approach proposed here can support such dynamic lifecycles by providing insight into the dependencies in data integration systems.

The NeOn project [NeOn 2005] attempts to address these limitations. NeOn is a large European Research project developing an infrastructure and tools for large-scale semantic applications in distributed organisations. Within NeOn, a reference architecture of ontology management is under development that is capable of coping with dynamic and evolving environments.

The NeOn project has created an Eclipse [Eclipse] based ontology development and management toolkit – also called NeOn. The NeOn toolkit supports the addition of plug-ins through the Eclipse plug-in infrastructure.

The Evolva [Zablith 2009] methodology for ontology evolution proposes to support the evolution of ontologies covering both change management and adaptation of the ontology. An initial version of Evolva is available as a plug-in for the NeOn toolkit. In its current early implementation, the Evolva plug-in provides support for the evolution of the ontologies and not the mappings that may exist in the system.

In [Hepp et al. 2008], a lifecycle for ontology mapping (alignments) is described. The lifecycle notes that once an ontology changes, the alignments also need to change. It is noted that to date very few tools offer support for mapping management.

In the Data Information and Process Integration with Semantic Web Services Project, DIP [Harth et al. 2004], a review of ontology management was undertaken that comprised of ontology specification languages, ontology storage and retrieval, change management for support of evolving ontologies and devices for enabling access to ontology repositories. It is important to note that this work covers change management and versioning related to ontologies only and does not deal with the management and evolution of mappings.

KAON [KAON] is an ontology and semantic web tool suite from the University of Karlsruhe. KAON provides both ontology development and management functionality. In KAON the user is provided with capabilities to customise and control the process of ontology evolution as detailed in [Stojanovic 2002].

In [An and Topaloglou 2007], the challenges associated with the maintenance of mappings are described as:

- The maintenance of the consistency of mapping when the database or ontology changes.
- Application of changes in the ontologies and schema after updating the semantic mappings (i.e. if the mappings are updated, then the schema or ontologies may also need to be changed – this is sometimes referred to as round tripping).
- Systematic maintenance process.

An approach to maintenance of semantic mappings is proposed that defines the semantic mappings as conjunctive formulas that encode a sub-tree of the ontology. The mapping is essentially a formula that defines parts of the ontology (in terms of a graph) that are mappings to a schema element. The approach proposed is capable of updating the semantic mappings in the local-as-view examples presented. The approach does not provide any maintenance information to support the analysis of mappings that already use the ontology property or database table attribute that is subject to change and as such assumes the existence of a tool that will process and select the mappings that need to change.

2.4.2.1 Analysis

The development of an ontology is a complex process that spans much more than just the ontology development tools [O'Sullivan D. 2005, An and Topaloglou 2007, KAON, NeOn 2005, Hepp et al. 2008].

Much fruitful research has been carried out [Hepp et al. 2008] and excellent tools developed [NeOn].

The ongoing maintenance of the ontologies is also of critical importance for any industrial deployment of an ontology-based integration solution as noted in [Wache et al. 2001, Uschold and Gruniger 2004, Hepp et al. 2008]. The NeOn project [Hepp et al. 2008] provides an excellent, extensible framework for the development and management of ontologies.

The evolution of semantic mappings is still in its early stages as noted by [Hepp et al. 2008] because of the difficulty of the task as noted by [An and Topaloglou 2007].

The ontology-based dependency modelling approach proposed in this thesis can support the ontology alignment lifecycle proposed in [Hepp et al. 2008] by automatically providing the candidate mappings that are dependent on the part of the ontology that is evolving.

2.5 State of the Art - Ontology-based Integration Approaches

This section covers information integration approaches that use ontologies. In experiment one and two, a generalised ontology-based integration test bed was created to support the integration of heterogeneous data sources. This section describes how ontologies can be used in ontology-based integration systems.

Ontological approaches to integration are defined as approaches that use ontologies to formally define a shared domain and use mappings to create semantic links between these ontologies [Cruz and Xiao 2005, Noy 2004, Wache et al. 2001]. As noted in Section 2.4, mappings were also used to support schema and ontology evolution. In contrast, this section focuses on why and how ontologies are used in integration systems. Part of this analysis focuses on the usage of mappings to support integration. The three headings used by Wache et al [Wache et al. 2001], are used here to discuss how and why ontologies are used in integration systems. The headings are:

- Use of ontologies in Integration Systems.
- Ontology representation in Integration Systems.
- Use of mappings in Integration Systems.

These three headings provide suitable criteria to discuss the generalised ontology-based integration test bed used in this thesis because the integration test bed was designed based on these principles. A discussion of the generalised ontology-based integration test bed against these headings is contained in the respective analysis sections below.

(Note that Wache discussed a fourth heading, Ontology Engineering. This is not discussed in this research as the focus was on data integration approaches and not how the integration ontologies can be created. In this thesis, the Protégé ontology development tool [Protégé] has been used to create the integration ontologies used through out the experiments.)

Recent ontology-based integration systems [Wu et al. 2006, Zhou and Wang 2006 , Biffl et al. 2010, Beneventano et al. 2009, Kwak and Yong 2008, Fu et al. 2008, Cruz et al. 2004, Dong and Linpeng 2008] are discussed in Section 2.5.4 against these three headings. These were selected because they make instrumental usage of ontologies and

thus can be compared to the state of the art and to the generalised ontology-based integration test bed created in this research.

2.5.1 Use of Ontologies in Integration Systems

Nearly all integration systems that use ontologies employ them for the explicit description of information that is in the information sources managed by the integration systems. The most common definition of an ontology, from Gruber [Gruber 1993], is that an ontology represents a formal and explicit specification of shared conceptualisation. In [Noy 2004], Noy defined an ontology as a formal description of a domain of discourse.

In the integration context, the key usage of the ontology is to enable sharing of information across application domains by leveraging an ontologies ability to perform reasoning.

In [Cruz and Xiao 2005], Cruz and Xiao identify five uses of ontologies in data integration:

- **Metadata Representation.** Metadata (i.e. source schemas) in each data source can be explicitly represented by a local ontology, using a single language.
- **Global Conceptualisation.** The global ontology provides a conceptual view over the schematically heterogeneous source schemas.
- **Support for High-level Queries.** Given a high-level view of the sources, as provided by a global ontology, the user can formulate a query without specific knowledge of the different data sources. The query is then rewritten into queries over the sources, based on the semantic mappings between the global and local ontologies.
- **Declarative Mediation.** Query processing in a hybrid peer-to-peer system uses the global ontology as a declarative mediator for query rewriting between peers.
- **Mapping Support.** A thesaurus, formalised in terms of an ontology, can be used for the mapping process to facilitate its automation.

From [Wache et al. 2001, Cruz and Xiao 2005], three types of architecture have been identified for making use of ontologies a) single ontology approaches where a single global ontology represents all of the semantic of the underlying data sources, b) multiple ontology approaches where each data source is described by its own ontology

and c) hybrid ontology approaches where global and local sources ontologies are arranged in a hierarchy.

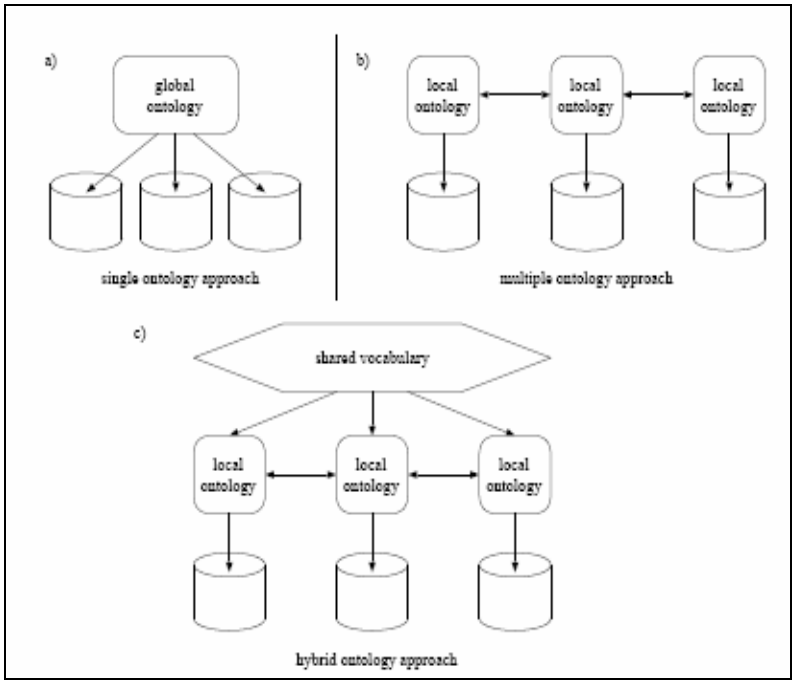

Figure 2-3: Three Ontology Approaches from [Wache et al. 2001]

Single Ontology Approach. All source schemas are directly related to a shared global ontology that provides a uniform interface to the user. However, this approach requires that all sources have nearly the same view on a domain, with the same level of granularity.

Multiple Ontology Approach. Each data source is described by its own (local) ontology separately. Instead of using a common ontology, local ontologies are mapped to each other. For this purpose, additional representation formalism is necessary for defining the inter-ontology mappings.

Hybrid Ontology Approach. A combination of the two preceding approaches is used. First, a local ontology is built for each source schema that is not mapped to other local ontologies, but to a global shared ontology. New sources can be easily added with no need to modify existing mappings between the data sources.

In [Uschold and Gruniger 2004], Uschold and Gruniger define “Common Access to Information” as one of the four main categories to apply ontologies. In this context, the

ontology avoids the need to create and maintain many translators while making it easier to introduce new systems and formats to the system. This is important because Bernstein [Bernstein and Melnik 2007, Bernstein and Haas 2008] indicates that significant costs, resulting from ongoing maintenance of the integration systems, can be encountered in enterprise integration projects.

2.5.1.1 Analysis

The hybrid ontology approach (Figure 2-3) provides some benefits over the other approaches. New data sources can be added by creating new source ontologies. The addition of new source ontologies is easier for the hybrid approach when the local ontology can adopt the Local-As-View approach. Local-As-View represents local schema in terms of the global schema. This makes the hybrid ontology approach more appropriate for building integration systems that provide a global or central representation of data.

However, in the hybrid approach, impedance mismatches between the data source ontologies and the shared ontology can arise. Impedance mismatches between ontologies can occur if the representation format or modelling granularity is different. These impedance mismatches can make mapping creation more difficult.

It is also worth noting that in integration systems the shared ontology may need to represent integration semantics as well as representing the shared domain. Integration semantics are formal definitions of knowledge that are used to support the integration process (e.g. integration goal, ontology versioning). These integration semantics may or may not sit within the main shared ontology.

The generalised ontology-based test bed in this thesis uses the hybrid ontology approach, when the global schema is used to represent the domain of interest. The local ontologies provided a common data model to represent the data sources. This approach is taken by several ontology-based systems that are discussed later in Section 2.5.4. In the case of the integration test bed in this thesis, the hybrid ontology approach also enabled a functional separation of the domain ontologies, mappings and lower ontologies. This was appropriate for the test bed because the domain ontologies for different domains could be easily swapped in because the integration test bed was developed so that the functional separation was maintained as shown in the design chapter (Section 3.3).

2.5.2 Ontology Representations in Integration Systems

Description logics supplemented by rules languages are now the popular approach for representing ontologies. However, some integration systems use frame based systems. A full treatment of the expressive power of these representation types is described in [Corcho and Gomez-Perez 2000] by Corcho and Gomez-Perez.

In [Cruz and Xiao 2005], Cruz and Xiao discuss the following ontology languages:

- **XML Schema.** Strictly speaking, XML Schema is a semantic mark-up language for Web data. The database-compatible data types supported by XML Schema provide a way to specify a hierarchical model. However, there are no explicit constructs for defining classes and properties in XML Schema, therefore ambiguities may arise when mapping an XML-based data model to a semantic model.
- **RDF and RDFS.** RDF (Resource Description Framework) is a data model developed by the W3C[8] for describing web resources. RDF allows for the specification of the semantics of data in a standardised, interoperable manner. In RDF, a pair of resources (nodes) connected by a property (edge) forms a statement: (resource, property, value). RDFS (RDF Schema)[9] is a language for describing vocabularies of RDF data in terms of primitives e.g. rdfs:Class, rdfs:Property, rdfs:domain and rdfs:range. Therefore, RDFS is used to define the semantic relationships between properties and resources.
- **DAML+OIL.** DAML+OIL (DARPA Agent Markup Language Ontology Interface Language) is a fully-fledged Web-based ontology language developed on top of RDFS. It features an XML-based syntax and a layered architecture. DAML+OIL provides modelling primitives commonly used in frame-based approaches to ontology engineering and formal semantics and reasoning support found in description logic approaches. It also integrates XML Schema data types for semantic interoperability in XML.
- **OWL.** OWL (Web Ontology Language) is a semantic mark up language for publishing and sharing ontologies on the Web. It is developed as a vocabulary extension of RDF and is derived from DAML+OIL.

[8] W3C. The World Wide Web Consortium is the standards organisation for web technologies.
[9] RDF Schema - http://www.w3.org/TR/rdf-schema/

A comprehensive review of the state of the art in ontology representation formalisms is presented reported in [Harth et al. 2004]. This additionally adds Topic Maps [Topic Maps] and the Unified Modelling Language (UML) [UML]. An ontology evaluation schema is developed and each ontology language discussed in that context.

- **Topic Maps.** A topic map consists of a collection of topics, each of which represents some concept. Topics are related to each other by associations, which are typed n-ary combinations of topics. A topic may also be related to any number of resources by its occurrences.
- **Unified Modelling Language** (UML) is a standardised specification language for object modelling that includes a graphical notation used to create an abstract model of a system, referred to as a UML model.

2.5.2.1 Analysis

For data integration systems, ontology languages provide two main advantages over schema based approaches. Ontologies provide significantly more expressive power than simpler XML schema. Additionally the formal semantics of the ontology representation has enabled many reasoning tools to be developed.

As OWL provides three variants (Full, Lite and DL), OWL's expressive flexibility is useful when describing both domain (e.g. supply chain information) and integration specific semantics (e.g. integration process). This is important because one representation language can model different aspects and thus reduce the impedance mismatch problem.

It is also important that the ontology language has a supporting query language to enable the integration application to extract knowledge from the ontology. SPARQL [SPARQL] is the predominant query language for ontologies. SPARQL is designed for querying RDF and adopts a triple format to match ontology instances. However, it does not have a natural ability to query basic ontology constructs (e.g. OWL object properties). OWL-QL [OWL-QL] supports query-answering dialogues in which the answering agent may use automated reasoning methods to derive answers to queries. OWL-QL is a candidate standard language and protocol for query-answering dialogues among Semantic Web computational agents using knowledge represented in OWL.

Ontologies were used in two places in the generalised ontology-based integration test bed used in the experiments in this thesis. A shared domain vocabulary (called the upper ontology in design chapter (Section 3.3.2)) was implemented in OWL-DL using the Protégé ontology development environment. The ontology provides a domain description in experiment one and two for the product line management and logistics domains respectively. The data sources in experiment one and two were represented as RDF descriptions – these are called the lower ontologies in the design chapter (Section 3.3.2). SPARQL was used as the query language, rather than OWL-QL, as SPARQL was fully integrated with the Jena OWL API [Jena]. The usage of OWL and SPARQL provided benefits for the generalised ontology-based integration test bed because OWL reasoning (using the Pellet reasoner [Pellet]) could be used to verify the correctness of the domain ontology created and SPARQL could be used to query both ontologies.

2.5.3 Mapping Usage in Integration Systems

In section 2.4, the usage of mappings to support ontology and schema evolution was discussed. This section focuses on how mappings are used within ontology-based integration system to support integration.

Mappings may serve to relate ontologies to other ontologies (inter-ontology mappings) or to relate ontologies to underlying information sources (e.g. a database) [Wache et al. 2001]. For integration systems, both types of mappings are needed. For ontology to information source mappings, there are a number of general approaches to mappings from [Wache et al. 2001]:

- Structure Resemblance: This approach simply converts the data source structure into the ontology language.
- Definition of Terms: This approach adds more semantics to the ontology that are not explicitly represented in the data source.
- Structure Enrichment: A combination of the two previously mentioned approaches.
- Meta-Annotation: This approach requires the addition of semantic information to the data source.

For ontology to ontology mappings, the general approaches from [Wache et al. 2001] are:

- Defined Mappings: Simple point to point and complex mappings can be defined by a user.
- Lexical Relations: A defined set of linguistic relationships can be applied to ontologies.
- Top-Level Grounding: Ontologies can map to a single top level ontology using common super classes.

Noy [Noy 2004] describes two major architectures for mapping discovery, the shared ontology approach and heuristics and machine learning based approaches. Using a shared global ontology facilitates easier mapping creation because the domain specific ontologies extend the global ontology and are thus grounded in a common vocabulary. Heuristics and machine learning based approaches typically allow for semi-automatic discovery of mappings by using features of the ontology such as class hierarchy or property definitions. Kalfoglou and Schorlemmer [Kalfoglou and Schorlemmer 2003] provide a comprehensive review of mapping techniques.

Once discovered, mappings themselves need to be represented and stored. Mappings are typically stored either within the ontology itself using its description language or externally using a defined mapping language. Noy [Noy 2004] describes several mapping representations such as bridging axioms [Dou et al. 2003] in first-order logic to represent transformations, using views [Calvanese et al. 2001] to describe mappings from a global ontology to local ontologies and mappings that are represented as instances of an ontology of mappings [Maedche et al. 2002].

Once mappings are created, they can be used to perform various integration tasks such as data transformation or query answering. Reasoning is used to perform these tasks and can be run over the ontology and/or the mappings. The OntoMerge [Dou et al. 2003] system uses reasoning over the ontology to perform several ontology translation tasks. Other tools [Crubezy and Musen 2003] process instances of the mappings to perform integration tasks.

Several approaches exist to automatically generate ontologies and mappings from databases. The D2RQ [D2RQ API] API is a declarative language to describe mappings between relational database schemata and OWL/RDFS ontologies. The D2RQ platform uses these mappings to enable applications to access an RDF view on a non-RDF database.

2.5.3.1 Analysis

Most ontology mappings (and matching) frameworks are semi-automatic because they require human and manual intervention to support the mapping process. This is especially true where complex conversions between structures are needed (e.g. converting quarterly revenue to monthly revenue). These complex mappings tend not to be discoverable in an automatic way. Thus, mapping generation requires strong tool sets to support both the creation and the evolution of mappings.

Another issue with the mapping approach is that the mappings themselves tend to create bindings (or dependencies) in the system from top level ontologies to bottom level data sources. This is especially true when the hybrid ontology approach is applied because there are multiple layers of dependency from the top level ontological concept through the mapping to the lowest level data source item. Furthermore, these dependencies can become more complex if there are multiple levels in the hierarchy of ontologies.

Tools and processes that support the evolution of mappings after development is finished are limited. In [Seidenberg and Rector 2006] methodologies for maintaining both simple and semantically complex mappings are presented. However, the maintenance model covers only ontology to ontology mappings and does not deal with dependencies as they manifest themselves in an ontology-based integration system.

Mappings were used in two places in the generalised ontology-based integration test bed used in the experiments in this thesis. The upper and lower ontologies of the generalised ontology-based test bed (Section 3.3.2) are connected using mappings based on the INRIA [Euzenat 2004] mapping format. These represent ontology to ontology mappings and are used to support rewriting of the queries against the upper ontology. These mappings were created manually because the mapping creation process requires in depth knowledge of the domain ontology and the data source to ensure the appropriate mappings are created.

The lower ontologies of the generalised ontology-based test bed are connected to the data sources using mappings provided by the D2RQ API [D2RQ API]. These represent ontology to data source type mappings and are also used to support rewriting of SPARQL queries to SQL queries on the data sources. These mappings were created automatically by the D2RQ API from the data sources.

2.5.4 Implementations of Ontology-based integration Systems

This section reviews recent implementations of ontology-based integration systems based on their architecture, usage of ontologies and mappings. The approaches were selected because they make instrumental use of ontologies to enable integration and are this likely to take advantage of latest research on ontology-based integration. These approaches are compared to the generalised ontology-based test bed created in this research in the analysis Section 2.5.4.1.

In [Zhou and Wang 2006], Zhou and Wang propose a semantic grid architecture for enterprise information integration. The authors state that the architecture requires the "convergence of peer-to-peer, grid and semantic computing". The information integration system proposed is comprised of Data Peers (DP), Semantic Peers (SP) and Applications Peers (AP). Each peer has a schema describing the data held by that peer and a set of mappings that specify relationships with the data exported by other peers. The approach does not clearly fit into the three architectures defined by Wache in [Wache et al. 2001] but could be considered a hybrid approach given semantic descriptions at different levels in the system. The approach does not have a global or mediated schema.

The approach makes use of four kinds of mapping (i.e. DP-SP, SP-SP, AP-AP, SP-AP). The mappings are used to support query rewriting against queries that can be issued at the DP, SP or AP. The mappings and schema representations for the data peers are encoded using WSML [WSML]. The mappings are created at design time using the WSMT [WSMT] tool.

The DartGrid system [Wu et al. 2006], proposed by Wu et al., provides a framework for integrating heterogeneous relational databases. The framework includes tools for mapping creation, ontology query and search. DartGrid follows the single ontology approach because it contains a global ontology, in RDF, and mappings to ontology representations of the relational data sources. The mappings are used to define relationships between the global ontology and relational schemas. The mappings are used to support rewriting of queries issued in SPARQL [SPARQL] against the global ontology to SQL queries against the data sources. The representation of mappings is not discussed but the mappings contain information about which database tables and properties are mapped to which RDF class in the global ontology. A tool is provided to support the development of the mappings at design time. The approaches taken to

support the management and evolution of the system (including the mappings) are not discussed.

Biffl et al [Biffl et al. 2010] introduce a framework for the semantic integration of data sources related to the management of the software development lifecycle (e.g. bug tracking systems). The framework adopts the single ontology approach because it provides a single domain conceptualisation that represents three data sources (bug tracker, code management and a mailing list). The domain ontology is created in OWL using Protégé. The framework is implemented as a Java application that uses the Jena API to load and process the domain ontology. The system does not have a formal mapping approach but uses bespoke coded adaptors to extract data from the underlying data sources and populate the ontology instances. In this framework, the mappings are used to support the populating of the ontology instances. As this system is at an early stage of development, the approaches taken to support the management and evolution of the system (including the mappings) are not discussed.

The MOMIN-STASIS system, described by Beneventano et al [Beneventano et al. 2009], is an ontology-based integration system (called MOMIS-STASIS) that combines the two previous works - MOMIS [Beneventano et al. 2003] and STASIS [Abels et al. 2008]. The MOMIS data integration system uses a single ontology approach where WordNet is used as the shared vocabulary for the specification of the semantics of the data sources. The STASIS system is a general framework to simplify the mapping creation process between different schemas. The combined output of the MOMIS-STASIS system is a global schema and a set of mappings that relate the global schema to the data sources. The advantages of this approach are that global schema can be generated automatically. To support this approach, the local sources need to be annotated using simple name matching techniques. The mapping format in the system supports simple and complex relationships (e.g. equivalence, more general, less general and disjointness). The mappings are used to support the generation of the global schema.

The approach taken for semantic integration in [Kwak and Yong 2008] describes how a global ontology is used to integrate data sources from the automotive parts industry. The domain ontology is mapped to the data source using three types of mapping (equivalence, subclass/superclass, and disjointness). Although there are several ontologies defined to represent automotive parts, the approach represents the single

ontology approach as the ontologies are essentially a single domain description. The ontologies are represented in OWL. The mappings in the system are used to convert instances of the data sources to instance of the ontology; however the mappings are limited to 1-1 relationships between the data sources and global ontology. The authors cite the management of the mappings as future work.

In [Fu et al. 2008], the hybrid ontology approach is adopted to provide an integration framework for E-business and Logistics systems. The global ontology represents a conceptualisation of the domain that relies on the usage of lexicons (e.g. ebXML[10], WordNet[11]) and upper ontologies (e.g. BULO[12]). The data sources are described using local ontologies. The local ontologies are generated using a set of rules to convert aspects of the data sources into ontologies in OWL (e.g. database attributes are mapped to OWL datatype properties). Mappings are also used to establish relationships between the local ontologies and global ontologies. The mappings are used to support the translations of SPARQL queries on the global ontology to SQL queries on the data sources. The format of the mappings used to relate the global schema to local schema is not discussed.

Cruz et al describes an approach to the integration of XML schema using ontologies in [Cruz et al. 2004]. The approach adopts the hybrid ontology approach where a global ontology is created in RDF that is created by merging RDF representations (the local ontologies) of the data sources. The RDF vocabulary was extended by the addition of a "contains" property to support the representation of nesting structure of XML documents. During the merging process, mappings are used relate the global and local ontologies. The mappings are used to support translation of queries directed at the global ontology to queries over the local ontologies and are maintained in a mapping table by integration system for use later when performing query translation. Mappings are generated at design time using the PROMPT [Noy and Musen 2000] system.

A similar approach is proposed, by Dong and Linpeng [Dong and Linpeng 2008], to integrate XML schemas. This work defined thirteen heuristic rules to enable the conversion of the XML schemas to the RDF local ontologies. A further heuristic eight rules are defined to create a global ontology (in OWL) from the local ontologies. Again,

[10] Electronic Business using XML, ebXML. http://www.ebxml.org/
[11] WordNet. A lexical database for English. http://wordnet.princeton.edu/
[12] Base Upper Level Ontology. http://proton.semanticweb.org/

the system maintains a mapping table that is used populate data instances in the global ontologies. This differs from other approaches that do not store the data at the global ontology layer.

2.5.4.1 Analysis

The ontology-based integration systems described demonstrate the broad range applications areas. The approaches follow both the single and hybrid ontology approach described in the state of the art (section 2.5.1). Mappings are also used widely in the approaches to provide query rewriting [Wu et al. 2006, Cruz et al. 2004, Fu et al. 2008]. Many different and bespoke mapping representations (e.g. XML, MOMIS-STATIS [Abels et al. 2008]) are used and therefore there is little consensus on the best mapping representations. This lack of standardisation for mappings was also noted in [Hendrik et al 2009].

It is also clear that while each approach has focused on the delivering an ontology-based solution, the "integration quality" of the systems is not measured. Similarly the approaches taken to manage the mappings in the integration systems are not described.

The generalised ontology-based integration test bed that was used in this thesis adopts the hybrid ontology approach. The approach created a functional decomposition of the domain conceptualisation, mappings and local ontologies. This approach was selected as the most appropriate for the development of the test bed because different local data sources from entirely different domains (e.g. sales in experiment one and logistics in experiment two) was tested with system. The global ontology was developed in OWL because the expressivity of OWL ensured that a wide range of domains can be represented. The mappings used in test bed provide extra functionality that is not present in any of the systems described above. The transformations function can be assigned to each mapping to carry out instance level transformation. This was a requirement for the use cases used in the experiments in this thesis – because simple integrated views (without transformation) of heterogeneous data would need further development work by the application consuming the views to transform the data.

It is important to note that the test system is not a fully featured integration system as discussed in Section 3.3. In spite of this, the architecture of the test bed mirrors many of the design approaches taken in the systems described above.

2.6 Summary Analysis

In this review, it was identified how and why ontologies can be used in integration systems. It was shown that there are several options for ontology representation and that OWL [OWL] is a suitable candidate due to its flexible and formal semantics.

While mappings are a fundamental part of non ontology-based integration systems, the approaches taken for the evolution of schemas and mappings are different enough to make them difficult to apply in the ontology-based integration domain [Kondylakis et al. 2009, Noy and Klein 2002].

The lack of support for maintenance of the schemas and mappings after deployment was identified as a fundamental weakness of all integration systems that use mappings to create semantic links between entities in the systems [Bernstein and Melnik 2007, Haas 2007, Kondylakis et al. 2009]. In [Doan and Halevy 2005], Doan and Halevy identify the maintenance problem but concede that "it has received relatively little attention".

The author of this thesis believes that the maintenance of mappings is at least as important as their initial construction in any industrial context.

The ontology-based dependency modelling approach proposed in this thesis provides the framework to support the independent management of semantic mappings by modelling the dependencies they exhibit. The author believes that this is a key first step in the management and evolution of the mappings in ontology-based integration systems. While some of the current implementations of ontology-based integrations systems recognise this problem, most have not developed approaches towards a resolution of the problem.

Dependencies and dependency analysis has been used across many domains [Borner and Paech 2009, Varol and Bayrak 2010, Luo and Diao 2009, Drabble et al. 2009, Wang and Capretz 2009, Maddox and Shin 2009]. Only a few approaches provide formal representation of dependency that can be used to reason about. Most representations of dependency are based on simple notions of dependency without any behaviour aspects modelled as in the approach taken in this thesis.

This work distinguishes itself by modelling, and thus making explicit, the dependencies that occur in the generalised ontology-based integration system between the mappings ontologies and data sources. This explication of dependencies

compliments existing ontology integration techniques because it enables more flexible approaches to the maintenance and scalability of the key knowledge assets of the integration system (i.e. mappings and ontologies).

The compact ontology-based dependency metamodel provided an excellent basis to construct the domain specific model. The formal semantics associated with the dependency model enabled the automation of the computation of chains of dependent elements.

2.7 Background Design Choices

This section describes the choices taken for the tools and technologies that were used to implement and test the ontology-based dependency management approach proposed in this thesis.

2.7.1 Measuring "Integration Quality": THALIA Integration Benchmark

The research question for this thesis required the measurement of the ability of the ontology-based test bed to carry out integration over heterogeneous data sources. This has been defined as the "Integration Quality" metric in the introduction (Section 1.2) and was measured using the THALIA (**T**est **H**arness for the **A**ssessment of **L**egacy information **I**ntegration **A**pproaches) integration benchmark.

THALIA is a publicly available and independently developed test bed and benchmark for testing and evaluating integration technologies [Stonebraker 2005]. The system provides researchers and practitioners with downloadable data sources that provide a rich source of syntactic and semantic heterogeneities. In addition, the system provides a set of twelve benchmark queries for ranking the ability of an integration system to carry out integrations across a wide range of heterogeneities.

A score out of twelve can be assigned to an integration system based on how many of the 12 THALIA tests the system can integrate successfully.

The 12 tests are summarised below:

Table 2-1 THALIA Tests

Test No.	Name	Description
Test 1	Synonyms	Attributes with different names that convey the same meaning

Test 2	Simple Mapping	Related attributes in different schemas differ by a mathematical transformation of their values. (E.g. Euros to Dollars)
Test 3	Union Types	Attributes in different schemas use different data types to represent the same information.
Test 4	Complex Mapping	Related attributes differ by a complex transformation of their values.
Test 5	Language Expression	Names or values of identical attributes are expressed in different languages.
Test 6	Nulls	The attribute value does not exist in one schema but exists in the other
Test 7	Virtual Columns	Information that explicitly provided in one schema is only implicitly available in the other schema.
Test 8	Semantic Incompatibility	A real-world concept that is modelled by an attribute does not exist in the other schema
Test 9	Same Attributes exist in different structures	The same or related attributes may be located in different position in different schemas.
Test 10	Handling Sets	A set of values is represented using a single, set-valued attribute in one schema vs. a collection of single-valued hierarchical attributes in another schema
Test 11	Attribute name does not reflect semantics	The name does not adequately describe the meaning of the value that is stored.
Test 12	Attribute composition	The same information can be represented either by a single attribute or by a set of attributes

As THALIA provided only a score out of twelve, for this research the THALIA system was extended by introducing an effort classification system so that each query result in THALIA could be assigned an effort estimate based on how automatic the solution is.

Efforts are categorised as follows:

- Fully automatic: no code, mapping or ontology changes needed.
- Automatic: Automatic regeneration of ontology or other configuration artefact.
- Semi Automatic: A mapping needs to be changed manually.
- Manual: Non core code artefact needs to be updated or added manually. (e.g. a function associated with a mapping)

- Fail: core code changes needed. (e.g. core test bed code needs to be changed)

These effort classifications are specific to the ontology-based integration test bed defined earlier in this chapter.

The THALIA system also provides dataset that can be used to provide test integration systems. In this thesis, a comprehensive industrial data set was used because it already contained nine of the twelve test data. To allow the full THALIA suite to be run, the databases were supplemented by additional complexity in three areas (language expression and virtual columns, nulls – see Table 2-1, Chapter 3). This was achieved by adding specific data items to databases to cover these tests.

2.7.2 Supporting Technology Choices

This section provides a brief summary of the supporting technologies that were used in the construction of the metamodel, domain specific model and implementations of the TomE (Ontology-based dependency modelling tool, Section 3.2.6) and HotFusion (Generalised Ontology-based integration test bed, Section 3.3) tools.

2.7.2.1 Ontology Representation

The Web Ontology Language (OWL) was chosen as the ontology language to represent the ontology-based dependency metamodel and dependency model. OWL is a family of knowledge representation languages for authoring ontologies, and is endorsed by the World Wide Web Consortium. In this research OWL was used extensively to create the dependency model, metamodel and integration ontologies. The formal semantics that underpin OWL enable the reasoning over the ontologies using OWL reasoners such as Pellet. The OWL-DL subset was used throughout this research from the W3C Recommendation 10 February 2004.

2.7.2.2 Ontology Development APIs

The Jena API [Jena] was chosen as the development API to support the development of OWL based ontologies in the dependency model. Jena (from Hewlett Packard) is a Java framework for building Semantic Web applications. It provides a programmatic environment for RDF, RDFS and OWL, SPARQL and includes a rule-based inference engine. Jena was used in the generalised ontology-based integration system and the dependency modelling tools. In both these tools, Jena provided instantiation and query

operation and reasoning over the integration and dependency ontologies respectively. Version 2.0 of Jena was used in this research.

The D2RQ API [D2RQ API] was chosen to lift the relational data to the ontological level. D2RQ is a declarative language to describe mappings between relational database schemata and OWL/RDFS ontologies. This allows for automatic generation of the ontologies from the databases and once instantiated in a JENA model, the ontologies can be queried using SPARQL. The D2RQ API automatically converts the SPARQL queries to SQL and returns a set of triples to the caller. The API is used in the generalised ontology-based integration system. Version 0.5 of D2RQ was used in this research.

2.7.2.3 Ontology Reasoning

Pellet [Pellet] was selected as the ontological reasoner for this work. Pellet provides reasoning services for OWL ontologies. Pellet has been used to provide reasoning of the dependency model both programmatically using the Jena toolkit and also using the DIG interface from Protégé. Version 2.0.0-rc4 of Pellet was used in this research.

2.7.2.4 Ontology Editor

Protégé [Protégé] was chosen as the development environment for building ontologies. Protégé is a free, open source ontology editor. Protégé was used extensively to develop and test both the integration ontologies and the dependency models. Version 3.2 of Protege was used in this research.

2.7.2.5 Dependency Graph Visualisation

GraphML [GraphML] was chosen to provide serialisation of dependency graphs. GraphML is a comprehensive and easy-to-use file format for graphs. It consists of a language core to describe the structural properties of a graph and a flexible extension mechanism to add application-specific data. The GraphML format is used to represent the dependency graphs that are generated by the dependency model. Version 1.0 of GraphML was used in this research.

Prefuse is a set of software tools for creating rich interactive data visualisations. The original Prefuse toolkit provides a visualisation framework for the Java programming language. The Prefuse Flare toolkit provides visualisation and animation tools for ActionScript and the Adobe Flash Player. The Prefuse Java API was used to build the

visualisation of the dependency graphs in the TomE tool. Version 2007.10.21 of Prefuse was used in this research.

2.7.2.6 User Interface Development

CloudGarden's Jigloo GUI Builder was chosen as the user interface design tool. It provides a plug-in for the Eclipse Java IDE and WebSphere Studio that enables the building and management of both Swing and SWT GUI classes. This API is used to render and manage the user interface for both the generalised ontology-based integration system and the dependency modelling tool. Version 4.2.0 of Jigloo was used in this research.

2.7.2.7 XML Processing API (for mapping files)

SAX (Simple API for XML) [SAX] was chosen as the serial access parser API for XML. SAX provides a mechanism for reading data from an XML document. The SAX API is used for reading and processing the XML mappings in the generalised ontology-based integration system and the dependency modelling tool in this research.

2.7.2.8 Statistical Analysis Package

R [R] was chosen the statistical package to carry out statistical analysis of the data for Experiment Three (The performance of the ontology-based dependency model). R is a language and environment for statistical computing and graphics. It is a GNU project that is similar to the S language and environment that was developed at Bell Laboratories (formerly AT&T, then Lucent Technologies, now Alcatel-Lucent) by John Chambers and colleagues. R provides a wide variety of statistical and graphical techniques. R was used in Experiment Three.

2.8 Summary

This chapter has reviewed the state of the art dependency modelling and analysis, schema and ontology evolution and ontology-based approaches to information integration. It also provided background information for the tools and APIs used to design and implement the dependency models and tools which are implemented in Chapter 3.

3 DESIGN AND IMPLEMENTATION

3.1 Introduction

This chapter describes the design and implementation of the models, tools and test beds that were created to support the dependency modelling approach taken in this research. The dependency modelling approach enabled the analysis of the dependencies in a generalised ontology-based integration system.

A compact ontology-based dependency metamodel was designed to support the construction of a domain specific ontology-based dependency model (OBDM).

The OBDM was applied to the mappings from a generalised ontology-based integration test bed. The OBDM was implemented in a tool, called TomE (Towards Ontology Mapping Evolution), which uses ontological reasoning over the OBDM to build views of the dependencies in the system. The reasoning uses the Pellet OWL reasoner [Pellet].

The rest of this chapter is organised as described below. The design considerations for dependency analysis and the dependency metamodel are described in Sections 3.2.1, 3.2.2 and 3.2.3. The domain specific ontology-based dependency model (OBDM) is described in Section 3.2.5. The TomE tool that was created to support the OBDM is described in Section 0. Section 3.3 describes the generalised ontology-based integration test bed that was used in Experiment one.

3.2 Dependency Model Design

3.2.1 Design considerations for Dependency Analysis

It is important to consider the identification and analysis of dependencies in the design of enterprise systems as they become more logically integrated but physically distributed [Keller et al. 2000, Cox et al. 2001].

Systems can be logically integrated using such technologies as data federation, distributed query or ontology-based approaches with the underlying data and systems remaining physically separate.

Dependency analysis approaches have been used to support fault and event management [Gruschke 1998, Katker and Paterok 1997], service management [Dreo Rodosek and Lewis 2001, Varol and Bayrak 2010] and software configuration management [Luo and Diao 2009, Borner and Paech 2009]. Dependency analysis provides utility in each of these areas of application. However, the dependencies that exist between these domains and within these domains tend to be defined implicitly as noted by Keller in [Keller et al. 2000]. Furthermore, the analysis of dependencies in these areas has generally focused on inter-system dependencies.

In the context of ontology-based integration systems, the author of this thesis believes that dependencies between components of a single system are also useful to understand as they can support analysis and evolution of the system. If the dependencies are explicitly defined, then they can be reasoned over to provide useful insight in the evolution of the system. This was the approach taken in the dependency analysis tool (called TomE) that was built to support dependency analysis of the mappings in the generalised ontology-based integration domain that was used in experiments one and two.

While the dependency model proposed in this work focuses on dependencies within a single system of interest (i.e. ontology-based integration system), the approach can be applied in other domains as shown in experiment five.

Following a the literature review of the application and usage of dependencies (Section 2.3) in the service management, fault isolation and software configuration domains, the key requirements for the design of a more general dependency analysis system were summarised as:

- Selection of the appropriate abstraction level to cater for a range of dependencies that might exist in different domains (e.g. inter system, inter domain and intra system). [Requirement 1]
- Selection of the method to support computation of dependencies (e.g. the ability to traverse the dependencies to the deepest level to enable full root cause analysis that is important for service management [Keller et al. 2000].) [Requirement 2]
- Approach for extracting the domain or system knowledge about dependencies to inject into the dependency model. [Requirement 3]

The approach taken in this thesis for requirement 1 and 2 was to define a compact ontology-based metamodel for representing dependencies and a process to construct domain specific models from the metamodel. A metamodel provides an extensible set of concepts to enable the creation of domain specific models. The dependency metamodel in this research provides the basic building blocks needed to simplify the computation of dependencies (Requirement 2).

A compact metamodel has the following advantages:

- Domain specific models can be constructed using all or part of the metamodel affording design flexibility.
- Domain specific models can inherit key behaviours from the metamodel (e.g. transitive relations) enabling reuse of key metamodel features.
- Metamodel and domain model can be evolved independently.
- The approach is non-intrusive as the system under test does not require code updates because the dependency metamodel is external to the system under test.

A number of disadvantages for the metamodelling approach are:

- Difficultly in selected the appropriate abstraction level. (E.g. a very abstract metamodel can be difficult for domain modellers to understand while low abstraction level may not properly model all domains.
- A maintenance process to ensure controlled evolution for updates to the metamodel and domain specific model may be required.

The approach taken for requirement 3 required the definition of a domain decomposition process to enable the domain and system knowledge to be represented in the dependency model, and this process is described in section 3.2.4.

3.2.2 Dependency Abstractions used in the metamodel

Based on the state of the art review in Section 2.3.2, the abstractions for the dependency model in this work are based upon the following key ideas: dependent relations, dependent elements, simple dependencies, dependency chains and dependency graphs. This classification was selected following the state of the art review of the usage of dependency models and the previous attempts to formalise dependency relations (Section 0.)

These abstractions were designed to address the first and second design requirements and are described below.

Dependent Relations & Dependent Elements

The central concept of the model is the dependency relation. The dependency relations are classical binary relations [Fraissé 1986] between dependent elements. Dependency elements are representations of the entities in the domain under study that exhibit dependency relationships. Dependent elements can be derived from domain descriptions such as functional or design specifications or from experts in the domain (e.g. in experiment two on ontology-based integration systems presented later, dependent elements are derived from the descriptions of the systems semantic mappings.) In its most general sense, a dependent relation D is defined as an ordered triple (S^1, S^2, G) where:

D: Binary relation ("depends on")
S^1: Set of domain elements
S^2: Set of codomain elements
G: Subset of Cartesian product of S^1 and S^2

Behavioural & Descriptive Attributes of dependencies

Dependency relations can be defined with different attributes that describes the behaviour of the dependency relation (called behavioural attributes) or some descriptive information (called descriptive attributes) about the dependency relation. Behaviour attributes are used to represent the behaviour of the dependency relation

such as transitivity, symmetry or functional as described later in the ontology-based dependency metamodel.

Dependencies may also have descriptive attributes associated with them. Keller [Keller et al. 2000] defines a classification of dependencies based upon six descriptive attributes that are informed by the context of dependency analysis in enterprise service management. Descriptive attributes are used to describe some information aspect related to the dependency relation such as the importance or strength of the dependency as described later in the ontology-based dependency metamodel.

Simple Dependencies & Dependency Chains

In this work, a simple dependency is defined as a pair wise dependent relation between two nodes resulting from the decomposition process. A dependency chain can be created by traversing the dependency relations to join multiple dependencies together by following a single relation type appropriately (e.g. transitive relation).

For example, assume element (O) depends on an element (M) via the dependent relation R and the element (M) depends on element (D) via dependent relation R':

$$O \rightarrow M \text{ (via R that is transitive)}$$

$$M \rightarrow D \text{ (via R' that is transitive)}$$

If the dependent relations (R and R') are transitive, then the dependency model can be traversed to build the full dependency chain:

$$O \rightarrow M \rightarrow D$$

The depth of the dependency chain (i.e. the number of elements in the chain) can be computed by simply iterating a counter for each dependency relation found and assigning it to that relation. This introduces a quantitative measure of direct and indirect dependency relations. This is represented by a "Strength" attribute discussed later.

The type of dependency can also be handled in a similar way. The type of dependency can be seen as identification of the cause of the dependency relation (later in the evaluation chapter we will see overlapping and function-based dependency types). This is represented by a "Cause" attribute discussed later.

Dependency Graphs

Dependency graphs are used to visualise the dependency chains as illustrated in Figure 3-1. The nodes (vertices) of the graph represent the instances of the sets of elements in the domain of the system under analysis. The edges represent the dependency relations as described earlier. The graph represents the full dependency analysis of the domain of interest and is computed by following the behaviour attributes of the dependency relations. The graph is labelled with appropriate metadata to provide a description of the domain, graph type and version.

All of the concepts described above are visualised in Figure 3-1. The figure shows two dependency chains. The first dependency chain is for a UE (Upper Entity) called "UE-carriers-names". This dependency chain has a MP (Mapping Point) called "MP-c1" and GEs (Ground Entities) called "GE-exp_test_db2-logistics-Awards" and GE-exp_test_db2-Logistics-Awards". The UEs represent entities in the domain ontology of the integration system, the MPs represent mappings in the domain ontology and the GEs represented the datasources entities which are accessed by the integration system.

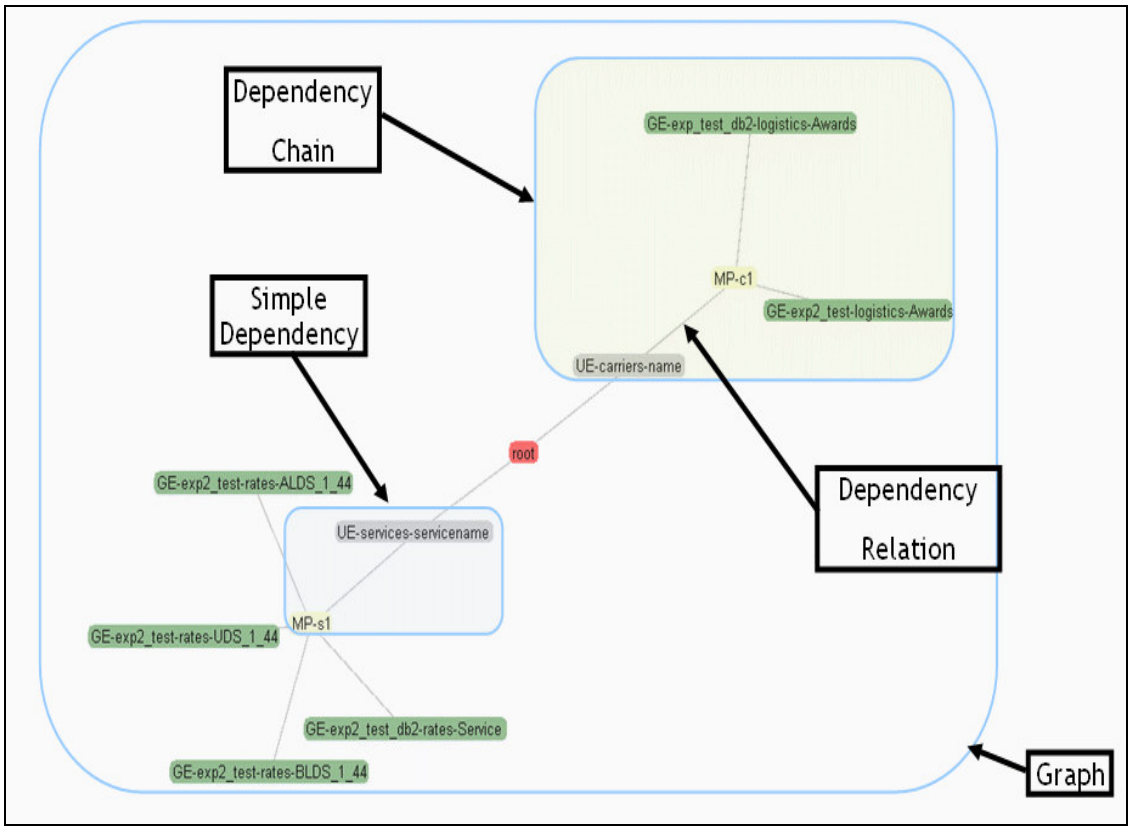

Figure 3-1: Illustration of Graph, Dependency and Dependency Chain

Benefits of the abstraction level selected

The dependency abstractions selected in this research are domain independent and general enough to be easily described to prospective users of the system. This enables them to be applied in a variety of domains. The compact nature of the abstractions facilitates easy adoption of the approach because the learning curve to understand the abstractions is small. The approach of supporting the dependency relations with formal semantics means that the domain dependency modeller can use the formal semantics to automate the computation of dependency chains using ontological reasoning as shown in Section 3.2.3 where chains of dependency elements are automatically created using the dependency model and the Pellet reasoner [Pellet].

3.2.3 Dependency Metamodel Design

This section describes the dependency metamodel that was created using the abstractions described in section 3.2.2.

The dependency metamodel provides an extensible set of concepts related to modelling of dependencies. This extensible set of concepts enables the creation of a wide variety of domain specific dependency models. For example, the metamodel can be used to represent dependencies between software components by describing abstraction representations of the components and the relationships between components. The metamodel provides a palette of attributes for dependency relationships (e.g. symmetric, transitive) that can be customised to the domain of interest.

The metamodel was realised in OWL-DL [OWL]. OWL-DL was selected because it is a dialect of OWL which provides maximum expressiveness without losing computational completeness. In the context of the dependency metamodel, OWL-DL provides the formal semantics for the dependency relations over which reasoning can occur using ontological reasoning. This removes the need to carry out some complex programming tasks to compute chains of dependent items – this task is passed to the reasoner and inferred from the semantics of the model. The metamodel was designed using the Protégé ontology design tool [Protégé].

The metamodelling approach provides a compact solution because the metamodel needs only to focus on the core aspects of dependency and not domain specific items. The compact nature of the metamodel allows easier creation of domain dependency models because the learning curve to understanding the abstractions is small. The metamodelling approach also enables the domain specific model to remain decoupled from the metamodel and thus allows additions to the metamodel to be made without affecting the domain specific model.

The key concepts of the metamodel, based on general conceptualisation of the dependency abstractions that was described in Section 3.2.2, are described below:

Architectural Entity: The concept is used to represent the dependent elements described in Section 3.2.2. An Architectural Entity is a concept that represents the nodes or elements in the system under study that exhibit dependencies. The domain

under study is composed of these architectural entities. The process of selection of the architectural entities for any given domain is carried out only when the domain specific model needs to be created. In the ontology-based metamodel, this concept is encoded as an OWL class with the following datatype properties:

- **Id [Mandatory]:** This is a property to represent the name of the entity. This is represented in the metamodel as an "rdf:ID" when the concept is created.

```
<owl:Class rdf:ID="UE">
<rdfs:subClassOf rdf:resource="#ArchitecturalEntities"/>
</owl:Class>
```

Code 1: Architectural Entity for a concept called UE with instance called "UE1"

- **Type [Optional]:** This property is used to represent the type of the entity. This is encoded in the metamodel as a datatype property of type "String". In the metamodel, the value range for this property is unrestricted (i.e. it can take any string value). The Type attribute can be used to specify any domain specific grouping or information that could be used to distinguish between forms of architectural entities.

Dependency Relation Attributes: These concepts are used to represent the dependency relations that are supported by the metamodel. A set of dependency relations are provided by the metamodel that represent transitive, symmetric and functional dependant relations between architectural entities. These represent the behavioural attributes described in Section 3.2.2 and illustrated in Figure 3-2. The definition of each relation is given below:

- **Transitive relation [Optional]:** A transitive relation implies that if X has a transitive relation with Y and Y has a transitive relation with Z, then X and Z also have the transitive relation.
- **Symmetric relation [Optional]:** A symmetric relation implies that if X has a symmetric relation with Y, then Y also has the relationship with X.

- **Functional relation [Optional]:** A functional relation implies that if X has a functional relation with Y and X has a functional relation with Z, then Y and Z must be the same.
- **TransitiveSymmetric relation [Optional]:** A transitive and symmetric relation provides the combined behaviour of the transitivity and symmetry.
- **InverseFunctional relation [Optional]:** An inverse functional relation provides the inverse of the behaviour provided by the functional relation.

In the ontology-based metamodel, these dependency relations are represented as OWL object properties with the appropriate behavioural attributes set (e.g. transitive) using the relevant OWL object property attribute. The behavioural attributes are set when the metamodel is created using the Protégé ontology development environment. They are subclasses of a general dependency relation object property called "DependencyRelation" as shown in Figure 3-2 below.

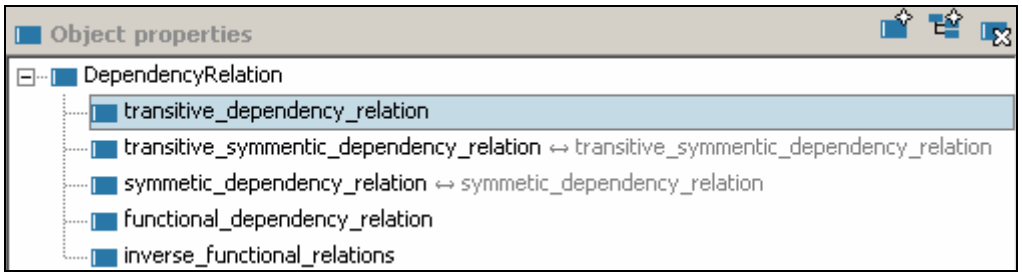

Figure 3-2: Dependency Relations in the metamodel

The metamodel also defined the following descriptive properties for the dependency relationships. These attributes play an important role in supporting the users understanding of the origin and importance of any computed dependency.

In the ontology-based metamodel, the descriptive attributes are defined as an OWL class to represent the each of the dependency attributes. The attributes can be associated with any dependency relation using OWL object properties.

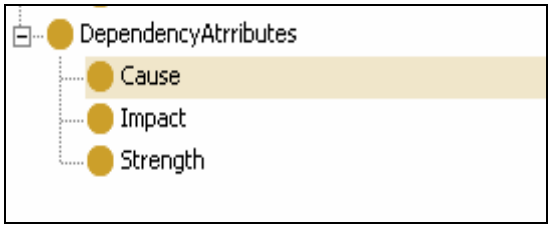

Figure 3-3: Descriptive Dependency Attributes supported in the metamodel.

The descriptive dependency attributes are optional because they provide supplementary information for the dependency model and as such do not need to be in the computation of the dependencies chains. However, the usage of "Strength" attribute enabled more detailed dependency analysis as shown in Section 3.2.6.4.

The descriptive dependency attributes, shown in Figure 3-3, are described below:

Strength (Level) [Optional]: This attribute is a measure of the frequency of the need or the importance of this dependency from any architectural entities viewpoint. In the context of ontology-based integration systems, this can be interpreted as the level at which the dependency occurs. For example, if element A depends on element B and element B depends on element C then the second dependency relationship is at the second level from the viewpoint of element A.

In the ontology-based metamodel, this is represented as the "Strength" concept that can be associated with a dependency relation using the "hasstrenghtattribute" object relation. This is a property (integer) to represent the level at which the dependency occurs. In the metamodel, the value range for this property is unrestricted (i.e. it can take any integer value).

Impact [Optional]: This attribute is used to define a measure of how the entity's function is affected by compromise or failure at this particular dependency. This can be interpreted as the extent to which the elements, that are part of the dependency, are critical to the operation of the integration system. For example, if the elements that comprise the dependency relation are used in all (or many) integration use cases then the failure to evolve the mapping would have a high impact on the integration system.

In the ontology-based metamodel, this is represented as the "Impact" concept that can be associated with a dependency relation using the "hasimpactattribute" object relation.

This is represented as a datatype property of type String. In the metamodel, the value range for this property is unrestricted (i.e. it can take any string value).

Cause [Optional]: This attribute provides a definition of the underlying cause of the dependency relationship. As will be seen later in the case studies, in ontology-based integration systems, we see dependencies occurring between mappings due to overlapping of data elements (called overlapping dependency) or overlapping of the functions specified for a mapping (called function-based dependency).

In the ontology-based metamodel, this is represented as the "Cause" concept that can be associated with a dependency relation using the "hascauseattribute" object relation. This is represented as a datatype property of type String to represent the underlying reason for the dependency. In the metamodel, the value range for this property is unrestricted (i.e. it can take any string value).

The OWL code for these relationships is shown the figure below (Code 2).

```
<owl:ObjectProperty rdf:ID="hascauseattribute">
        <rdfs:domain rdf:resource="#ArchitecturalEntities"/>
        <rdfs:range rdf:resource="#Cause"/>
 </owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="hasimpactattribute">
        <rdfs:domain rdf:resource="#ArchitecturalEntities"/>
        <rdfs:range rdf:resource="#Impact"/>
 </owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="hasstrenghtattribute">
        <rdfs:domain rdf:resource="#ArchitecturalEntities"/>
        <rdfs:range rdf:resource="#Strength"/>
 </owl:ObjectProperty>
```

Code 2: Metamodel Descriptive Attributes

Dependency Graph: This concept represents the domain that is under study and, as seen later, is represented by the root node in the graph visualisations. The graph concept supports the following attributes:

- **Type [Optional]:** A property of type String to represent the graph type (e.g. cyclic, acyclic, direct, undirected). This can be used to support the automatic rendering of the graph in the visualisation factory code seen later.

In the metamodel, the value range for this property is unrestricted (i.e. it can take any string value).

- **Name [Optional]:** A property of type String to provide a name for the graph. In the metamodel, the value range for this property is unrestricted (i.e. it can take any string value).
- **Version [Optional]:** A property of type String to represent version number of the graph. In the metamodel, the value range for this property is unrestricted (i.e. it can take any string value).

The OWL code for the metamodel is provided in full in the Appendix. I.

3.2.4 Domain Specific Dependency Model Creation Process

The dependency metamodel can be used to create domain specific dependency models using the process defined in (Figure 3-4). This process was developed during experiment two by analysing the steps needed to build the domain specific model for the generalised ontology-based integration test system. The steps were then generalised to the process shown below:

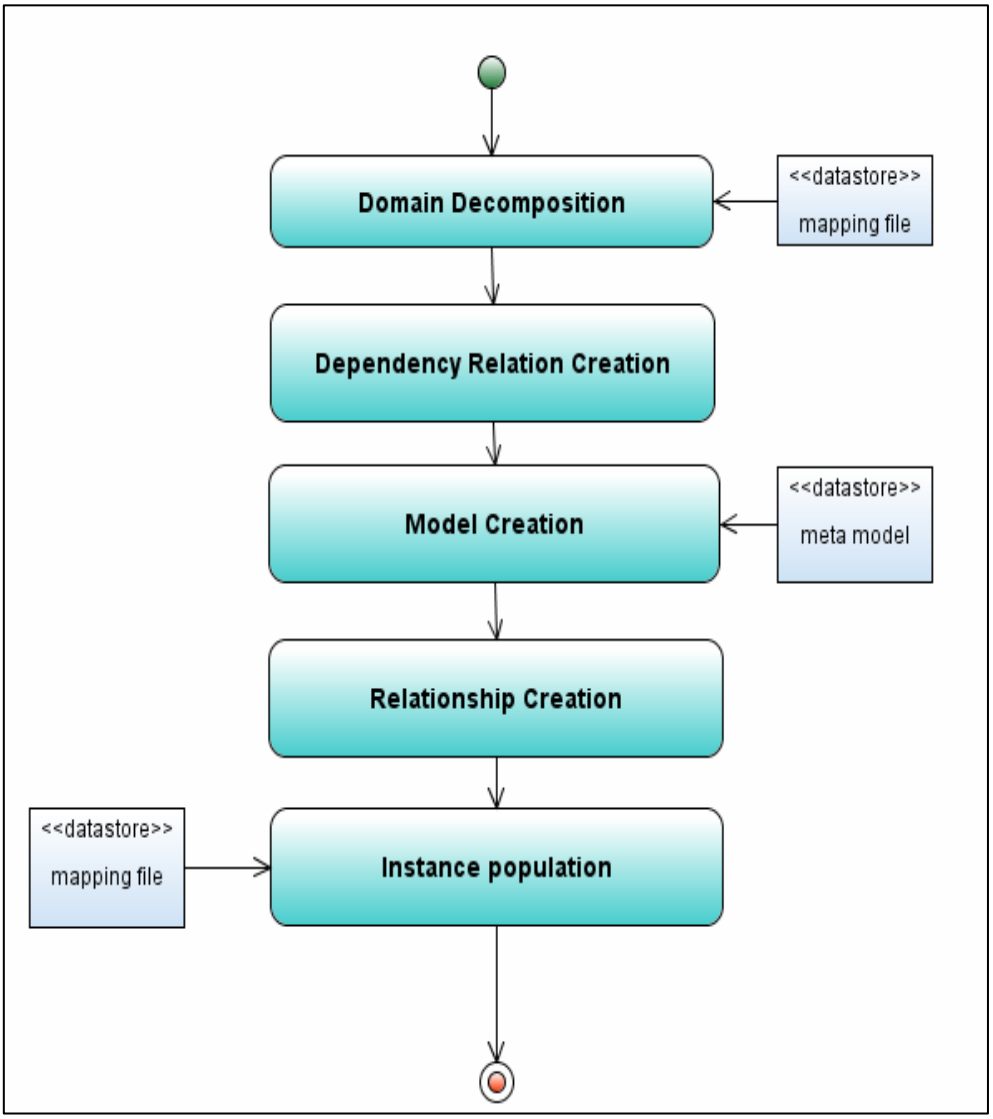

Figure 3-4: Process for domain model creation

Step 1 - Domain Decomposition: This step is responsible for the decomposition of the domain into abstract entities that represent the architectural parts of the domain. This process step is required to create the abstract elements that participate in dependency relationships. The purpose of decomposition is to identify the key elements in the

system under study. Once these key elements have been identified, appropriate abstractions of these elements can be created and represented as Architectural entities from the metamodel.

Step 2 - Dependency Relation Creation: This step is responsible for the design of the dependency relations that exist between the architectural elements defined in the previous step. This step is required to understand the relationships between the elements in the domain under study. Input from domain experts is needed to support the development of the relations. The experts need to select which types of relationship exist between the elements in the domain from the available relationship types supported by the metamodel. This step creates the input needed to describe the behavioural and descriptive attributes of the dependency metamodel.

Step 3 - Model Creation: This step creates subclasses of the metamodel architectural entities for all entities designed in step 1. These subclasses are domain specific and form the parts of the system that participate in the dependency relationships.

Step 4 - Relation Creation: This step creates child object properties of the metamodel dependency relations for all relations designed in step 2. These properties are domain specific and are formed between the domain specific architectural elements from the model creation step. Once this step is complete, the domain specific dependency model has been created. The final step is to populate the model with data instances.

Step 5 - Instance Population: This step creates and populates instances of the architectural elements into the domain specific model. Data from the system under test is required to carry out this step. The data sources must identify instances of the architectural entities and the first dependency of that architectural entity. In the context of the dependency model that was created for generalised ontology-based integration system in this thesis, the mapping file provided the source of this data.

3.2.5 A Domain Specific Ontology-Based Dependency Model (OBDM)

This section describes the ontology-based dependency model that was created using the process from Figure 3-4 (Section 3.2.4) and the ontology-based metamodel from Section 3.2.3.

This dependency model was created to support the evolution of mappings in a generalised ontology-based integration system that is described in detail in Section 3.3.

The generalised ontology-based integration test system was deployed to resolve the heterogeneity in the logistics data in experiment two. The mapping file from the integration system was used to support the development of the ontology-based dependency model.

This section discusses steps one to four from the process (Figure 3-4) to create the ontology-based dependency model from the mapping file for such an integration system. The remaining step for the process is described in Section 3.2.6 where are tool (called Towards Ontology Mapping Evolution) was developed to support step five.

Domain Decomposition

The decomposition step enabled the creation of sets of architectural abstractions (called architectural entities) that represented key features of the integration system.

The mapping file was analysed to identify the major integration system components referenced in the mapping file. The analysis was carried out by drawing out each mapping using a simple graphical form where nodes represented ontology classes, data properties and mappings and arrows represented a dependency relationship. Using this analysis the ontology integration system was decomposed into its core architectural elements. This yielded a list of the key elements that form the architectural elements of the ontology-based dependency model. The architectural elements were represented in OWL classes and are subclasses of the metamodel architectural elements and thus inherit the metamodel behaviour.

Dependency Relation Creation

This step required the creation of pair wise dependency relations between each of these sets of architectural elements already identified. The relations between the architectural elements form the dependency relations between any two elements in the system. The relations were created based on expert understanding of the architecture of the integration system. Each relation created in this process step is added to the model as a sub relation of the metamodel dependency relationships.

In the case of generalised ontology-based integration system, the dependency relations were derived from the analysis of the mapping file and how the mapping file is used (executed) by the integration system.

The sequence of execution of the generalised ontology-based integration system indicates that the concepts in integration ontologies depend on the mapping specification and the mapping specification depends on the lower ontology concepts that in turn depend on the data sources.

Model Creation

The previous steps had identified the architectural elements and dependency relations between them. The model creation step used the Protégé ontology development tool [Protégé] to create a dependency model in OWL [OWL]. This step required that the dependency metamodel is imported into Protégé. The architectural elements were created as sub concepts of the metamodel "Architectural Entity" concept. The dependency relations were created as sub properties of the metamodel "Dependency relations".

The resulting domain specific ontology-based dependency model is shown in Figure 3-5.

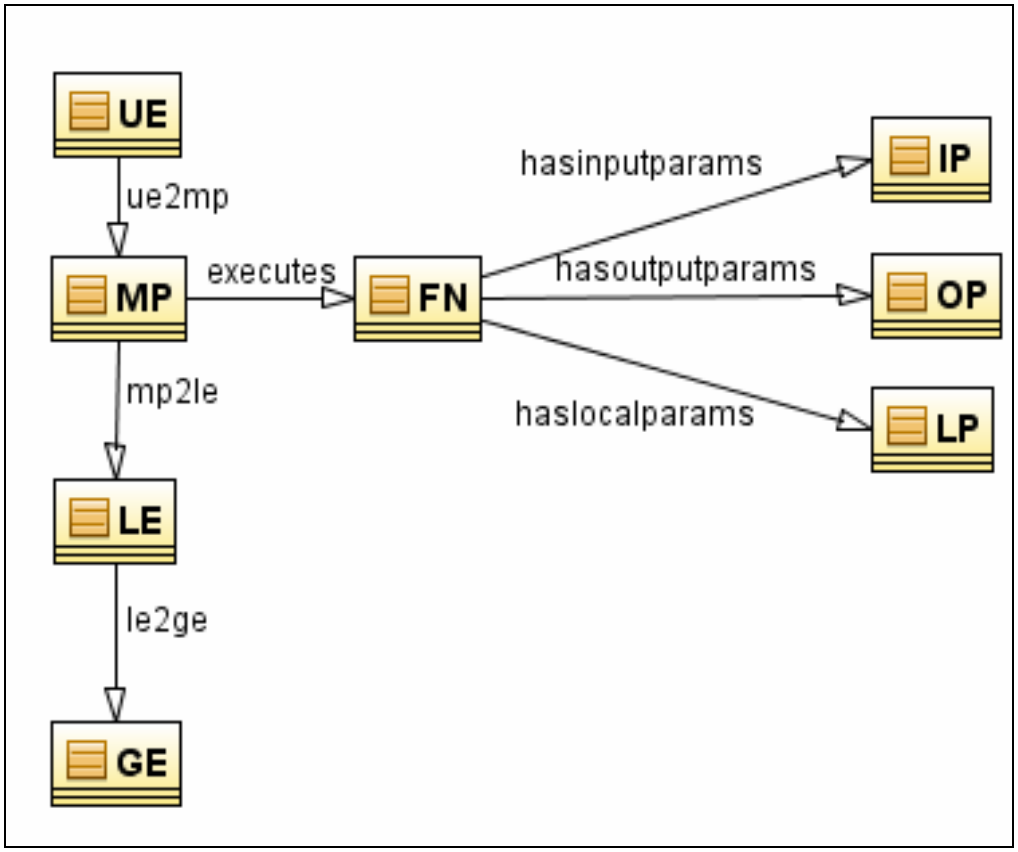

Figure 3-5: Domain Specific Dependency Model

The model consists of the following parts:

Architectural Elements

The fundamental parts of the integration system are represented by the following abstracted elements.

- **Upper Entity (UE):** Represents an ontology class or property from the integration ontology of the generalised ontology-based integration system. Each upper entity has a dependent object relationship with a mapping using the "ue2mp" dependency relationship.
- **Mapping (MP):** Represents an ontology mapping from the generalised ontology-based integration system. Each mapping has a dependant object relationship with a lower entity using the "mp2le" dependency relationship. Each mapping has a function (FN) associated with it using the "executes" dependency relationship.

- **Lower Entity (LE):** Represents an ontology property from the lower ontology. This represents a URI[13] (that has been created automatically using the D2RQ API [D2RQ API]. Each lower entity has a dependant object relationship with a grounded entity (GE, discussed next).
- **Grounded Entity (GE):** Represents a database property from the data sources used by the generalised ontology-based integration system.
- **Function (FN):** This concept represents the executable function that is used to transform the data sources elements into the ontology class. These functions are referenced by the mappings in generalised ontology-based integration system.
- **Input Parameters (IP):** The input parameters of the mapping function (i.e. names of the UE, MP, LE or GE elements that are used in the input parameters of a mapping function).
- **Output Parameters (OP):** The names of the UE, MP, LE or GE that are used in the output parameters of a mapping function.
- **Local Parameters (LP):** The names of the UE, MP, LE or GE that are used in the local code of a mapping function.

The following object relations form the dependency relationships between the architectural elements in the model.

- **UE2MP**: Transitive and symmetric object property with domain UE and range MP.
- **MP2LE**: Transitive and symmetric object property with domain MP and range LE.
- **LE2GE**: Transitive and symmetric object property with domain LE and range GE.
- **EXECUTES**: An object property with domain MP and range FN.
- **HASINPUTPARAMS**: An object property with domain FN and range IP.
- **HASOUTPUTPARAMS**: An object property with domain FN and range OP.

[13] A Universal Resource Identifier.

- **HASLOCALPARAMS**: An object property with domain FN and range LP.

The dependent relations (ue2mp, mp2le, le2ge and executes) are sub properties of the appropriate metamodel relations (e.g. transitive_symmentic_dependency_relation") that is in turn a sub property of the general "DependencyRelation" object property from the metamodel. This allows transitive propagation to occur at the more general "DependencyRelation" relation level, enabling chains of dependencies to be built using an OWL reasoner.

Figure 3-6 provides an example of the dependency relations (ue2mp, mp2le and le2ge) described above. This figure was generated using the TomE tool described in Section 3.2.6 and the logistics data described in Experiments two and four (Section 4.4 and 4.8). A dependency chain for a UE (Upper entity) called "UE-carriers-names" is shown in the figure below. This dependency chain has a MP (Mapping Point) called "MP-c1" LEs called "LE-exp_test_db2-logistics-Awards" and "LE-exp_test_db2-Logistics-Awards" amd GEs (Ground Entities) called "GE-exp_test_db2-logistics-Awards" and "GE-exp_test_db2-Logistics-Awards".

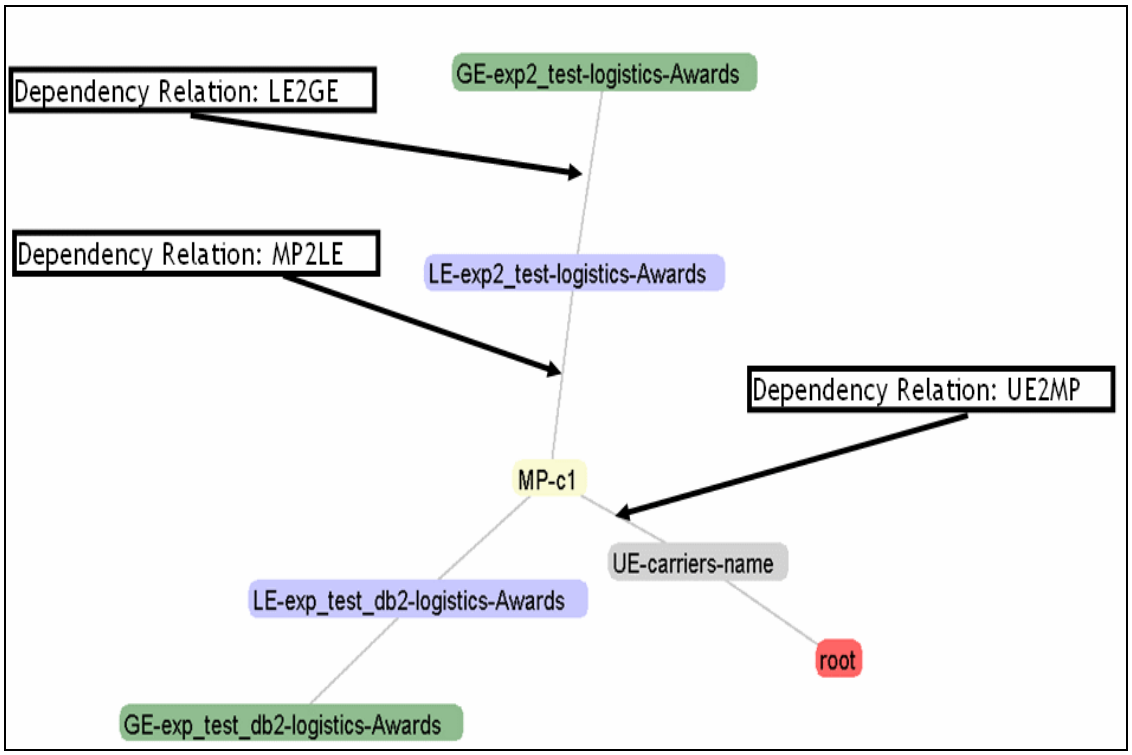

Figure 3-6: Illustration of Dependency Relations

The UE represents a concept in the domain ontology of the integration system, the MP represents the mapping between the domain ontology and datasources and the LEs and GEs represented the datasources entities which are accessed by the integration system. [Note that the TomE tool does not provide a view of the functions specified for this mapping – this limitation is discussed in Future work in Section 5.3.2]

3.2.6 Dependency Analysis Tool (TomE) Implementation

The ontology-based dependency metamodel and process for domain model creation (Figure 3-4) was used to create an ontology-based dependency model in OWL as described earlier.

A tool called TomE (Towards Ontology Mapping Evolution) was developed to provide software support for the last step in the process (that is, instance population described in section 3.2.4), and in addition to enable analysis and visualisation of the dependencies in the generalised ontology-based integration system.

The TomE tool was used in experiments three and four to support the analysis of the dependencies in the mappings in an ontology-based integration system from experiment one.

3.2.6.1 TomE Functional Architecture & Design

The TomE tool takes a mapping file from the generalised ontology-based integration system as input and produces visualisations of the dependencies between the architectural elements based on the dependency relations described in the ontology-based dependency mode.

The functional architecture of the TomE tool is shown in Figure 3-7 below.

The architecture is composed of four functional areas. Each functional area follows a factory design pattern where each functional area consumes data from the previous area, processes it and passes it to the next area. This approach provided functional segregation of code.

The “Mapping Factory” is responsible for loading the mapping file and generating dependency model instances. The “Model Factory” is responsible for loading and reasoning over the ontology-based dependency model. The “Dependency Factory” is responsible for generating dependency graphs using the ontology-based dependency model. The “Visualisation factory” is responsible for generating visualisations of the dependency graphs.

The design of each functional area is described below.

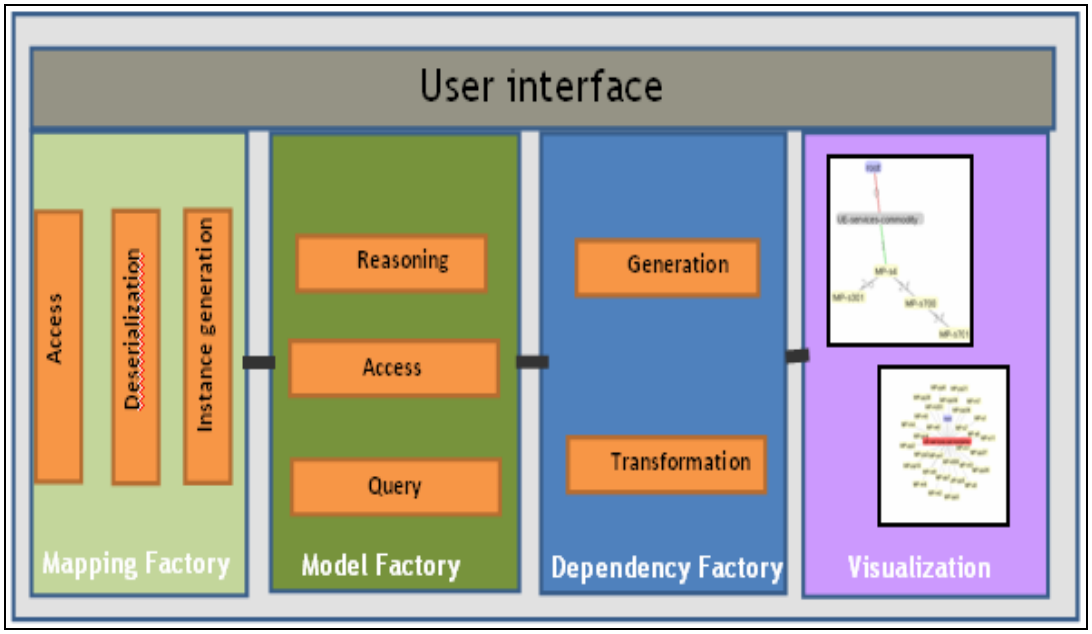

Figure 3-7: Functional Architecture TomE Tool

***Mapping factory**:* The mapping factory is a set of classes that provide access and deserialisation functionality of mappings. Mappings are decomposed by this subsystem into Java classes that represent each of the architectural elements in the domain model. In the current implementation of the tool, a bespoke adapter class is needed to convert the elements of the mappings into architectural elements as described in section 1 of the process above. The final step is to generate ontological concepts for the architectural entities. This is the role of the instance generation functionality.

The class diagram for the access and deserialisation is shown in Figure 3-8.

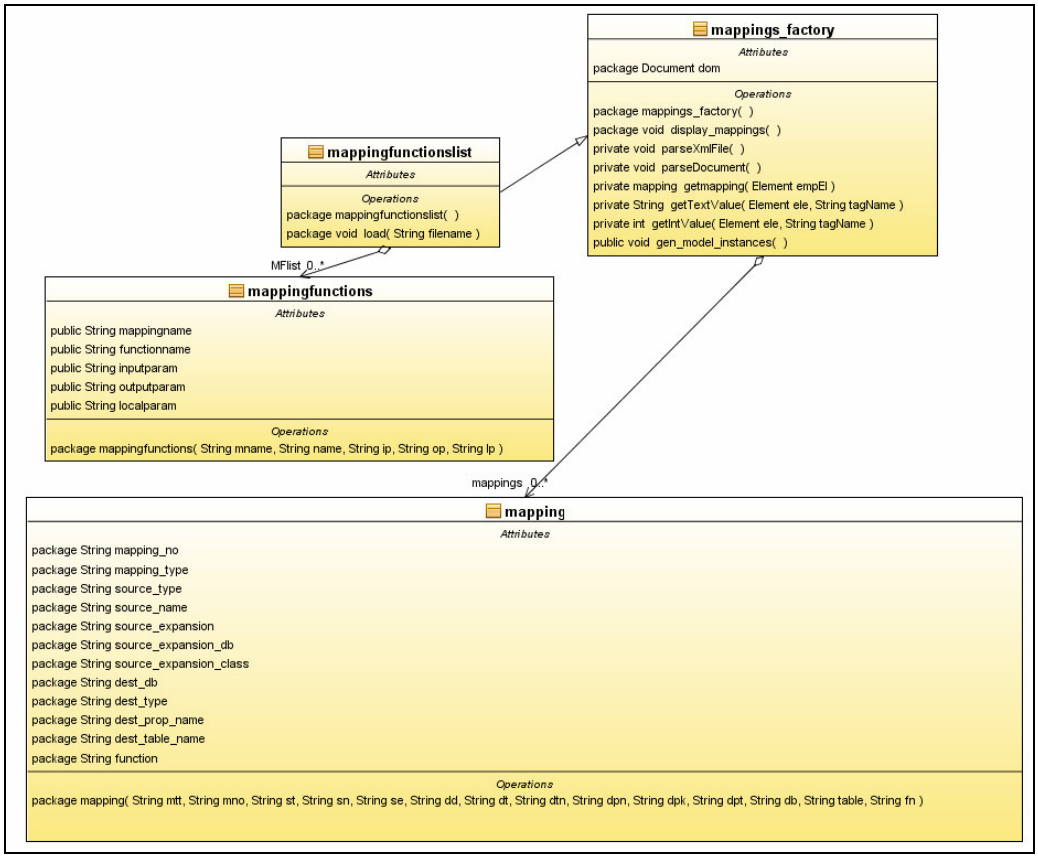

Figure 3-8: Class diagram for mapping factory

With reference to Figure 3-8, mappings are loaded from an XML file using Java SAX API [SAX] and stored in a temporary DOM document object. This is handled by the mapping_factory methods. The mapping document is parsed element by element to extract the mappings fields into a Java list of mapping objects. In the current implementation of the adaptor, a "mappingfunctionslist" class processes the mapping functions separately as the function names need to be manually extracted from the function descriptions in the generalised ontology-based integration system. The "mappingfunctionslist" class is responsible for loading the function names associated with each mapping.

The gen_model_instance() method of the mappings_factory class creates OWL model instances of architectural entities from the runtime list of mappings. In the current implementation, the adaptor is hand coded to decompose the mappings list to the appropriate OWL instances and relationships that have been designed in the domain specific model. The OWL instances are also saved to file to allow offline analysis and debug (in Protégé for example).

An example OWL instance is shown below:

```
<UE rdf:ID="UE-UE1 ">
<ue2mp rdf:resource="#MP-MP1"/>
</UE>
```

Code 3: Instance (UE-UE1) of an Architectural Entity called a UE.

The resource type UE-UE1 represents the architectural entities of type "UE". In this case, it has one property called "ue2mp". This property is an object relation that is defined in the domain specific model.

***Model factory**:* The ontology-based dependency model (OBDM) was created using Protégé [Protégé] and realised in OWL-DL as described earlier. The model factory is responsible for creation of the in memory ontology model from the dependent model, preparation and validation of the model. The model is created in memory using the Jena API [Jena] as follows:

```
DM_model =
ModelFactory.createOntologyModel( PelletReasonerFactory.THE_S
PEC );

DM_model.read( ont );


DM_model.validate();
```

Code 4: OWL model creation in Jena.

These steps create the dependency model, bind it to the Pellet reasoner [Pellet] and perform a validation of the model. Validation is used to verify the correctness of the ontology elements created (architectural entities) in previous steps. Dependency reasoning is to compute the dependent elements on any specified architectural element. A containment reasoning method (builddependencies) invokes reasoning over the model to compute the containment of any specified resource. The TomE tool computes the containment for every upper entity (UE) in its model. Each containment computation yields a list of the elements that depend on that UE. In the current implementation of TomE, these elements are stored in a global list structure using a Java array list type. The list structure is a simple representation of the dependencies

related to the ontology-based integration system domain that contains fields for upper entity (UE), mapping (MP), lower entity (LE) and ground entity (GE).

In the context of ontology-based integration system mappings, the in-memory dependency structure has the following form based on Java array lists (Figure 3-9).

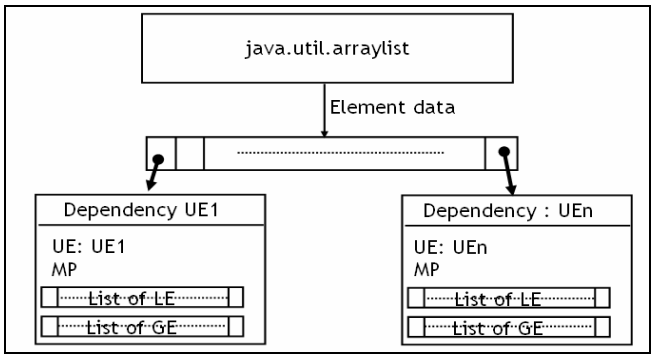

Figure 3-9: In memory Dependency

From these list structures, either the full dependency graph (all UEs) or individual dependency graphs (single UE) can be created. While this may not be the most optimum storage method, the array list is well supported for search in Java using Java Iterators and Collections class. This strategy made programming of the list data simple for the prototype.

The class diagram for the model factory is shown in Figure 3-10.

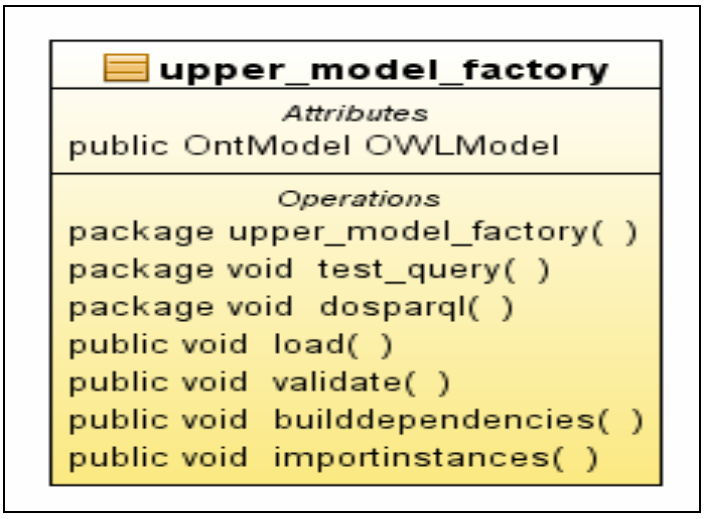

Figure 3-10: Class diagram for model factory

Dependency factory: The dependency factory is responsible for constructing the dependency graphs from the in-memory dependency lists. In the current

implementation of the tool, the dependency graphs for all architectural elements are pre-computed because the computation time is fast (minutes) even for the large datasets used with hundreds of architectural elements.

The dependency factory creates GraphML [GraphML] output format from the computed dependency graphs from the model factory. The transformation from dependency model to GraphML is straightforward because the dependency model Architectural Element is transformed to GraphML "node" and dependency model dependency relationship is transformed to GraphML "edge".

To simplify implementation and aid debugging, the TomE tool creates individual GraphML files for the views described in the visualisation section. GraphML also supported labels on edges of a graph. This is used in the prototype to assign levels and types to the dependency relationships.

Visualisation: The visualisation subsystem is responsible for displaying the computed graphs. The subsystem provides functionality for search, node expansion and zooming features on each of the three types of graphical views.

The three graphical views are:

- **Full dependency graph**: provides the graphical view of the computed dependency for each upper element in the dependency model.
- **Individual node dependency graph**: provides the graphical view of any user selected upper entity. This allows the user to drill down to a localised part of the dependency graph. (Figure 3-11).

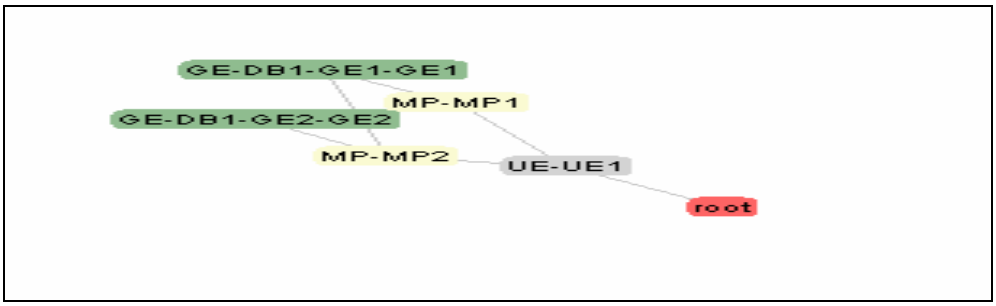

Figure 3-11: Sample Dependency Graph for a UE called "UE1"

Note that the dependency graph for individual nodes (Figure 3-11) does not display the function associated with the mapping point. However the TomE

system does include functions in the computation of dependencies. This is discussed in the Future Work 5.3.2.

- **Individual node dependency graph with levels and types**: provides the view of any user-selected node with the level and dependency types displayed (Figure 3-12). In the current implementation of the TomE tool, this view is fully automatic, as the user must update the tool with a list of UEs to compute this view because it is unlikely to be required for every upper entity.

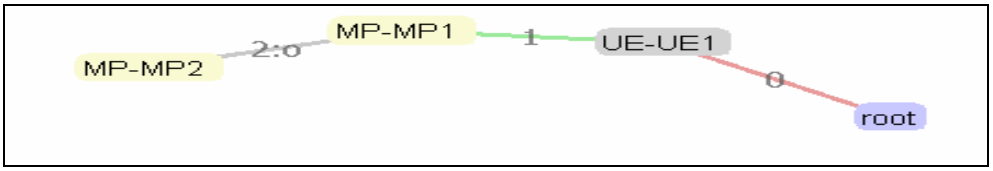

Figure 3-12: Sample Dependency Graph with levels and types.

3.2.6.2 TomE Call Sequence

The process to create and view dependencies is managed by the user who follows a number of screens in the TomE tool. The sequence of method invocation between the user interface screens and factory classes is shown in Figure 3-13.

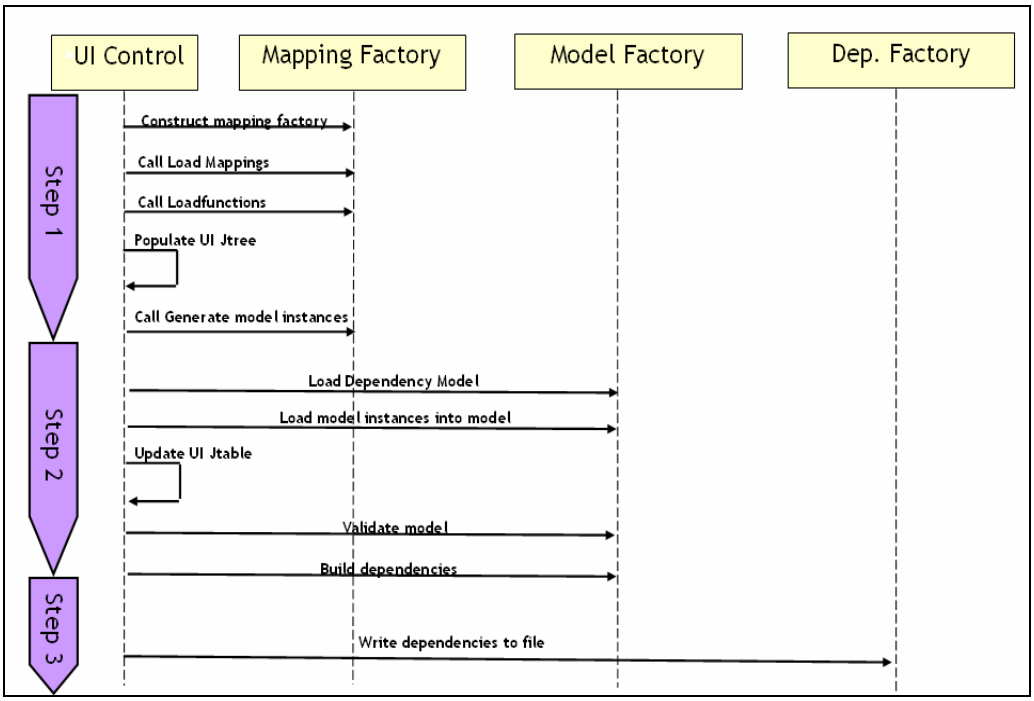

Figure 3-13: Call Sequence Diagram for TomE

The mappings which the integration system uses have already been created before the dependency management using the TomE tool can begin. For the work in this thesis, mappings were created manually using the help of domain experts as described in experimental setup (Section 4.2.5).

Dependency management using TomE has three steps. The first step involves calls to the mapping factory to load and process the mapping file. The mapping factory consumes mapping files and produces model instances.

The second step involves calls to the model factory. The model factory consumes the dependency model, dependency model instances (from step 1), run ontological reasoning over the OBDM and produces in memory representations (Java lists) of the dependencies.

The final step involves calls to the dependency factory. The dependency factory consumes the in memory representations of dependencies and produces GraphML representations that are ready for visualisation.

3.2.6.3 TomE Ontological Reasoning Operations

Reasoning over the OBDM is carried out in the model factory (3.2.6.1) as described below. The TomE tool performs ontological reasoning over the dependency model to automatically carry out some critical functions related to the creation of dependencies as described below. The reasoning is carried out by the Pellet reasoner bound to the Jena model that instantiates the ontology-based dependency model.

The first type of automated reasoning that TomE tool used is the model validation. This ensures that the instances of the dependency model that have been constructed in the mapping decomposition process are correct.

The TomE tool also uses reasoning over the model to pre-compute the dependencies for each UE defined in the model. This is the longest computation task that the tool needs to carry out. The OWL axiom for each UE takes the form shown below:

```
<owl:Class rdf:ID="InferDepsOf_UE1">
<owl:equivalentClass>
<owl:Restriction>
<owl:onProperty rdf:resource="#DependencyRelation"/>
<owl:hasValue rdf:resource="#UE1"/>
</owl:Restriction>
</owl:equivalentClass>
</owl:Class>
```

Code 5: OWL Axiom to infer dependency chain for any UE

This pre-computation approach might be a cause for concern if there were large (thousands) of UEs in the model. In the samples with approximately 100 UEs from the logistics domain used in experiment two, the computation still finishes in less than 4 minutes on a lower end machine.

A simple solution to this potential problem would be to compute the dependencies on demand based on the particular UE the user is interested in working on.

3.2.6.4 TomE Dependency Types and Levels Processing

The TomE tool also computes the levels and types of dependency relations so that these can be rendered in the graphical visualisation. In the current implementation of the tool, this view of the data is not pre-computed. A second invocation of the TomE because the tool is needed to compute and record the level and type for each relationship as it built the chain of dependencies. The second invocation is needed to allow the user to specify which entities to compute the types and levels for.

The level and types were computed using the procedure shown in Figure 3-14 below.

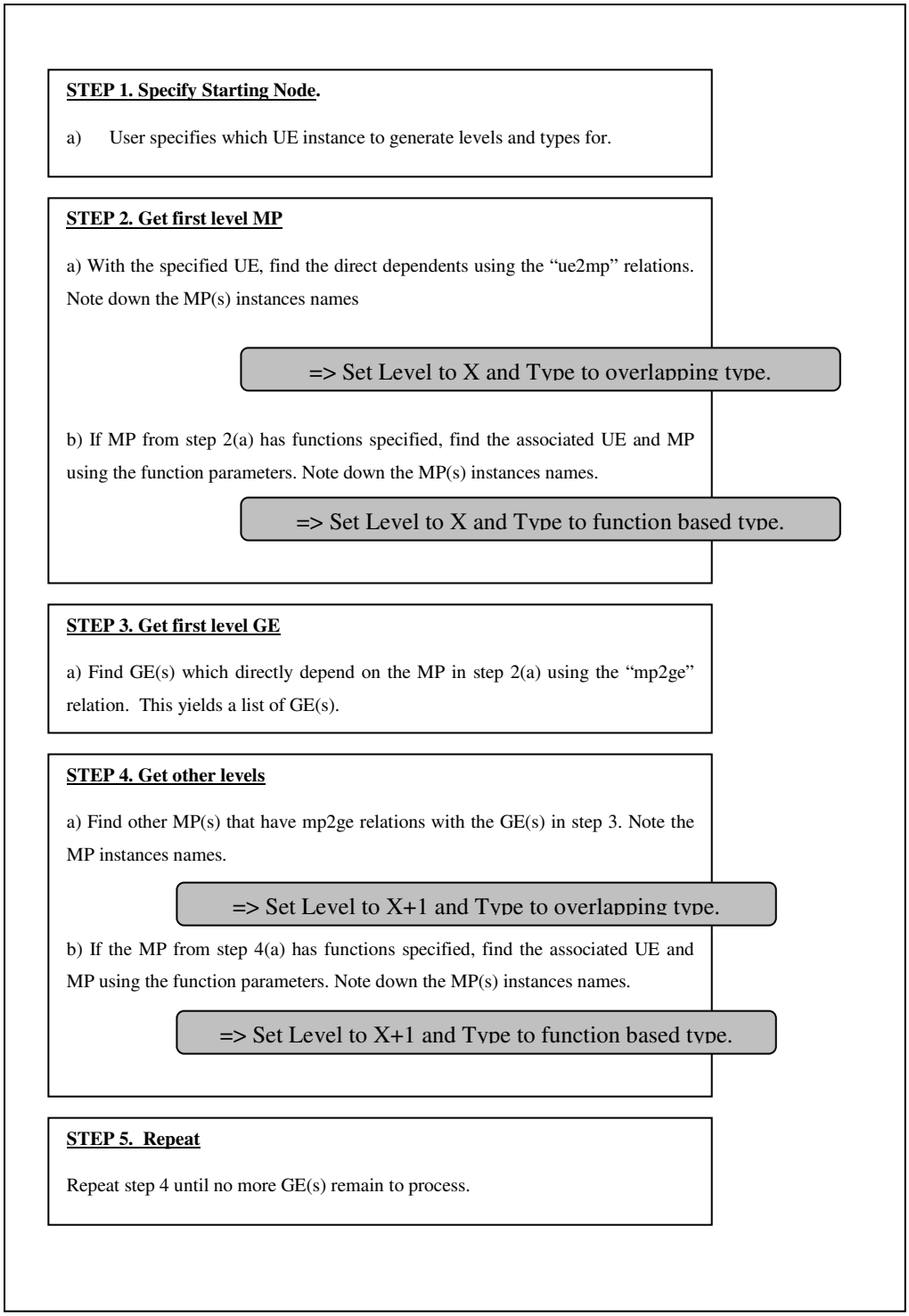

Figure 3-14: TomE Levels & Types Algorithm

The procedure is implemented in Java and does not make use of Pellet reasoning. This would have required the creation of OWL axioms to support the various search and find operations of the procedure. This java based approach was taken for the first implementation of the Types and Levels algorithm but should be implemented in future versions of TomE by adding the axioms to the domain specific model and

updating the TomE tool to enable the axiom to be processed. This would reduce the code complexity of the TomE tool by allowing reasoner to carry out parts of the algorithm.

The current implementation created a GraphML output file for the user specified UEs. The GraphML file can then be loaded in the TomE tool. The output file used dedicated tags to label the levels and types as shown in the GraphML snippet below:

```
<key id="type" for="edge" attr.name="type" attr.type="string"/>
<key id="level" for="edge" attr.name="level" attr.type="string"/>

<edge source="MP1" target="MP2">
<data key="level">2</data>
<data key="type">2</data>
</edge>
```

Code 6: GraphML code snippet

3.2.6.5 Technical Implementation

The TomE tool was implemented as a Java Desktop application using the Eclipse development environment [Eclipse]. The user interface provides a set of tabbed panes (Figure 3-16 and Figure 3-17) that map to the functional areas described above. The process to generate the graphical views can be controlled by user interface to allow inspection of the output at the end of each point in the process.

The third party libraries used by TomE are shown in Figure 3-15:

Component	Task	Library
Mapping factory	XML processing	SAX
Model factory	Ontology Management	Jena 2.0
Model factory	Ontology Reasoning	Pellet2.0.0rc4
Model factory	Ontology Query	ARQ
Visualisation	GMF, GraphML	Eclipse GMF, Java GraphML

Figure 3-15: API Usage

To carry out dependency analysis using the tool, the user needs to navigate through the TomE tool as described in the following three steps.

Step 1 – Load and Decompose Mappings

The first tabbed pane (Figure 3-16) of the TomE tool provides control over the mapping factory classes. This pane loads the mapping file and generates the instances of the dependency model from the mapping file.

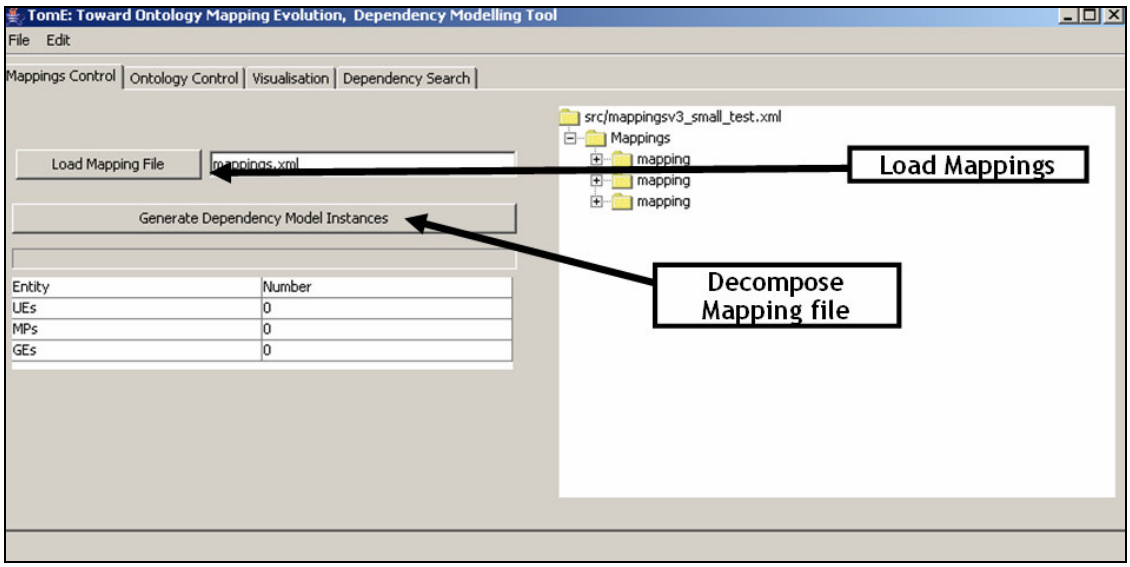

Figure 3-16: TomE Control Panel

Step 2 – Dependency Model Control and Dependency Generation

The second pane (Figure 3-17) of the tool provides user control over the ontology model factory. Using this pane, the user can load the model, run dependency inferences and create the GraphML outputs.

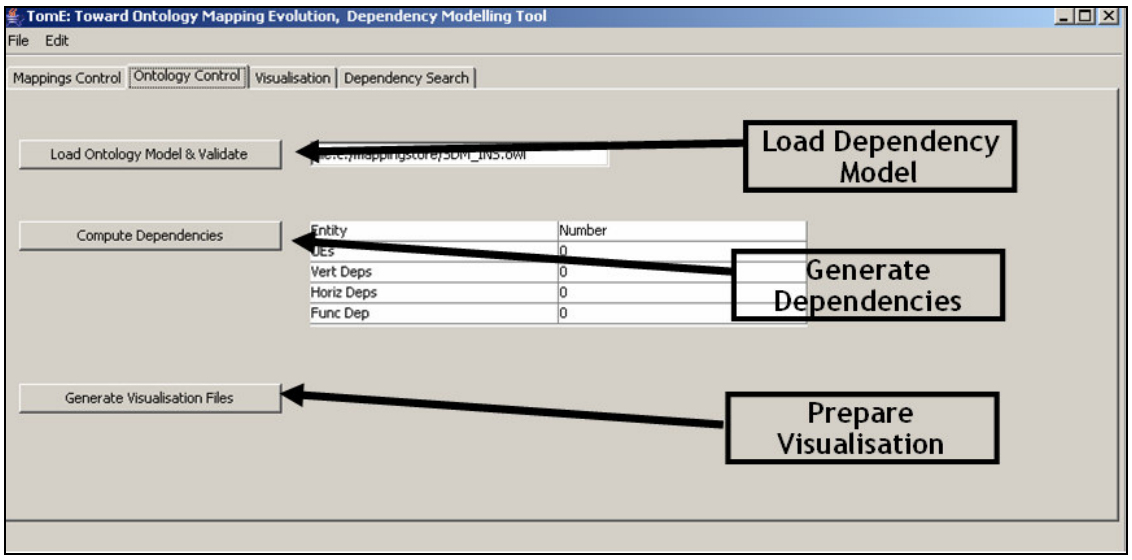

Figure 3-17: TomE Ontology Control

Step 3 – Run Visualisation

The final step is to load the visualisation of dependencies. The third pane (**Figure 3-18**) runs the visualisation. The visualisation provides four main areas of functionality. Dependency visualisations are loaded using the File menu.

Once loaded, the dependencies can be expanded, collapsed, zoomed and repositioned using the main dependency-viewing pane (Area 1 below). The behaviour of main viewing pane can be controlled using the graph control pane (Area 2 below) using the standard functions provided by the GraphML viewer.

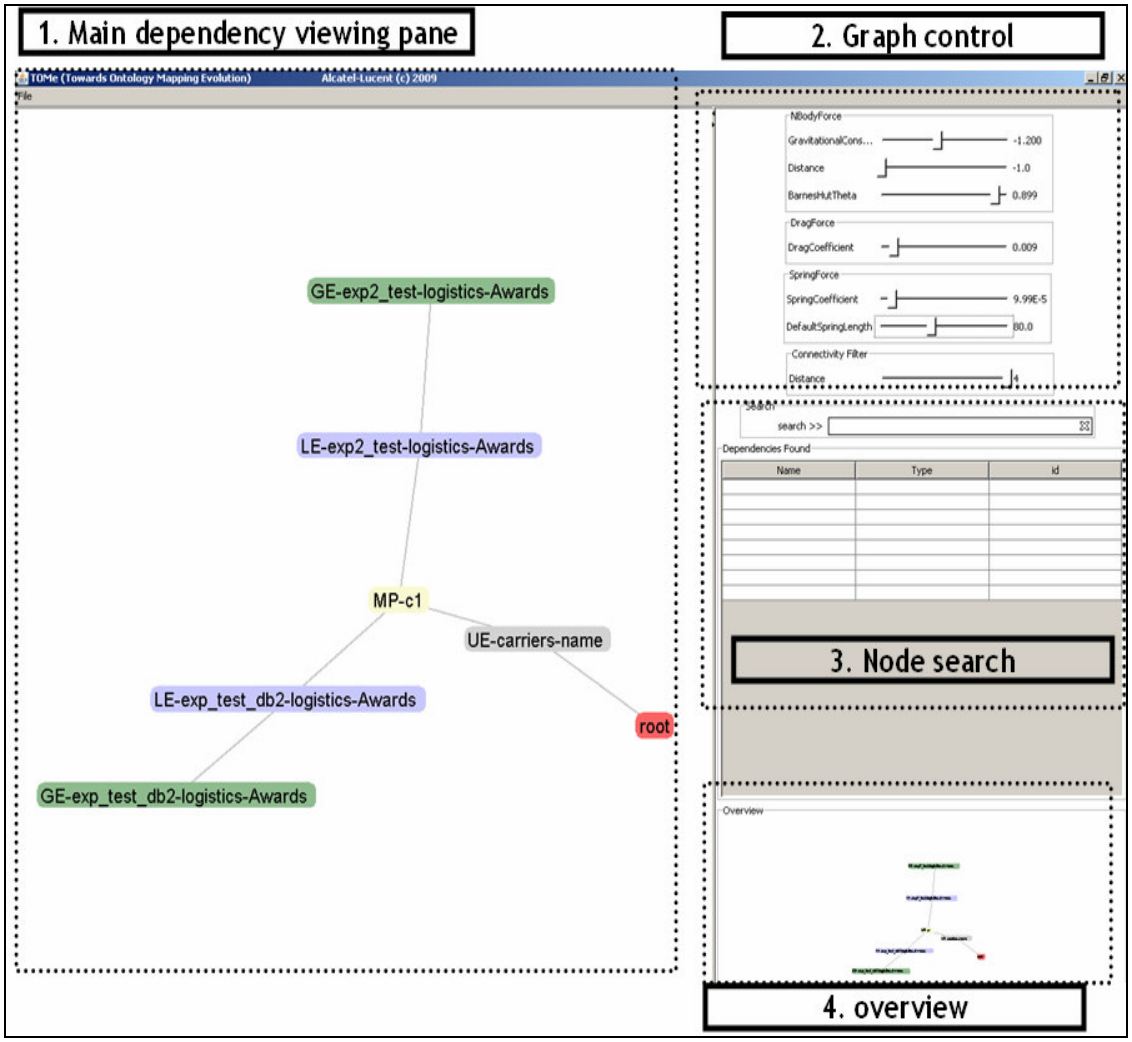

Figure 3-18: TomE Visualisation

Using the Node Search (Area 3 in Figure 3-18), any individual node can be searched for. The overview panel (Area 4 in Figure 3-18) provides an overview panel of the entire graph that is useful when looking at very large graphs.

3.2.6.6 Worked Example of TomE Usage

A worked example of the usage of the TomE tool is provided in Appendix III. It illustrates the input and outputs of the TomE tool. It assumes the existence of a mapping file based on a simplification of the mappings using in the generalised ontology-based integration system designed for experiment one.

3.3 Generalised Ontology-Based Integration Test System (HotFusion)

This section describes the design of the generalised ontology-based integration test system. The test system was used in experiments one and two to support the integration of the heterogeneous data sources based on the use cases described in the experiments.

Note that a fully functional ontology-based integration system was not needed, rather the test system focused on the specific requirements arising from the aims of experiment one. The requirements for the ontology-based integration test system are discussed in Section 3.3.1.

The test system was implemented in a tool created by the author of this thesis that was called HotFusion.

3.3.1 Design Requirements

The requirements for the ontology-based integration system arose from the aims of experiment one. The aim of experiment one was stated as "Discovery of key issues related to integration performance when applying an ontology-based integration approach in an industrial context."

Following a state of the art review of approaches to ontology-based integration, the hybrid ontology approach [Wache et al. 2001, Cruz and Xiao 2005] was adopted to create the generalised ontology-based test system. The hybrid approach offers improvements in implementation effort, support for semantic heterogeneities, adding, and removing of source over the single or multiple ontology approaches [Wache et al. 2001].

The key requirements for the design of a test system based on the hybrid approach were:

- Integration system to provide a general integration engine (code) that would operate the same way across different integration domain. (Requirement 1)
- Clear separation of the domain ontology, mappings and data sources. (Requirement 2)
- User interface to enable step-by-step analysis of the integration. (Requirement 3)

Requirement 1 was created to ensure that as different integration use cases were tested with the system, the basic integration engine (or code) did not have to change.

The system that was designed was tested using the THALIA integration benchmark [Stonebraker 2005] to ensure it provided adequate integration capability.

Requirement 2 was created to ensure that relationships between the major components of the system had well defined interfaces and that the minimum level of dependency existed between the major parts of the system. A factory design approach was used that divided the system into functional areas that consume an input, process it and pass it on to the next functional area.

Requirement 3 was created to ensure that as integration use case was running, the outputs of each step could be verified. To realise this, the generalised ontology-based integration system has a graphical user interface that provides control over the execution of the integration use case.

These specific requirements, defined above, for the generalised ontology-based integration system meant that a full functional ontology-based integration was not developed and was considered beyond the scope of this research.

In particular, the test system here does not provide functionality in a number of areas where a fully functional system would need. For example, a full functional integration system would provide user interface support of the creation of ontologies, mapping creation and integrated view creation and reporting.

3.3.2 System Overview

The generalised ontology-based integration test system consists of an upper ontology, that contains a high level definition of the business concepts used in the integration domain and lower ontologies that lift the database schema to a resource description framework (RDF) format [RDF]. The upper and lower ontologies are connected using mappings based on the INRIA [Euzenat 2004] mapping format. The lower ontologies are connected to the data sources using the D2RQ API [D2RQ API].

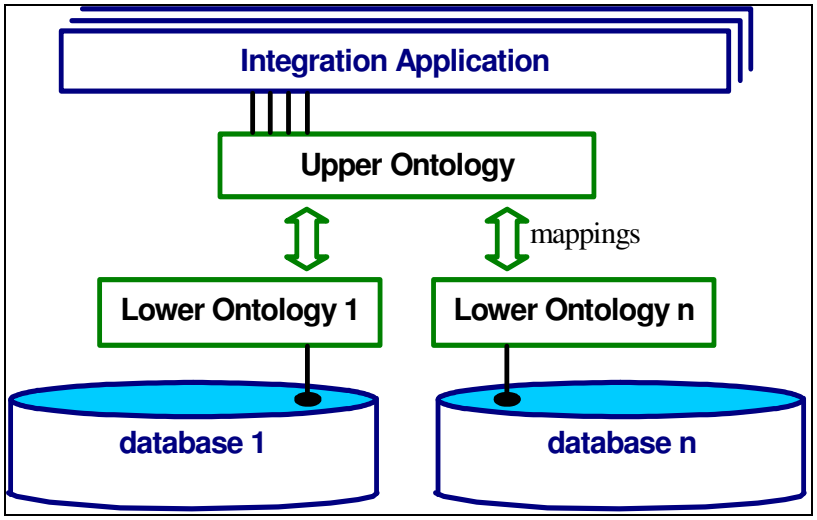

Figure 3-19 Integration Test Bed Overview

Upper Ontology

The upper ontology can be developed by gathering information about each domain from domain professionals. The approach taken in experiment one and two was to have each professional summarise their domain understanding in a short précis. These descriptions were used to create a common view of the domain.

By extracting the concepts and relations described in the précis, an ontology can be developed in OWL [OWL] using the Protégé development kit [Protégé].

Ontologies are instantiated in the integration application using the Jena API [Jena].

Lower Ontologies

The lower ontologies lift the basic database schema information into RDF using D2RQ API [D2RQ API]. This allows for automatic generation of the ontologies from the databases and once instantiated in a JENA model, the lower ontologies can be queried using SPARQL. The D2RQ API automatically converts the SPARQL queries to SQL and returns a set of triples to the caller.

The lower ontologies contains classes and properties for each of the underlying database schema items and are accessed through a set of mapping files automatically created by the D2RQ API.

Mappings

A bespoke mapping implementation was created by the author of this thesis that is based on the INRIA format but additionally allows a Java function to be called to execute a complex mapping.

The mappings used in this prototype support simple equivalence mappings (class to class, property to property), union type mappings (property A is the union of property B and property C) and complex conversion mappings (property A can be converted to property B using relation AB). In this prototype, relations are encoded as standalone Java functions.

A complex mapping (to sum three revenue fields into one) with a function specified looks like:

```
Entity1=http://someUrl/upperontology/#forecast_reveneue_q1

Entity2=http://someUrl/lowerontology/#forecast_revenue_m1,
        http://someUrl/lowerontology/#forecast_revenue_m2,
        http://someUrl/lowerontology/#forecast_revenue_m3,

Relation=function
FunctionHandle=sum_revenues
```

Code 7: Mapping Specification

This mapping method was realised using an XML format. The integration system executes the mappings in a predefined way as shown later in this chapter.

The XML realisation of this mapping scheme requires that each mapping has the following tags:

- Source ontology property name
- Destination ontology name
- Destination properties name
- Destination instance access information (Optional)

Some properties require “instance access” information. For example, to access a customer name, we need to know its customer id. This link information is also with the mappings (but raises some database schema knowledge into the mappings).

A simple example of a mapping in XML follows:

```
<mapping>
        <mapping_number>9</mapping_number>
        <source_type>p</source_type>
       <source_name>customers_region</source_name>
       <dest_ont>sales</dest_ont>
       <dest_type>p</dest_type>
       <dest_prop_name>region</dest_prop_name>
       <dest_class_name>customers</dest_class_name>
       <dest_pkey>id</dest_pkey>
</mapping>
```

Code 8: XML Mapping snippet

This mapping means that the "Customers_region" property from the upper ontology is mapped to the "region" property in the ontology "sales" and to access a region you need to have a "customers id".

A complex mapping is show below:

```
<mapping>
        <mapping_number>19</mapping_number>
        <source_type>p</source_type>
       <source_name>sales_revq1</source_name>
       <dest_ont>forecasts</dest_ont>
       <dest_type>p</dest_type>
       <dest_prop_name>revm1,revm2,revm3</dest_prop_name>
       <dest_class_name>forecasted_items</dest_class_name>
       <dest_pkey>oppid</dest_pkey>
       <function>sum_revenue</function>
</mapping>
```

Code 9: XML mapping snippet

This mapping means that the "sales_rev_q1" property from the upper ontology is mapped to the "revm1, revm2, revm3" properties in the ontology "forecasts" and to access it requires an "oppid" and execute the "sum_revenue" function.

Functions are implemented using Java dynamic class loading and an interface class as shown below.

```
public interface mappingif {
      public String convert(…);
}
```

Code 10: Java Code snippet - Dynamic Class Loading

For each complex mapping in the mapping file, a class must be created that implements the "convert" method from the interface class.

Mapping functions are dynamically loaded and executed by the integration system as follows:

```
URLClassLoader loader = null;
loader = new URLClassLoader(new URL[] {file.toURL()});
Class c = loader.loadClass(funcname);
Mappingif var = (mappingif) c.newInstance();
var.convert(...);
```

Code 11: Java Code snippet - Dynamic Class loading

The "funcname" is loaded from the mappings XML description.

This method of dynamic loading and mapping function naming is very flexible because the existing mapping functions can be updated without interfering with the mappings file or the main integration system code. New mapping functions can be added to the system by updating the mapping file.

Ontology and Database Query

Ontologies are instantiated in the integration application using JENA API [Jena]. The ARQ (SPARQL) API [SPARQL] is used to generate queries on the upper and lower ontologies.

3.3.3 Functional Architecture & Design

The functional architecture for generalised ontology-based integration system, called HotFusion, is shown in

Figure 3-20.

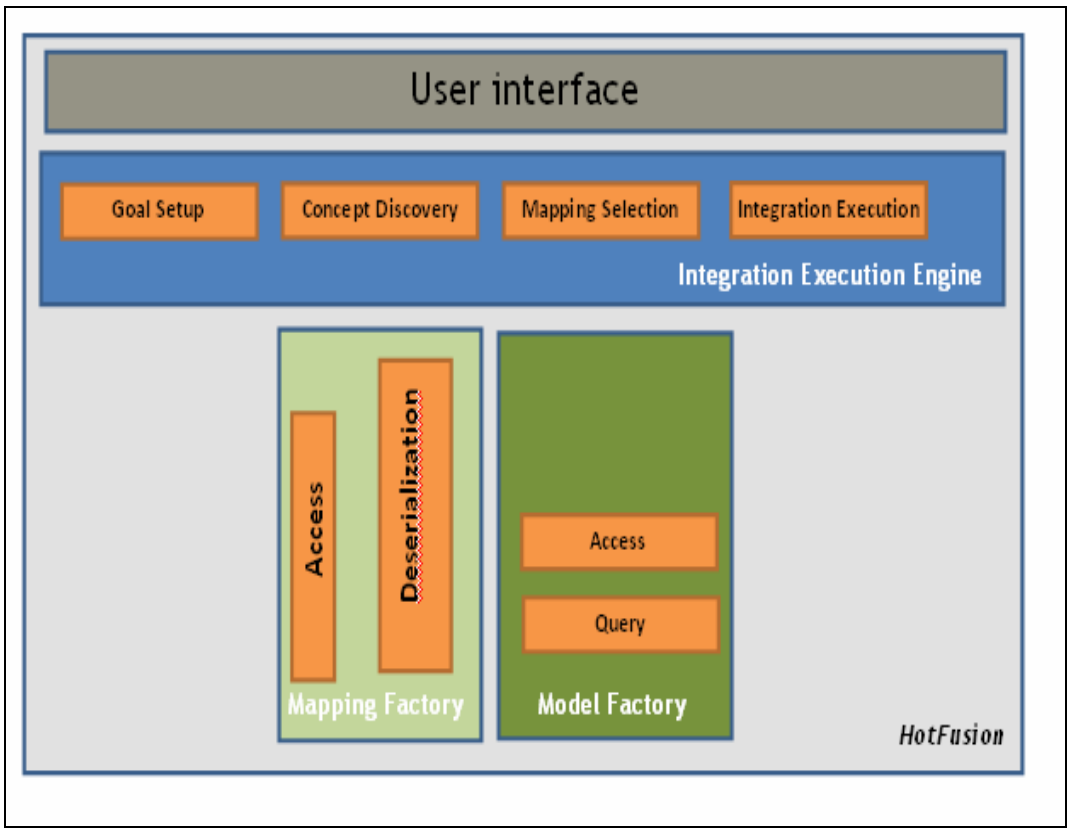

Figure 3-20: Integration System Functional Architecture (HotFusion)

To fulfill design requirement 2 from Section 3.3.1, the system was divided into the following functional areas:

- **Mapping Factory:** The mapping factory is responsible for loading and processing the mapping file for the integration system. The classes in this functional area convert the XML mapping file into an internal Java list structure to enable fast searching over the mappings. This class is very similar to the mapping factory class used in the TomE tool described earlier with respect to the loading of XML mappings. The main difference between implementations is that the search functions have different functionality based on different needs of the TomE tool and integration test bed.

- **Model Factory:** The model factory is responsible for loading, verifying and querying the upper ontology. It provides functionality to the Integration Execution Engine that is described next.
- **Integration Execution Engine:** The integration execution engine executes a series of steps that are described in Figure 3-21. This provides the functionality need to fulfil design requirement 1 from Section 3.3.1.
- **User Interface:** A tabbed graphical user interface is provided by this functional area. This provides control over the execution of the integration use case by allowing the user to execute the integration in a series of steps using buttons on the user interfaces.

The integration engine carries out the integration using the steps defined in Figure 3-21.

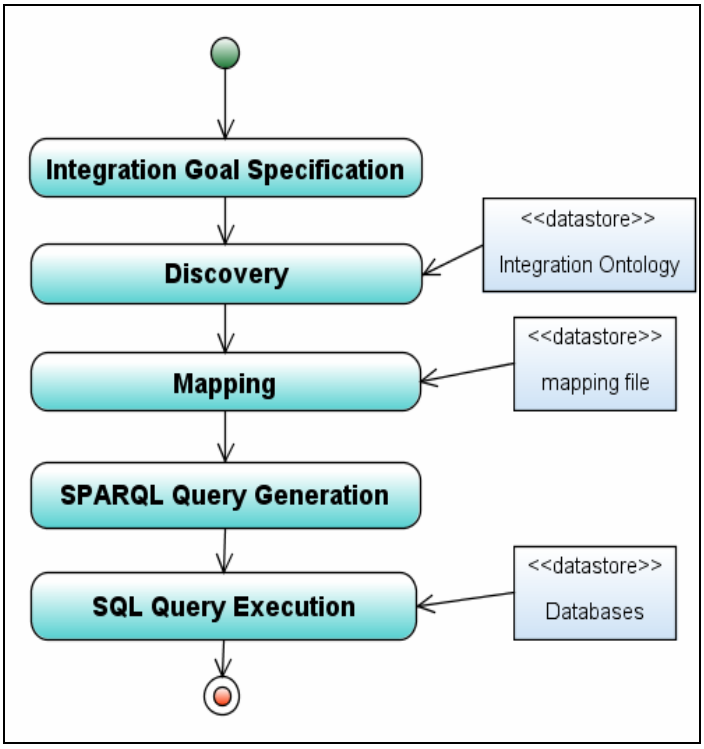

Figure 3-21: Integration Process

- **Integration Goal Specification**: The user or application specifies an integration goal. In the test system, the goal is hard coded into the application. The integration goal specifies what the users or applications wish to integrate and contains the concepts to integrate and the data needed to select the information (the key information).

- **Concept Discovery:** Using a SPARQL query [SPARQL] on the upper ontology, each concept in the goal is supplemented with the properties available for that concept. (e.g. customer_info concept 'becomes' customer_name, customer_id, customer_region etc…)
- **Mapping Execution:** The mappings are now applied to the concept and property names. This step then generates SPARQL queries on the lower ontologies.
- **Lower Ontology Query:** Output from the mappings step is a sequence of SPARQL queries that are run against the lower ontology. These queries are in turn converted to SQL queries by the D2RQ API [D2RQ API]
- **Presentation of Results:** Each requested property and the properties value is returned to the application. In our test system, we have no semantics to help us construct a formatted report so a simple list of attribute names and values are returned.

3.3.4 HotFusion Implementation

This section describes the implementation details of the mapping factory, model factory and user interface implementation.

3.3.4.1 Mapping factory

The mapping factory consists of a set of classes that provide access and manipulation of the mappings that have been described earlier. The component provides methods to load mappings from a specified file. Mappings are stored in memory during the integration process in a Java array list structure. The class diagram for the mapping factory is shown in Figure 3-22.

The class UCTest3Panel is the user interface class that is used to orchestrate the integration steps as shown earlier (Figure 3-21).

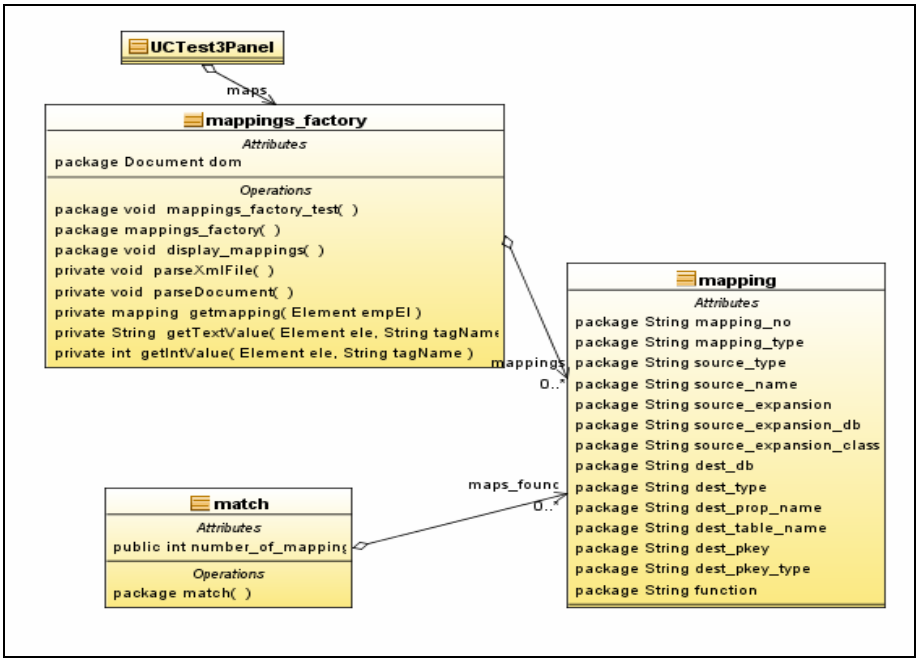

Figure 3-22: Class Diagram Mapping Factory

3.3.4.2 Model Factory

The model factory consists of a set of classes that provide methods for loading, verifying and querying integration ontology (upper ontology).

The class diagram for the model factory is shown in Figure 3-23

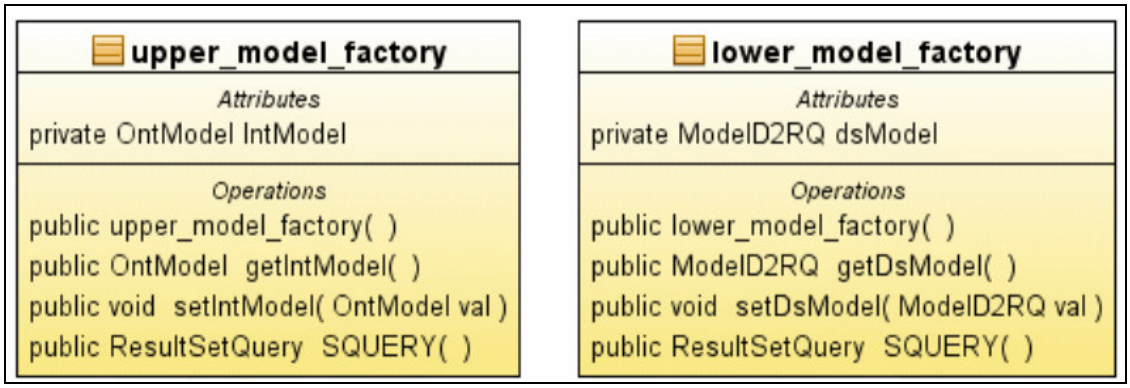

Figure 3-23: Class Diagrams for Model Factories

The model factory loads the integration ontologies using Jena API [Jena]. The integration is stored in memory as a Jena OntModel Object.

The model factory also provides classes to load the lower ontologies. Each of the lower ontologies are loaded as D2RQ objects "ModelD2RQ" from the D2RQ API [D2RQ

API]. Note that the integration code will need to load one lower ontology model for each data source that is included in the integration system.

In the current implementation of the HotFusion, these object references are managed in user interface code.

3.3.4.3 User Interface

The user interface (Figure 3-24) classes provides a set of Java tabbed panes to allow independent setup, execution and monitoring of the different use cases (e.g. logistics for experiment three).

In the current implementation of HotFusion, the orchestration of the use case needs to be manually coded in the button action handler of the appropriate tabbed pane.

This approach was taken to simplify coding and ensure focus on what the integration steps are and not how they might be automatically orchestrated (e.g. using some orchestration or workflow approach).

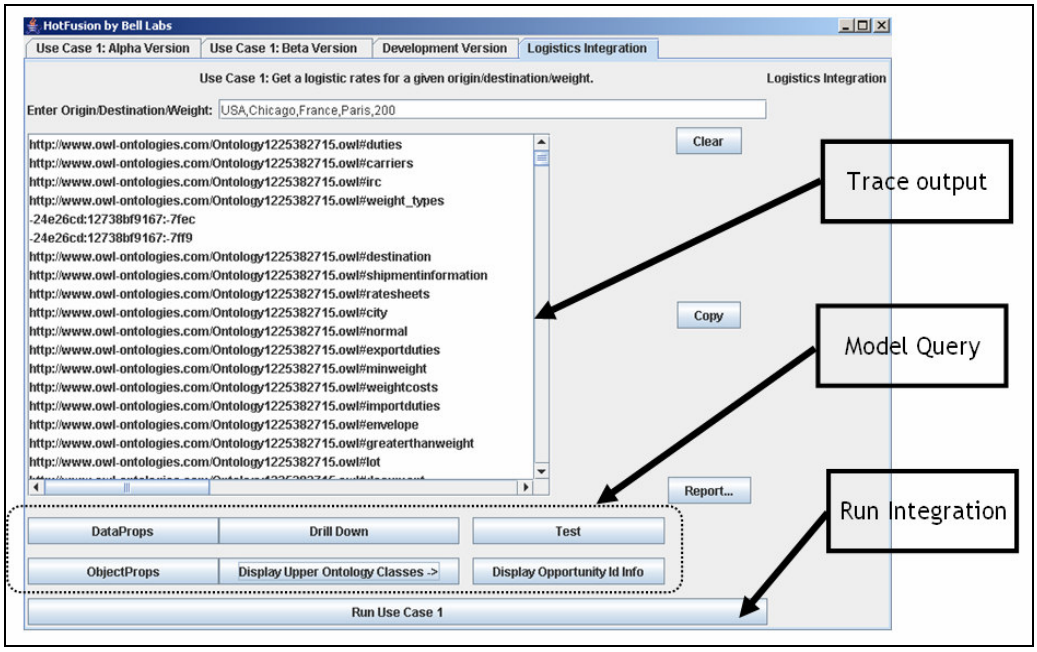

Figure 3-24: Integration System Control Panel (HotFusion)

3.4 Summary

This chapter described the design and implementation of the models, tools and test beds that were created to support the dependency modelling approach taken in this research. Chapter 4 describes how these models, tools and test beds were used to support the evaluation of the research question.

4 EVALUATION

4.1 Overview of Experiments

Four experiments and one corroborative study were conducted in keeping with the action-based research methodology of this research. Each experiment supports the analysis and evolution of the research question. The four experiments deal with the development and evaluation of an approach to support the evolution of mappings in an ontology-based integration system. The corroborative study provides an indication of the genericity of the ontology-based dependency metamodel by applying the metamodel in another domain.

Figure 4-1 provides an overview of the five experiments and the measurement approaches applied to them.

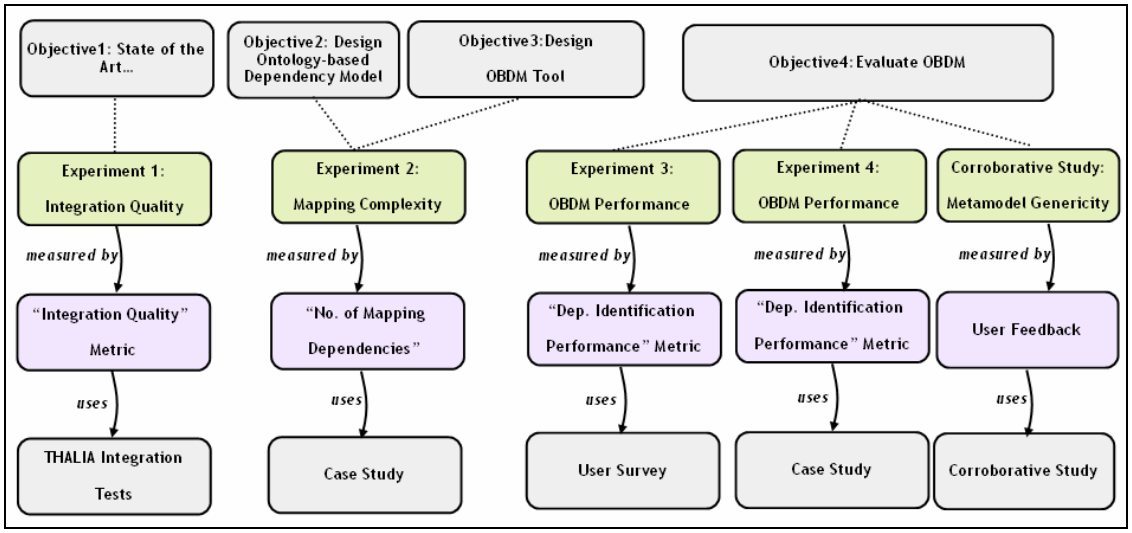

Figure 4-1: Relationship between Experiments and Objectives.

The first experiment developed an understanding of the integration performance and issues associated with ontology-based integration systems. The experiment created an environment to measure the performance of a generalised ontology-based integration system. To achieve this, a generalised ontology-based integration system test bed was created. The integration test bed used data from product line management systems from the Alcatel-Lucent supply chain. The "integration quality" metric, as defined in Section 1.2, of the system was measured using the THALIA [Stonebraker 2005] integration benchmark. In summary, experiment one showed that advantages are

gained by using the ontology approach because the solution can cope with semantic heterogeneity using mappings. However the analysis of the results of experiment one also showed that the mappings themselves create complex couplings between the components of the integration system. Experiment one led to the hypothesis that the complex nature of the mappings makes it difficult to quickly and accurately find the mappings that are impacted when a data source changes. Experiment two evaluated the hypothesis that this difficultly in managing changes to the data sources and mappings could be alleviated by modelling the dependencies that exist between the components of the ontology-based integration system.

The second experiment developed and evaluated an ontology-based dependency model (OBDM) that would support understanding of the complex coupled nature of mappings. The second experiment used mappings which resulted from the application of the generalised ontology-based integration test bed. The generalised ontology-based integration test bed was used to resolve the heterogeneities associated with data from the Alcatel-Lucent logistics supply chain. This experiment confirmed the hypothesis arising from experiment one regarding the complex nature of mappings by showing the complex dependent relationships they exhibit with other parts of the system. From analysis of the results of experiment two, the hypothesis was developed that the mapping dependency relationships are difficult to identify without tool support. This hypothesis is confirmed in experiment three.

The third experiment evaluated the performance of the dependency modelling approach in terms of the accuracy and time taken to find answers in comparison to a manual process oriented approach that does not have tool support. The manual approach represents the key steps needed to find dependencies in semantic mappings. The manual process was developed the author of this thesis by interviewing integration and logistics experts to identify the key processes required to find the dependencies within a theoretical set of mappings based on logistics data. A group of 18 users were given an exercise to find predefined dependencies between sample mappings using the manual process. A theoretical set of mappings was used during the experiment to ensure an even distribution of the complexity of the answers across the exercises. Metrics were collected during the exercises to enable statistical analysis of the time taken and accuracy of users as they executed the process. The results of the evaluation

show that the ontology-based dependency model significantly outperforms the manual process in both accuracy and time.

In experiment four the ontology-based dependency model and tool was used to support the evolution of the mappings when performing a real evolution based on an industrial dataset. For this experiment, a new logistics carrier was introduced that provides transportation services by sea. This required the analysis and updating of the mappings used in experiment two. A case study was carried out that identified the dependencies that arise when the new logistics data source was added to the integration system from experiment two. The results of experiment four demonstrate how the ontology-based dependency model can support mapping evolution by showing the dependencies that exist thus allowing the user to examine the dependency relationships in detail.

Finally, to provide an indication of the genericity of the ontology-based metamodel, a corroborative study was carried out to test the ability of the metamodel to be applied outside the ontology-based integration system. A scoped electrical circuit from a domestic setting was selected as the domain and an electrical engineer was instructed on the process to create a domain model from the metamodel. A sample dependency analysis was run on the model based on the requirements of the electrical engineer. The dependency analysis supported the identification of faults in the circuits based on the analysis of dependent components in the circuit.

4.2 Experiment One – Measurement of "Integration Quality" Metric

4.2.1 Overview

During this experiment the technical environment to measure the performance of a generalised ontology-based integration system was created. The technical environment consisted of the generalised ontology-based integration system test bed and the THALIA [Stonebraker 2005] integration benchmark. The integration test bed was populated with data from product line management systems based on a use case from the Alcatel-Lucent supply chain. The THALIA integration benchmark was used to measure the "Integration Quality" metric of integration system performance as defined in the introduction (Section 1.2).

In summary, experiment one showed that while advantages are gained by using the ontology approach because the solution can cope with semantic heterogeneity using mappings. However, as a result of experiment one a hypothesis was developed that suggested that the complex nature of the mappings makes it difficult to quickly and accurately find which mappings are impacted when a data source changes

Section 4.2.2 describes the objectives of the experiment in the context of the research question.

Section 4.2.3 provides the background to the supply chain based use case that was used for this experiment.

Section 4.2.5 describes in the detail the approach taken for this experiment.

Section 4.2.6 and 4.2.7 describe the results and conclusions of this experiment.

4.2.2 Objectives & Hypotheses

In the Introduction chapter, the first objective that was derived to evaluate the research question was stated as "Perform a state of the art review of approaches for semantically linking local[14] schema and aggregate or global schema[15]". To address this, experiment one derived the following sub-objectives:

i) Identify the generalised ontology-based integration approach using a literature review to identify the different approaches to support integration.

[14] Local schema refers to a schema that represents the local sources to be integrated.
[15] Global schema refers to a common view of sources to be integrated.

ii) Assess the integration performance of this approach using the generalised ontology-based test bed and THALIA integration benchmark.

iii) Identify the issues that would affect upon an industrial deployment.

Hypotheses

From the state of the art review, the hypothesis was developed that ontology-based integration approaches can support the semantic integration of heterogeneous data sources. This experiment tests how well this hypothesis holds when using an industrial use case and data set.

4.2.3 Use Case Background

Supply chains of large companies are typically comprised of many IT systems that have developed over time to support various supply chain functions (e.g. Customer Relationship Management, Demand Forecasting, Production, and Logistics). Each stage of a product's life is managed by one or more IT systems. While these systems have introduced many productivity improvements in their individual areas, they have also contributed to the creation of separate islands of data in the enterprise.

An important part of many supply chains is Product Lifecycle Management (PLM). PLM is a supply chain process that manages enterprises' products through all stages of their life from initial sales opportunity, demand forecasting, product realisation, manufacturing, delivery to customer and support to end of life. It is within this area of the supply chain that data consistency and visibility issues were identified between the systems that manage the Sales and Forecasting part of the product lifecycle. Lack of consistency can lead to failure to deliver on time or result in excess of inventory.

To mitigate any risk associated with lack of consistency between sales and forecasting views of the PLM, organisations attempt to balance forecasting and sales opportunities [Gilliland 2002]. In the Alcatel-Lucent supply chain, these risks are managed using a manual integration of financial information from each system. The report that is produced by this manual integration supplements the financial information with an integrated view of the customers and products. This involves many manual steps to export data from the databases and rework with a spreadsheet where the various heterogeneities are resolved manually.

4.2.4 Experimental Approach

Resulting from objective (i) for this experiment (as described above), a state of the art review of approaches to ontology-based integration systems was undertaken. The hybrid approach was selected, as discussed in the state of art review, because it offers potential improvements in implementation effort, support for semantic heterogeneities, adding, and removing of source over the single or multiple ontology approaches [Wache et al. 2001].

Resulting from objectives (ii) and (iii) for this experiment, a generalised ontology-based integration test bed was created, as described in the design chapter. The generalised ontology-based integration test bed was populated with data based on the use case from the Alcatel-Lucent supply chain as described in section 4.2.3. The THALIA integration benchmark [Stonebraker 2005] provided an ideal framework to measure the "Integration Quality" measurement. In the Introduction chapter, the "Integration Quality" measurement was defined as:

- A measure of the ability of the system to carry out integrations across a range of different types of data heterogeneity.

The THALIA benchmark system and tests specify 12 types of heterogeneity that can be used to test performance of integration systems. The approach taken to run the benchmark involved the following steps:

- Assess which THALIA tests are covered by the PLM data and manually add data for any tests not covered. (Section 4.2.5)
- Set up the integration test bed to carry out the integrations based on the 12 THALIA tests (Section 4.2.5)
- Run the integration test bed to generate the integrated report. (Section 4.2.6)

4.2.5 Experimental Setup

This generalised ontology-based integration test bed required the creation of the upper ontology, mappings and lower ontologies for this domain. The approaches adopted for these tasks are described below.

Integration (Upper) Ontology

The upper ontology (Figure 4-2) was developed by gathering information about each domain from three supply chain professionals, one working on forecasting, one working on sales and one working on the current manual integration of the systems. Each professional summarised his or her understanding of the domain in a short précis. These descriptions were used to create a common view of the sales and forecasting area. By extracting the concepts and relations described in the précis, an ontology was developed by the author of this thesis in OWL [OWL] using the Protégé development kit [Protege]. Ontologies are instantiated in the integration application using the Jena API [Jena]. The ontology contained 8 classes, 20 data type properties and 5 object properties.

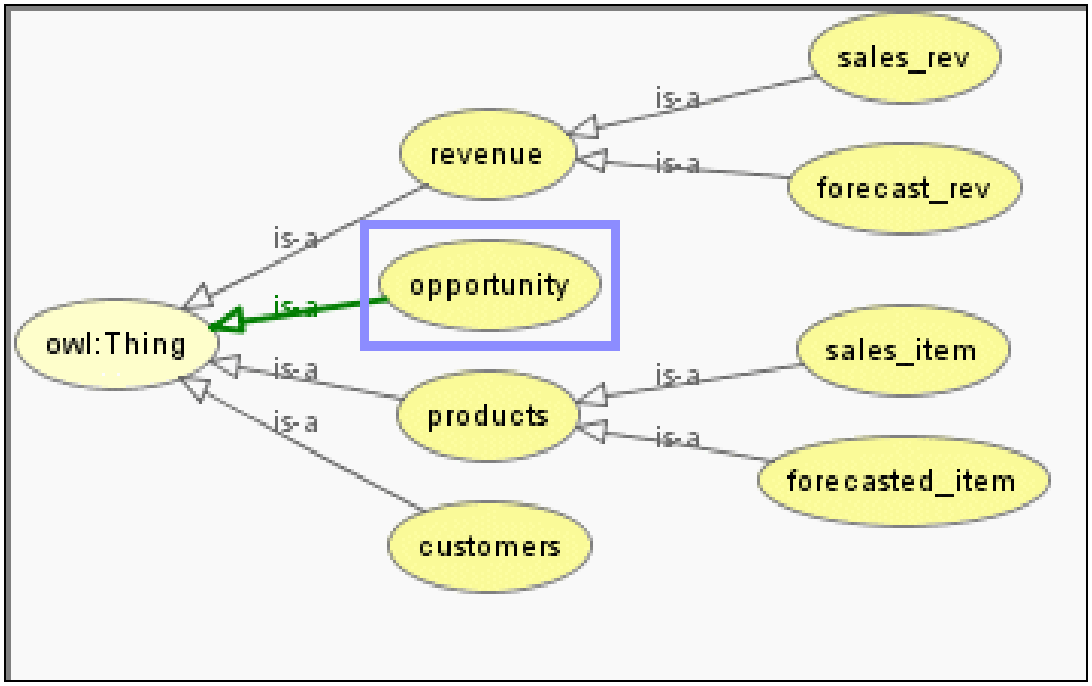

Figure 4-2: Excerpt from Upper Ontology

Creation of Mappings

The mapping format was described in the design chapter (Section 3.3). For this experiment, the relationships between the ontology concepts and database fields were identified during the ontology design process with the help of the domain experts. The mappings were then encoded manually into the mapping format. Thirty-one mappings were needed to implement the use case. The mappings are listed in full in Appendix II.

Ten mappings required complex conversion functions to cater for conversions of different product code formats, quarterly and monthly revenue figures, currency and date conversions. In experiment one; these are referred to as complex mappings as they contain a bespoke conversion functions coded in Java. The remaining 21 mappings did not require conversion functions as they lift the data from the databases without conversion.

Lower Ontologies

The lower ontologies were created automatically from the databases (described in the next section) using the D2RQ API [D2RQ API] as noted in the design chapter. The lower ontologies lift the basic database schema information into RDF (using D2RQ API.) This allows for automatic generation of the ontologies from the databases and once instantiated in a JENA model [Jena], the lower ontologies can be queried using SPARQL [SPARQL].

Database Setup

Two databases from the Alcatel-Lucent supply chain were chosen based on the use case requirements. The first is an Oracle based system that manages sales opportunities. It contains high level product and financial information and detailed customer information. This system has 58 tables and over 1200 attributes. The second system is a Sybase based system that manages product forecasting. It contains high level customer information but detailed product and financial information. This system has 50 tables with over 1500 attributes.

As these systems were very large and in active daily use, each database schema was examined to extract the tables and data that were relevant to the integration use case and this reduced data set was recreated in two MySQL databases. The integration use case enabled reduction of the original dataset (tables and properties) to only that data used in the use case. For example, one database also contains multiple levels of customer contact detail that is not relevant to the integration use case. This reduced the data sizes to 8 tables for each database. All schema and real data from the original databases were preserved in the MySQL versions. To allow the full THALIA benchmark to be run, the databases needed to be supplemented by additional complexity in three areas (language expression and virtual columns, nulls – see Table

2-1, Section 2.7.1). This was achieved by adding specific data items to databases to cover these tests.

The integration scenario involved the integration of financial information from each database, ordered by sales opportunity and supplemented with financial information with an integrated view of the customers and products. Real instance data from the Alcatel-Lucent supply chain systems was loaded into the MySQL database to run the use case.

The integrated report could be represented as shown in **Figure 4-3**

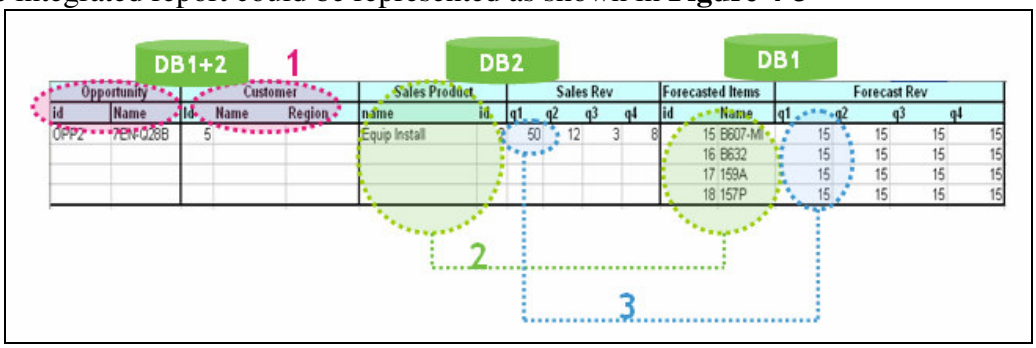

Figure 4-3: Integrated Report

Sales Opportunity (Opportunity) and Customer information (Customer) need to be expanded from high level concepts to contain more detailed information such as customer id and geographical region. This involves integrating information from both databases (identified as 1 in **Figure 4-3**).

The Sales view of a product needs to be expanded from a single high level item (Equip Install) to a collection of items (with ids 15, 16, 17, 18) in the Forecasted view (identified as 2 in **Figure 4-3**).

Sales and Forecasted revenue needs to be converted from monthly to quarterly views (identified as 3 in **Figure 4-3**).

The key heterogeneities that exist in the underlying data:

- Structural – Simple conversions

  Example 1: currency units in one schema need to be converted to a different unit in the second schema.

- Structural – Relations

A single product (high level description) in one schema is represented by a list of parts (low level description) in the second schema. For example, a product at the sales database is defined as “ADSL Access Platform”, in the forecasting database this is broken down into many parts (frames, cards, cabinets)

- Structural - complex conversions

  Example 1: Revenue figures in one schema are stored monthly compared with quarterly revenue in other schema. The upper ontology deals with quarterly revenue and a conversion (summing) of monthly to quarterly revenue needs to occur.

  Example 2: “Long codes” used in one schema are comprised of three subfields in the second schema.

- Semantic - Different class and property names conveying same information

  Example 1: Upper ontology has a class called “customers” with properties “name”, “id” and “region”. Lower ontologies have classes “custs”, “account” and properties “name”, “id” and “FTS-Tier”

- Semantic - Same property name conveys different information

  Example: product_id is used in both the lower schemas but conveys different information with different granularity

4.2.6 Experimental Results (based on THALIA)

This section contains the results related to the objective to measure the performance of our approach based on the THALIA integration benchmark (objective ii).

With respect to the THALIA integration benchmark, using the generalised approach, 50% automated integration (6/12 tests passed) was achieved. A test was deemed to have passed if the integration test bed could perform the integration in at least a semi-automatic way. Table 4-1 below shows the detailed results:

Test	*Result*	*Effort*
1. Synonyms	PASS	Semi Automatic
2. Simple Mapping	FAIL	Manual
3. Union Types	PASS	Semi Automatic
4. Complex Mapping	FAIL	Manual
5. Language Expression	PASS	Semi Automatic
6. Nulls	PASS	Fully Automatic
7. Virtual Columns	FAIL	Manual
8. Semantic Incompatibility	PASS	Semi Automatic
9. Same Attribute in different Structure	FAIL	Manual
10. Handling Sets	FAIL	Fail
11. Attribute names does not define semantics	PASS	Semi Automatic
12. Attribute Composition	FAIL	Manual

Table 4-1: THALIA Integration Benchmark Results

Efforts are categorised as follows:

- Fully automatic: no code, mapping or ontology changes needed.
- Automatic: Automatic regeneration of ontology or other configuration artefact.
- Semi Automatic: A mapping needs to be changed manually.
- Manual: Non core code artefact needs to be changed or added manually. (e.g. a function associated with a mapping)
- Fail: core code changes needed. (e.g. core test bed code needs to be changed)

(Note: this is an extended method of classification that is not part of the core THALIA system)

In total, 31 mappings were needed to implement the use case. Of these, 21 mappings were simple (e.g. point to point relations) between ontologies and the remaining 10 were complex mappings requiring supporting 'function code' to be written.

As Table 4-1 indicates, tests 2,4,7,9 and 12 fail. This was because they required conversions to be constructed, that in turn required some mapping code to be produced. Examples of these are:

- In one schema, product id is encoded in a longer representation called "longcode" and the product-id needs to be extracted (test 7).

Tests 1,3,5,8 and 11 require a mapping to be created that does not require any mapping conversion function to be written.

Examples of these are:

- customer_name in one ontology is mapped to cust_name in another (test 1)
- product_description in the upper ontology is the union of product information in the lower ontologies (test 3).
- customer_region in one ontology is mapped to "client" (test 5)

Test 10 fails outright because the mapping format used did not support such "set type" constructs. To support this would require changes to the integration system code itself in the way its handles the mappings (i.e. the semantics of the mappings would need to change).

4.2.7 Discussion of Experimental Results

The goal of experiment one was to discover the key issues when applying the ontology-based integration approach using the THALIA benchmark. The integration performance and issues found are discussed below.

Integration Performance

The results of applying the THALIA benchmark show the complexity of carrying out integration of heterogeneity data sources. The score achieved of 50% automated integration reflects the variety of challenges that the THALIA tests provides.

Extra support could be provided during the mapping process to simplify the effort needed for test 2 (simple mapping), test 4 (complex mapping) and test 9 (Same Attributes exist in different structures). This could be achieved by providing the user with a set of well-known conversions for important (or often used) data types. To incorporate these conversions in the integration system would require update to both the mapping structure (to identify which canned conversion to use and specify how to apply it) and to the integration test system to enable execution of the appropriate conversions.

To improve the performance of the system for test 7 (Virtual Columns), the ontology design process could be improved to explicitly identify data of this type in the data sources and then to make the relationships explicit in the ontology.

It is difficult to envisage how the current integration test bed could be improved for test 10 and 12 without additional coding of new functionality for the integration test bed.

While every data integration use case may not contain each type of heterogeneity specified by THALIA, as the new integration use cases are added, new heterogeneities can be expected due to different data designs implemented in the underlying databases.

The goal of the integration test bed was only to provide insight into the integration system and mapping complexity and thus the improvements proposed above have not been subsequently implemented.

Mappings create tight coupling.

A third of the mappings used to resolve the heterogeneities in our database were of the complex type (tests 2, 4, 7, 9, 12). Unfortunately, these mappings create a tighter

coupling between the upper and lower ontologies than is desirable because the conversion functions that need to be written tend to require information from both the upper and lower ontologies. For example, a non trivial conversion function is needed to sum the revenue for three months from the lower ontology into a quarterly value for the upper ontology; however, the function specification for this summation needs to know from which lower ontology resource to obtain the monthly value.

Furthermore, the number of mappings required will grow as different integration use cases are implemented because different data properties may need to be mapped between the lower and upper ontologies. This is problematic because it will require the user to remember which mappings have already been implemented and what complex conversions the mappings use. If the integration use case requires a specialisation or generalisation of an existing concept in the domain ontology then as new mappings are created they may well refer to some of the database properties used by existing mappings. This is also problematic because it will require the user to develop an understanding of the relationships between mappings, the data source and ontologies.

The abstraction level of the upper and lower ontologies also negatively impacts the coupling. At the lower ontology, there is a low abstraction (few semantics) ontology and at the upper ontology there is a high abstraction (domain conceptualisation). This forced some aspects of the integration to be resolved in the application and not in the ontologies or mappings. For example, there are a number of cases where a property could be used to find other properties ("opportunity id" allows us to find a "customer id" and that allows a customer name to be found). However, given the "opportunity id", this linkage is not encoded currently in the ontology or in the mappings.

Reasoning in the upper ontology

The integration test bed was designed to support integration by traversing the ontology (via its OWL object properties) and use the associated mappings to build integrations of the data that the ontology represents. While this represents reasoning over the ontology, it did not use a separate ontology reasoner (e.g. the Pellet reasoner [Pellet]) to carry out this functionality since it requires also query functionality over the ontology. This is implemented in the integration test bed by a piece of Java code and used the SPARQL API to query the domain ontology. This approach means that the

code is reusable over any other domain ontology. The Pellet reasoner was used to carry out validation of the integration ontology during both the ontology design process and when the ontology is instanced in the integration system using the JENA API [Jena].

4.2.8 Summary of Conclusions, Open Issues and Limitations

Experiment one measured the performance of the ontology-based integration test bed using an industrial use case and data set and the THALIA benchmark system. The integration system achieved 50% automated integration. This score reflects the variety of challenges that the THALIA tests provides.

From analysis of the results of this experiment, a hypothesis was developed that mappings exhibit complex dependency relationships with the data sources and ontologies. Furthermore support for understanding the mapping dependencies would bring benefits to the integration system when the mappings need to be changed.

While the THALIA score could be improved by further development of the ontology-based integration test system, a more important issue for an industrial deployment was identified concerning how the mappings can be evolved as new integration use cases are added to the system.

The THALIA benchmark provides a simple measure (i.e. a score out of twelve) of the ability of the system to perform integrations across the twelve types of heterogeneity. A more comprehensive suite of performance measurements (e.g. runtime performance) would be needed to confirm the integration systems suitability for industrial deployment. These aspects of performance were not tested in this research as the focus here was to investigate the complexity of the mappings.

The THALIA benchmark system does not provide quantitative data on how much effort is needed to run each test. This is important because, while a test may pass, it may require costly manual intervention (e.g. mappings updates) that would impact the scalability of the system. To address this in experiment one an effort classification was developed and used that provides qualitative estimation of the effort needed for each test in THALIA.

4.3 Next Steps in Action Methodology

From analysis of the results of experiment one, the hypothesis was developed that the mappings exhibit complex dependency relationships with the data sources and ontologies. Furthermore support for understanding the mapping dependencies would bring benefits to the integration system when the mappings need to be changed

The next iteration in the action-based methodology was to create the ontology-based dependency model to develop an understanding of the dependencies between mappings, data sources and ontologies to confirm this hypothesis.

4.4 Experiment Two – Mapping Complexity Analysis

4.4.1 Overview

This experiment created the technical environment to support the analysis of the complexity of the mappings in the generalised ontology-based integration system. This was achieved by modelling the dependencies that existed between the mappings in the generalised ontology-based integration system developed as part of experiment one. A model of dependencies in the ontology-based integration systems was developed in OWL [OWL]. This model was called the ontology-based dependency model (OBDM) as described in the design chapter. A tool called TomE (Towards Ontology Mapping Evolution) was created to load the OBDM and support the analysis of the dependencies.

The mappings used in the generalised ontology-based integration system for this experiment came from a second use case from the logistics domain of the Alcatel-Lucent supply chain. The new use case for this experiment required the creation by the author of this thesis of a new integration ontology for the logistics domain, mappings and lower ontologies. These were created using the same approaches as described experiment one.

From the analysis of the mappings in this experiment, it was found that 30% of the mappings exhibit complex dependency relationships with other mappings. It was hypothesised that these complex relationships are difficult to identify without tool support and thus makes the first step of mapping evolution difficult for integrators.

Section 4.4.2 describes in the objectives of the experiment in the context of the research question.

Section 4.3.3 provides the background to the supply chain based use case that was used for this experiment.

Section 4.4.4 describes in the detail the approach taken for this experiment.

Section 4.4.6 and 4.3.7 describe the results and conclusions of this experiment.

4.4.2 Objectives & Hypotheses

In the Introduction chapter, the second and third objectives that were derived to evaluate the research question were stated as:

- "Research and develop a model to define the dependencies that arise when creating semantic links between schemas to support an ontology-based integration approach between local schemas to global schemas."
- "Research and develop a prototype tool capable of supporting this dependency modelling approach."

To address these objectives, experiment two derived the following sub-objectives:

i) Develop and evaluate an ontology-based dependency model (OBDM) that would support understanding of the complex coupled nature of mappings.

ii) Confirm the hypothesis from experiment one that the mappings exhibit complex dependencies relationships with the data sources and ontologies.

Hypothesis

The hypothesis for experiment two was that the complex nature of the mappings in the generalised ontology-based integration makes it difficult to quickly and accurately find which mappings are impacted when a data source changes.

4.4.3 Use Case Background

For this experiment, the ontology-based integration systems from experiment one was tested with a new dataset from the Alcatel-Lucent reverse logistics supply chain[16]. The ontology-based integration system was used to replace a manual database update process for a logistics optimization tool developed in Alcatel-Lucent called ALTO[17]. ALTO is an enterprise system that generates simple cost optimized routing instructions called routing guides. These routing guides specify the lowest cost logistics company

[16] Reverse Logistics is responsible to repair and return of faulty equipment to customers.
[17] Alcatel-Lucent Transport Optimization (ALTO) is deployed in reverse logistics supply chain.

and service to use for any user specified origin/destination/weight. ALTO stores the rates for all logistics carriers in a relational database that was created using ETL[18] techniques. New logistics carriers and rate updates for existing carriers need to be incorporated into the ALTO database regularly. To simplify the database update process (Figure 4-4), the generalised ontology-based integration platform was tested against this logistics use case to integrate the heterogeneous carrier rates formats into a single common model of logistics. From the central model, the scripts to load the ALTO database could be automatically generated.

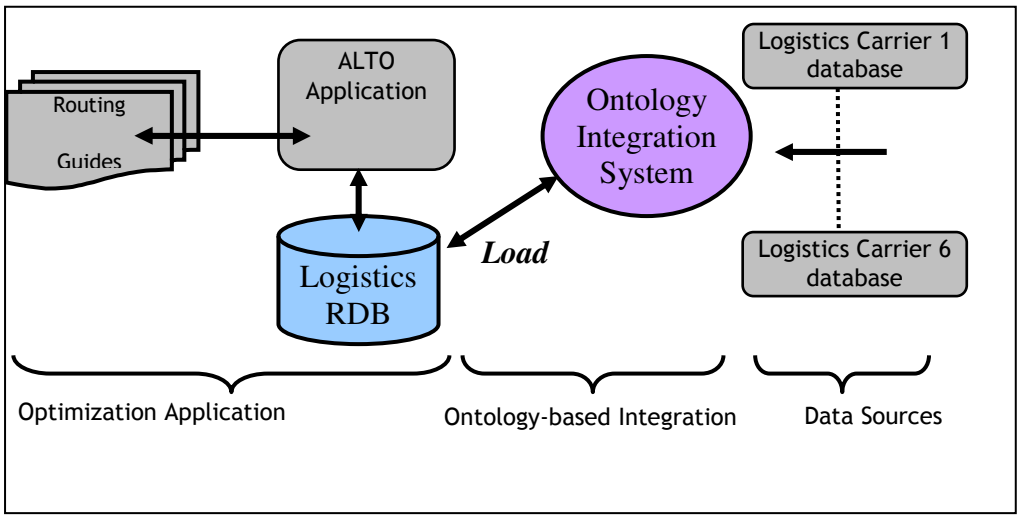

Figure 4-4: Logistics Rates Integration and Optimisation Applications.

4.4.4 Experimental Approach

To achieve objective (i) for this experiment (as described above), a domain specific model in OWL [OWL] to represent the dependencies in the system was developed. This is called the ontology-based dependency model (OBDM). The ontology-based dependency model was created using a metamodelling approach as described in the design chapter.

A tool called TomE (Towards Ontology Mapping Evolution) was developed to instantiate the OBDM and to support the analysis of dependencies in the ontology-based integration system.

Resulting from objective (ii) for this experiment, the ontology-based dependency model was used to carry out analysis of the inter-relationships that the mappings exhibit in the logistics based use case presented in Section 4.4.3

Experiment two required the following steps to be carried out:

[18] Extract, Transform and Load

- Setup the generalised ontology-based integration system to carry out integrations based on the logistics based use case. (Section 4.4.5)
- Carry out an analysis of the mapping dependencies using the ontology-based dependency model (OBDM) (Section 4.4.4.1)

4.4.4.1 Dependency Analysis Approach using the OBDM

The TomE tool provides tool support for the analysis of dependencies in the generalised ontology-based integration system. To carry out analysis of the dependencies the user must navigate a set of graphical tabbed panes in TomE. The tabbed panes are called "Mapping Control", "Ontology Control" and "Visualisation" as shown in the design chapter (Chapter 3, Figure 3-16: TomE Control Panel). This sequence of steps taken to analyse the dependencies is described below.

Step 1 - Use "Mapping Control" tab to Load mapping file

Using the "mapping control" tab, the user can select and load the mapping file.

Step 2 - Use "Ontology Control" tab to generate dependencies.

Using the "ontology control" tab, the user loads the dependency model and model instances (that are generated from the mapping file). The system tend computes the dependencies for each elements in the system and generates GraphML data for the dependencies.

Step 2 - Use "Visualisation" tab to launch visualisation

Using the "Visualisation" tab (Figure 4-5) the user can launch the three types of graphical views of the dependencies as described in the design chapter.

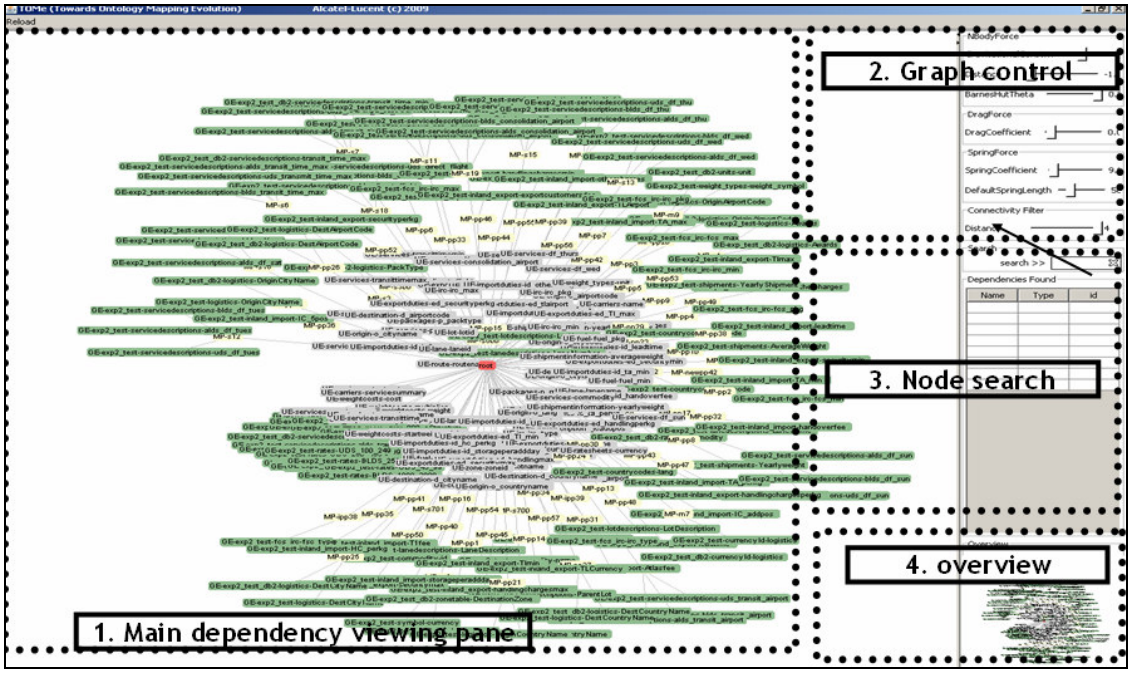

Figure 4-5: Dependency Visualization in TomE

4.4.5 Experimental Setup

To create the mappings to load into the TomE tool, the generalised ontology-based integration test bed needed to be updated with new upper ontology, mappings and lower ontologies for this logistics based use case. The approaches adopted for these tasks are the same to those adopted for experiment one and are described briefly below.

Database Setup

Although the ALTO system incorporates logistics rates from six databases, each database represents the data from logistics providers that are either door-to-door (DTD) services or airport-to-airport (ATA) services. In the ALTO system, four logistics companies provide ATA services and two provide DTD services. The schema for each the four ATA databases and two DTD databases were very similar. To avoid the creation of 6 similar databases for the generalised ontology-based integration system, it was decided to use one ATA and one DTD database. By selecting one ATA and one DTD type dataset, all of the key heterogeneity was preserved. This step reduced the number of databases in the ontology-based integration system from six to two for this experiment. For this logistics dataset, the integration system needed to setup semantic mappings between the integration ontology and the data sources to resolve the

heterogeneities in the data sources. A sample of the heterogeneity in the databases is given below:

- ***Service definition***: Services exhibit generalization conflicts.
- ***Destination country specification***: Some logistics groups use zones to represent a group of destination countries.
- ***Weight specification***: Conflicts between unit specifications, single and range weight specifications.
- ***Import/Export Charges***: Semantic conflicts between the definitions of terms with the same names.

The ontology-based integration system produces data integrations shown in MS-Excel format below. (Note: All costs are normalised to USD based on a hand coded exchange rate.)

Lot/Lane	Origin			Destination			Service				Cost	Weight	Surcharges			
	Country	City	Airport	Country	City	Airport	name	description	ptype	commodity			fuel	irc	ie	ii
"1-1"	France	Paris	CDG	USA	Chicago	ORD	Carrier1	Express	Package	Duty	157	20	4	12	12	12
							Carrier2	ALDS	Package	Duty	127	20	0	12	12	12

Figure 4-6: Logistics Report

Integration Ontology

The domain (upper ontology) for this case is described below (Figure 4-7). This ontology was constructed to enable the collection of rates information from each logistics carrier for any weight and is not designed to be full domain ontology for logistics. The ontology was developed by interviewing four experts from the logistics domain in Alcatel-Lucent.

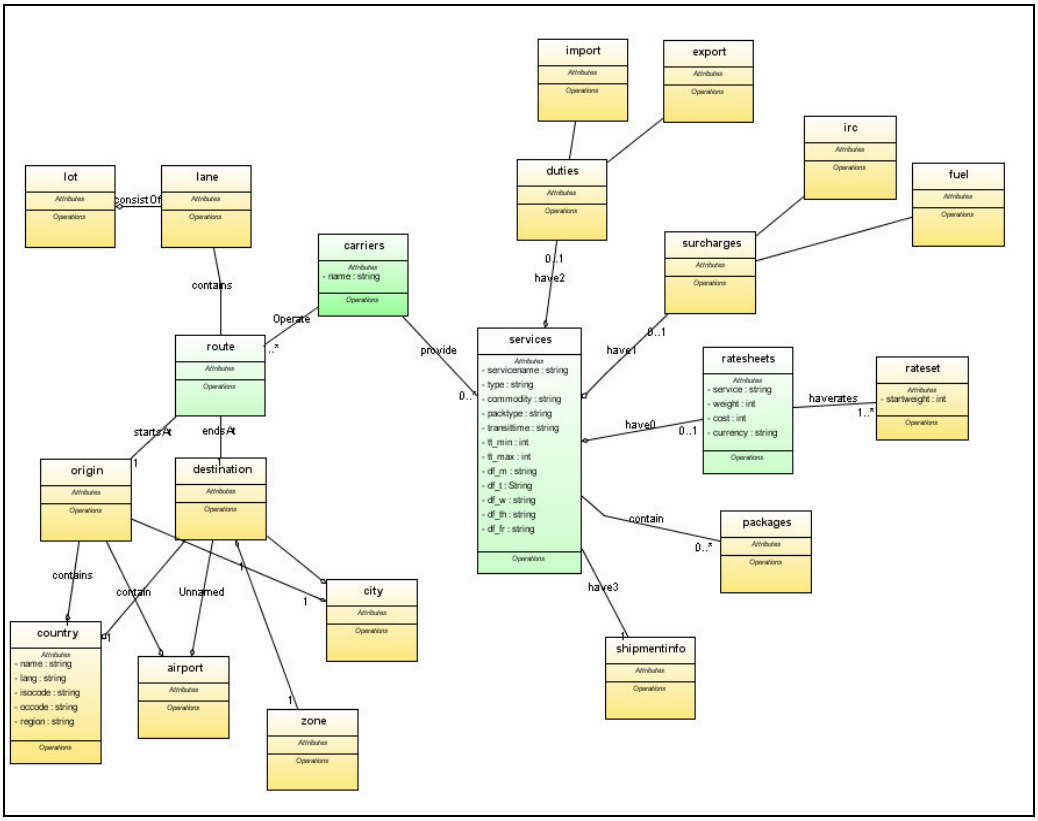

Figure 4-7: Concept overview from Logistics Ontology

The main classes in the logistics ontology are summarised below:

- Carriers: A class to represent the details of the logistics provider.
- Services: A class represent the details of the services provided by the logistics provider
- Ratessheets: A class to represent the rates information provides by the logistics' provider.
- Surcharges: Classes to represent the various surcharges associated with logistics (e.g. fuel surcharge, import and export duties).
- Routes: Classes to represent the origin and destination of the logistics routes.

Creation of Mappings

The mapping format was described in the design chapter (Section 3.3). For this experiment, the relationships between the ontology concepts and database fields were identified during the ontology design process with the help of the domain experts. The mappings were then encoded manually into the format was described in the design chapter (Section 3.3). For this logistics dataset, the integration system needed 92

mappings to perform the integrations across 2 databases containing a total of 19 database tables and 234 database fields.

Lower Ontology
The lower ontologies were created automatically from the databases (described in the next section) using the D2RQ API [D2RQ API] as noted in the design chapter.

4.4.6 Experimental Results

The TomE tool was used to develop an understanding of the complexity of the mappings from the reverse logistics application use case by carrying out an analysis of the inter-relationships between the mappings in the system.

The mapping file contained 92 mappings that were decomposed into 92 upper entities (UE), 92 mappings (MP) and 149 lower and ground entities (LE and GE). This section describes the dependencies that were found, how they were formed and what impact they have on the complexity of the mappings.

4.4.6.1 Definition of Dependency Types and Views

In the design chapter, a dependency chain was defined as the set of dependent elements created by joining simple dependencies together to form a chain. A simple dependency was defined as a dependent relationship between a pair of architectural elements from the model (e.g. UE->MP).

The TomE tool creates three different views of dependency based on how dependency relationships in the model are processed. These types of dependency chain can be viewed individually using the tool.

The first type of dependency chain created represents a view of the full graph of dependencies for all UE in the system. This is the default view loaded when the system starts and can be used to navigate to the other views described below.

The second type of dependency chains created represent views of the dependent elements within a single UE. These are inferred using its “ue2mp” and “mp2ge” dependent relations from the OBDM.

The third type of dependency chains created represent views of how dependencies for a single UE extend across other mappings in the systems. These chains are inferred using

the general "depends" relationships from the OBDM. As shown below, this type of dependency chain can arise for two reasons:

- When some mappings (MP) that refer to a GE used in another mapping
- When some mappings using functions that refer to either UE used in another mapping

4.4.6.2 Analysis of Dependency Types

The TomE tool created a total of 92 dependencies chains by inferring chains of dependencies for each UE in the system using OWL axioms defined by the TomE for each UE in the system as described in the design chapter (Section 3.2.6.3).

An analysis of the dependencies in the system using the different views provided by the TomE tool shows that there different types of dependency exhibited as detailed below:

- **Non-Overlapping Dependency:** This is the simplest dependency type and occurs when the GE specified in the mapping do not overlap with any other mapping.
- **Overlapping Dependency:** This type of dependency occur when mappings share a GE concept. These are called overlapping dependencies in the analysis below. (GE concepts represent entities in the data sources).
- **Function-Based Dependency:** This type of dependency occurs where a function refers to a UE that is part of another dependency. These are called function-based dependencies in the analysis below.

A description of each of these types is given below.

Non-Overlapping Dependency.

These dependencies occur when a chain of dependent elements exist as shown here.

UE->MP->LE->GE

This is the simplest dependency type. It is a chain of dependent elements as shown in Figure 4-8. This type of dependency arises from the mapping shown in XML snippet below.

```
<mapping>
<mapping_number>c1</mapping_number>
<mapping_type>ps</mapping_type>
<source_name>carriers:name</source_name>
<dest_ont>log1:log2 </dest_ont>
<dest_prop_name>Awards:cname </dest_prop_name>
<dest_uri_name>logistics:rates</dest_uri_name>
<function>null:null</function>
</mapping>
```

Code 12: Mapping Example

This mapping simply states that the ontology property (name) in class (carriers) is composed of the set (mapping type) of properties found in lower ontologies as specified by "log1/logistics/awards" and "log2/rates/cname". This is modelled using the following chain of architectural elements and can be "typed" as a non-overlapping dependency. There were 92 dependencies of this type.

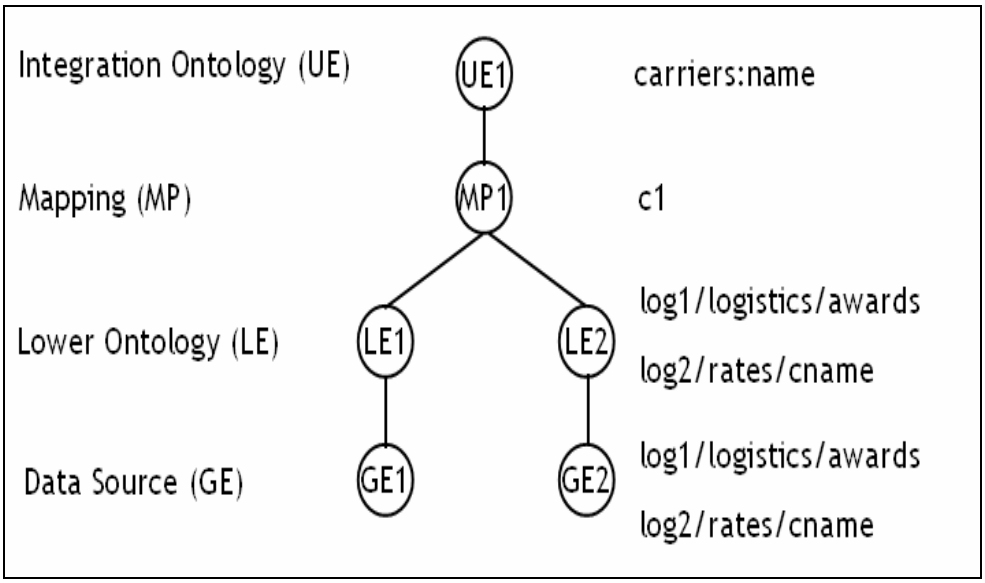

Figure 4-8: Non-overlapping Dependency

Overlapping Dependency

When two mappings share the same lower or ground entity (e.g. GE2 is shared below), the dependency chain that is inferred includes the elements from both mappings. For

example, in Figure 4-9, UE2 will be inferred to be dependent on MP2, MP3, LE1, LE2, LE3, GE1, GE2 and GE3. This effectively means the mapping h1 and mapping h2 are dependent. There were nine dependencies of this type identified in the analysis. This type of dependency occurs when two concepts in the integration ontology use have different abstraction levels for a concept and overlap partially. In this example, the "carriers" concept (UE2) has a lightweight representation of service but the "services" concept (UE3) has a detailed representation.

Figure 4-9: Overlapping Dependency

Function-Based Dependency

Function-based dependencies occur when a function (part of a mapping) references an architectural element (a UE, LE or GE) that is part of another dependency. In the ontology-based dependency model, each mapping (MP) concept has a function associated with it. The function concept supports the definition of input, output and local parameters. In Figure 4-10, function F1 requires access to concept UE2 (ratestructure) in its local parameters. The "ratestructure" concept allows access to information in the database tables that allows transit times (among other things) to be interpreted as either door-to-door or airport-to-airport.

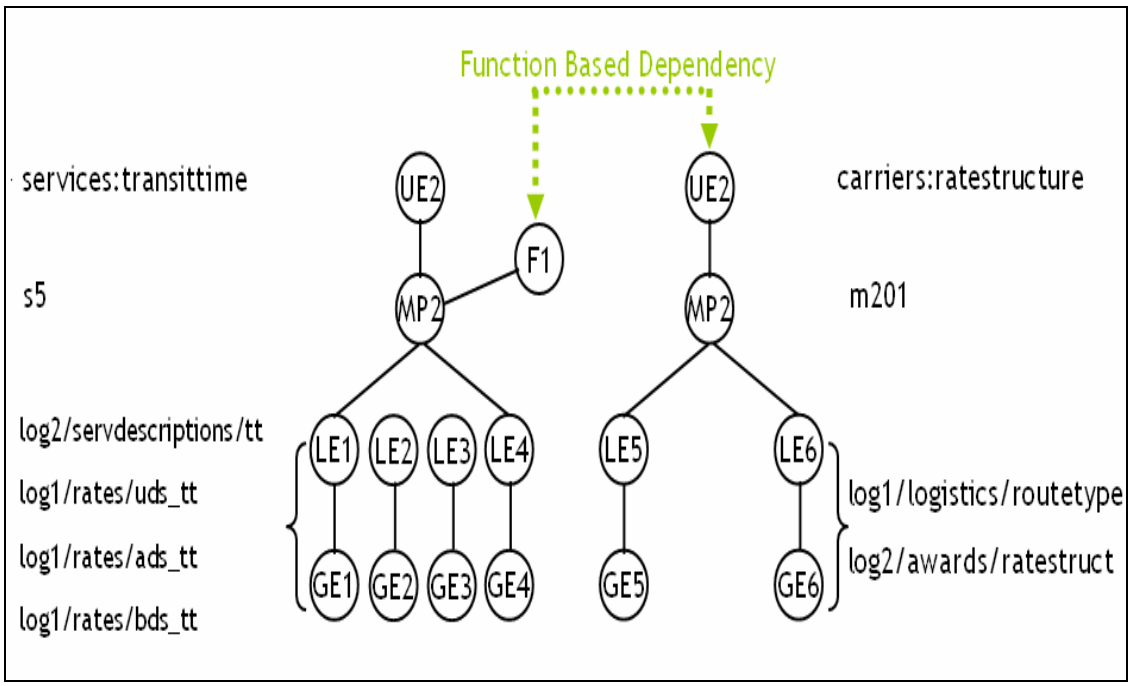

Figure 4-10: Function Based Dependency

20 function-based dependencies were found. The function class of the ontology-based dependency model requires the specification of input, output and local parameters. Currently these inputs need to be prepared manually by analyzing each function implementation (i.e. Java code) and selecting the local, input and local parameter used. In future, the mapping functions could be automatically parsed to create the input, output and local parameters.

Mixing Overlapping and Function-Based Dependencies

Mappings can also exhibit combinations of overlapping and function-based dependencies. This is a composition of the other basic types already presented.

In 11 (of the 20 function based) cases, function based dependencies span multiple dependencies.

Performance Measurements Taken for the TomE tool.

The dependency analysis was run on a low-end machine with 3G RAM, 2 GHz Dual Core Processor running Windows XP Professional.

The performance of the TomE tool was measured by noting the time taken to execute the computation associated with the dependency analysis. The time taken to run the

TomE tool up to the point when the tool is invoked to where the visualisations are ready for the user to load was measured over a sample of five iterations. This time measurement includes the time taken to load the mappings, run the ontological reasoning over the 92 ontological axioms and time taken by the user to navigate from screen to screen in the TomE tool. The average time taken to process the 92 mappings in the mapping file was 201 seconds.

The dependencies that were created by the TomE tool were desk checked for accuracy and the system achieved 100% accuracy.

4.4.7 Discussion of Experimental Results

This experiment developed an ontology-based dependency model (OBDM) and tool (TomE) to support the analysis of mappings arising from the logistics based use case that was applied to the generalised ontology-based integration system.

The ontology-based dependency model and tool provided a very fast method, averaging 201 seconds, to represent the dependencies that occurred in the logistics data set. The tool created 92 dependency chains. Further analysis of the dependency chains, using the TomE tool, shows the existence of 9 overlapping and 20 function-based dependencies.

This represents approximately 30% of the mappings. This significant proportion of the mappings that exhibit these “overlapping” and “function-based” dependencies provides clear evidence of the complex interconnections that the mappings exhibit.

The overlapping and function-based dependencies are particularly difficult to recognize manually. This difficulty is due to the number of mappings that would need to be manually viewed and correlated and the terse nature of data source names. In TomE, search and visualization features allows impacted elements to be quickly identified and provides much faster understanding of the complexity of the mappings in the system. It is hypothesised that these complex mapping relationships are difficult to identify without tool support and thus makes the first step of mapping evolution, finding which mappings are impacted by a change, difficult for integrators. This hypothesis is tested in experiment three.

In the current implementation of TomE, the mapping functions need to be manually reviewed to understand what other entities in the system they access. While not completely automatic in the current implementation of TomE, the automatic generation

of function derived dependencies is important because it is these functions that are likely to be subject to change as mappings are evolved.

The system is extensible in a number of ways. The technique of using an ontology-based dependency model to manage mapping evolution can be adapted to cater for other mapping formats by simply decomposing the mapping format into the core architectural entities. The other mapping formats were not tested explicitly in this thesis. However, in experiment five the ontology-based metamodel was used to build a dependency model for another domain.

4.4.8 Summary of Conclusions, Open Issues and Limitations

Experiment two has shown that approximately 30% of the mappings in the test system exhibit complex dependency relationships with other parts of the integration system. Each mapping exhibits a simple dependency with the ontology and data sources but also may exhibit more complicated dependencies due to concept specialisation and generalisation in the domain ontology and reuse of data source or ontology concepts in the mapping functions.

The classification of types of dependencies (simple, overlapping, function-based) presented in this experiment may not be exhaustive. However, the ontology-based dependency model is not prescriptive about "typing" dependencies. The model will compute all dependencies (irrespective of their type) based on the dependency relationships that have been setup by the dependency model designer. As noted in the future work section, an enhancement to the dependency model could use rules to classify each dependency chain into a type based on the requirements of the dependency model designer.

The data sources and ontologies have been selected from the industrial use case and represent a difficult integration use case. The data exhibits all types of heterogeneity specified in the THALIA tests except the "language expression" and "virtual columns", "nulls" tests as defined in Table 2-1, Section 2.7.1.

Only one mapping format (INRIA [Euzenat 2004]) was tested as part of this experiment. Other mappings formats could cause dependencies between different parts of the integration system not tested in this experiment. However, the approach taken in the design of the dependency metamodel and model creation process means that

irrespective of the mapping format, once the mapping decomposition process is carried out, the dependency model will be able to support other mapping formats.

Following analysis of the results of experiment two, a hypothesis was developed that the mapping relationships are difficult to identify without tool support and thus makes the first step of mapping evolution, finding which mappings are impacted by a change, difficult for integrators.

4.5 Next Steps in Action Methodology

The next iteration of the action based research focused on the performance of the ontology-based dependency model.

The hypothesis that was developed as a consequence of experiment two results analysis, which stated "that complex mapping relationships are difficult to identify without tool support and thus makes the first step of mapping evolution, finding which mappings are impacted by a change, difficult for integrators", now needed to be tested.

To achieve this, the performance and accuracy of a manual approach to dependency analysis and the OBDM were compared using the "Dependency Identification Performance" metric.

4.6 Experiment Three – OBDM Performance

4.6.1 Overview

This experiment aimed to demonstrate the difficulty associated with the identification of the dependencies between mappings within an ontology-based integration system without tool support. To measure the performance of dependency analysis without tool support, a manual process for dependency analysis was defined. The process was created by the author of this thesis following interviews with logistics and data integration specialists. The interviews enabled the definition of the basic steps of the process that is fully described in appendix II.

The manual process for dependency analysis was then provided to a group of 18 integration or logistics specialists. Using the process, this group was asked to carry out a series of dependency analyses using the process. The "Dependency Identification Performance" metric (Section 1.2) was used to measure the performance of the manual approach, i.e. the ability of the system to accurately and quickly identify the mapping dependencies. To measure the performance of the approach, metrics related to time and accuracy were collected during these exercises.

The results of experiment three shows that with the theoretical set of evolution needs as exemplified the mappings in the exercises, the dependency analysis is a very difficult process to carry out without tool support. Furthermore, the results show that the ontology-based dependency model (OBDM) provides fast, accurate and automatic

support of the first step of mapping evolution, i.e. to understand which parts of the system are impacted by the change.

Section 4.4.2 describes in the objectives of the experiment in the context of the research question.

Section 4.4.4 describes in detail the approach taken for this experiment.

Section 4.4.6 and 4.6.5.5 describe the results and conclusions of this experiment.

4.6.2 Objectives & Hypotheses

In the Introduction chapter, the fourth objective that was derived to evaluate the research question was stated as:

- Evaluate the dependency model and tool using a concrete industrial use case.

This objective was addressed by experiments 3 and 4. Experiment three evaluates the performance of the ontology-based dependency model using a theoretical set of mappings. (Experiment four used the ontology-based dependency model to carry out a real set of evolutions based on data from the Alcatel-Lucent logistics supply chain.)

Experiment three confirms the hypothesis derived from the results of experiment two. This states that complex mapping relationships are difficult to identify without tool support and thus makes the first step of mapping evolution, finding which mappings are impacted by a change, difficult for integrators.

The objectives of the third experiment were:

i) To demonstrate the difficulty of the first step of mapping evolution (i.e. identification of the dependencies in the system) by measuring the accuracy and speed of a manual process oriented approach when presented with a set of theoretical data source evolutions.

ii) To confirm the accuracy and runtime performance of the ontology-based dependency model given the same set of theoretical data source evolutions.

4.6.3 Experimental Approach

Objective (i) for this experiment required the development of the manual dependency analysis process, setup of the dependency analysis exercises, setup of statistical

framework to measure accuracy and time metrics. The approach taken for each of these tasks is summarised next.

Manual dependency analysis process

A manual dependency analysis process was developed by interviewing integration and logistics experts to identify the key processes required to find the dependencies within a theoretical set of mappings based on logistics data. The process is described in detail in Section 4.6.4.1.

Dependency analysis exercises

A set of 12 dependency analysis questions was designed by the author of this thesis based on a set of theoretical mappings. The mappings were divided into small, medium and large mappings data files. Each dataset has 4 questions associated with it. The 12 questions were created with predefined complexity, based on the type and depth of the dependency. The type of dependency was simple, overlapping or function-based (Section 4.4.6.1). The depth of the dependency was a measure of depth of the dependency chain that needed to be found in the exercises. The questions were designed to ensure an even distribution in the complexity of the answers across the exercises

A group of 18 users (described in detail in section 4.6.4.5) were given the datasets and questions and were asked to find the predefined dependencies between sample mappings using the manual process.

Statistical framework

Metrics were collected during the exercises to enable statistical analysis of the time taken and accuracy of users as they executed the process. These metrics were used to calculate accuracy and time of each exercise, and thus the dependency identification performance. The statistical analysis was carried out using the R statistical package (see section 3.5.10).

To achieve objective (ii) for this experiment, the TomE tool was used to compute the answers to the same questions used in the exercises. The time taken to run the tests for each data set was measured. The accuracy of the answers was also desk checked.

4.6.4 Experimental Setup

This section describes in detail the approach taken for creating the mapping data, the manual process, for selecting the user groups and the process to run the exercises.

4.6.4.1 Manual Process Definition

The manual process was created by interviewing three integration and logistics experts who have direct knowledge of the data (i.e. work in logistics) or are knowledge engineering experts with more than 5 years experience. The interview process enabled the identification of the important parts of the process to identify dependencies in mappings.

It is difficult to generalise the process created using this approach to other dependency analysis problems because the outputs of the interviews are relevant only to the particular case of a generalised ontology-based integration system with logistics data. This manual approach was taken because as noted in [Bernstein and Melnik 2007], there are very few industrial scale integration systems that use ontologies. Furthermore, the cost of non-ontology-based integration systems put them outside the scope of this work. Dependency analysis approaches used in other domains, as discussed in the state of the art, do not port easily to the data integration domain.

The key process steps defined by the interview process are summarised in Table 4-2.

Step	Description	Record Item
1	Identify the first row in the spreadsheet that matches the data property defined in the question.	Row Name
2	Find other rows that depend on the first row due to the overlap in elements of the GE columns.	Row Names
3	Iterate step 2 for every new row found	Row Names
4	For each row name recorded already, find other rows that depend on them due to the function column	Row Names

Table 4-2: Key Process Steps from interviews

Using these key steps, a detailed process was developed for the exercise and this is described in Appendix II. The detailed process added more navigation detail to help the users understand which columns and data items are referenced. The detailed

process also added some house keeping details regarding the time taken for each question, users name and clear identification of the dataset in use.

4.6.4.2 Theoretical Mapping Data Setup

Three theoretical mapping data sets were prepared. The datasets were derived from Alcatel-Lucent logistics data used in experiment two. The small dataset contained 51 mappings (small data set), the medium contained 71 mappings (medium dataset) and the large contained 102 mappings (large dataset). The different dataset sizes were designed to provide an indication as to how both manual and tool-based approaches scale with respect to time and accuracy as the number of mappings increases.

Within each data set, one question could be resolved by finding a simple dependency, one question could be resolved by finding an overlapping dependency and two questions by finding overlapping and function based dependencies. Each dataset was setup to contain the same level of complexity in terms of the type (simple, overlapping and function based) and depth of the dependencies.

The data properties names in the dataset were derived from the Logistics data. This provided the opportunity to understand if knowledge of the underlying data would improve the performance of dependency analysis. A control group of 3 users from the Alcatel-Lucent logistics team was asked to carry out the exercise. This group had day-to-day exposure to the logistics terminology that was used in naming the data properties in the mapping files.

4.6.4.3 Evaluation Process

Each user was given a tutorial before the survey that covered in detail the steps of the manual process that needed to be carried out. Each respondent was given 20 minutes to work through the four questions associated with each dataset. The following materials were provided:

Mapping files: The mapping file for each data set was provided electronically in MS-Excel format. The MS-Excel mapping file provided a simple view of the actually mappings from the logistics domain. A sample is shown below (Figure 4-11). All XML tags were removed and each mapping was represented on a single row in the spreadsheet. Column A represents the Upper Entity (UE), Column B represents the mapping (MP), Column's C, D, E represent the lower entities (LE) and column F represent the function identifier (F).

	A	B	C	D	E	F
1	UPPER ENTITY	MAPPING	LOWER ENTITY ADDRESS 1	LOWER ENTITY ADDRESS 2	LOWER ENTITY ADDRESS 3	FUNCTION
2	UE:UE1	mp1	ds1:ds2:ds2	field1:field50:field100	table1:table10:table20	
3	UE:UE2	mp2	ds1	field2	table2	UE:UE9

Figure 4-11: Excerpt from Excel mapping file

The mapping file for each data set is provided in Appendix II.

Question & Answer book: A booklet was provided to each user that contained the questions for each data set and answer space to note the dependencies found. A sample answer book is provided in Appendix II.

Dependency Analysis Process Description: The process to be used was the same for each question. The process was demonstrated using an animated PowerPoint presentation (using dummy data) to ensure each user understood the steps and could ask questions about the process. Each user received a hardcopy of process descriptions and slide ware. In summary, the process contained the following steps:

- **Step 1:** Note start time
- **Step 2:** Check data set name
- **Step 3:** Find the first row where the entity provided in the question occurs and note this row down in the answer space
- **Step 4:** Find other occurrences of the columns C, D, E in the rest of the rows of the spreadsheet and note these rows down in the answer space.
- **Step 5:** Check if any matched row found so far, has a function specified. If the row has a function specified, check in other rows for this identifier and note down these rows in the answer space.
- **Step 6:** Note end time.

The full process description is provided in Appendix II.

4.6.4.4 Performance Metrics

Using the measurements collected during the evaluation, the following statistical measures were used to understand the performance of the manual process.

Central tendency: this statistic was used to determine whether there is a central tendency for the automatic approach to outperform the manual approach with respect to time and accuracy.

Dispersion: this statistic was used to determine the dispersion in the measured data with respect to the time and accuracy. This was calculated using the standard deviation and range of the time and accuracy data.

Correlation: this statistic was used to determine:

1) The association between the manual approach, complexity of dataset with respect to time and accuracy.
2) The association between the automation approach complexity of dataset with respect to time and accuracy.
3) The association between accuracy with respect to type user group (integration versus logistics experts)

The following measurements were collected either during the evaluation or computed before the evaluation as noted below.

Time

Time to complete each question. (Collected from user)

Time to complete each dataset. (Collected from user)

Time to complete compete exercise. (Calculated)

Accuracy

Number of Valid Dependencies found (Calculated from user answer)

Number of Invalid Dependencies found (Calculated from user answer)

Number Dependencies not found. (Calculated from user answer)

Answer Complexity (Calculated)

No. Nodes: No of nodes in the dependency graph for each mapping.

No. Levels: Depth of computed dependency graph for each mapping.

4.6.4.5 Selection of Groups

The primary users of the ontology-based integration system will be integration system specialists and supply chain specialists. The population used for this evaluation has been selected to represent these two constituencies. From this population, a sample of representative users was randomly selected. The sample was divided into three groups. The first (main) group comes from engineering, computer science backgrounds who work on the research and development of the ontology-based information systems. They are expert in database, and ontology techniques. The second group comes from the professionals from the Supply Chain organisation within Alcatel-Lucent and are expert on the data content (i.e. logistics). A third control group were provided with a simpler manual process to carry out. The users in this third group had the same background as the first group (i.e. integration specialists).

4.6.4.6 Post Exercise Interviews

Each user was interviewed after the exercise to collect qualitative data on the user's perception of the exercises. This interview was divided into two parts. The user was asked to fill in a questionnaire which was attached to the back of the question and answer booklet (Appendix II).

The first three questions of the questionnaire relate to the users perception of the difficultly of the dependency analysis task (e.g. rate difficult of task, rate hardest question). After filling out the questionnaire the user was asked to comment on their perception of the exercise – using the first three questions as a common reference for comment for each user.

4.6.5 Experimental Results

The evaluation ran over a period of four weeks in November and December 2009.

4.6.5.1 Data Summary

Three groups of respondents participated in the exercise. The first group represents the integration specialists who have expertise either in enterprise or research database or data integration technologies (including ontologies). The second group of respondents is a smaller control group that consisted of logistics professionals from the Alcatel-Lucent supply chain. These respondents have a deep understanding the logistics data

but are not integration specialists. The third and final group consisted of a smaller group of integration specialists that were given a simplified MS-EXCEL format mapping file. By comparing the results of this group to the main group an understanding of the influence of the MS-EXCEL mapping format on the accuracy and timeliness of answers could be gained.

Group	No. of Users	Demographic	Mapping Format
Group1	12	Integration Specialists	Normal
Group2	3	Logistics Specialists	Normal
Group3	3	Integration Specialists	Simple

Table 4-3: Group Overview

Smaller groups sizes were used for Group two and Group three. The size of the logistics group (Group two) was limited by the availability of logistics experts at the local site to carry out the dependency exercise. The size of Group three was limited to three people as the analysis is used only access the effect of a new simpler MS-Excel format for mappings.

The results from each group were collected and collated for each user and was entered into a excel table of data. Figure 4-12 shows a sample of the collated answer data for a single user (labelled u1 in the figure). Each row represents the results that the user (e.g. user 1) gave to single answer (e.g. row 3 is the answer to Question 1 in the exercise). Each row in the spreadsheet contained a Group identifier(GROUP), the computer equipment type used (PC), the data set the question refers to (DSSIZ), the question identifier (QUESTION), the correct answers data (NODES, LEVELS, OVERLAPS, FUNCTIONS, MULTI, SINGLE), user identifiers(User, User Name), the actual answer performance (TIME, ACCURACY).

2	GROUP	PC	DS	SIZ	Questi	NODES	LEVELS	OVERLAP	FUNCTION	MULTI	SINGLE	User	User na	TIME	ACCURACY
3	KDEG	MAC	M		Q1	9	3	1	1	0	0	1	u1	720	100
4	KDEG	MAC	M		Q2	4	3	0	2	0	0	1	u1	255	0
5	KDEG	MAC	M		Q3	4	3	0	1	0	1	1	u1	105	25
6	KDEG	MAC	M		Q4	12	6	0	3	2	0	1	u1	120	0
7	KDEG	MAC	S		Q5	2	2	1	0	0	0	1	u1	75	100
8	KDEG	MAC	S		Q6	7	5	2	2	0	0	1	u1	90	28.5714286
9	KDEG	MAC	S		Q7	7	4	1	1	1	1	1	u1	150	57.1428571
10	KDEG	MAC	S		Q8	12	6	0	3	0	0	1	u1	165	16.6666667
11	KDEG	MAC	L		Q9	26	8	4	2	1	0	1	u1	70	3.84615385
12	KDEG	MAC	L		Q10	26	7	2	3	0	1	1	u1	170	7.69230769
13	KDEG	MAC	L		Q11	4	3	1	1	0	0	1	u1	75	75

Figure 4-12: Collated survey data

The answer accuracy was measured using a simple percentage of the number of correct nodes found. The full data set contained 216 samples. The data set was processed as follows:

Invalid Nodes: Answers that had invalid nodes were removed because the time users spent following chains of invalid nodes would impact on the time and accuracy. This step removed 45 samples from the dataset.

Out of time: Answers where the user noted "out of time" were removed as the question was not completed correctly and was deeded invalid. This step removed 55 samples from the dataset.

Missing Data: Answers where the user forgot to note timings or answer data was not intelligible were removed as the question was not completed correctly and was deeded invalid. This step removed 3 samples from the dataset.

Following these data processing steps, the data contained 120 samples.

It was noted during the evaluation that during the first dataset, some questions were asked about the process. This provides some concern that the process was still bedding in during the first few questions. To cater for this effect, correlation statistics are presented below that have the first two questions from the first data set removed.

Following these data processing steps, the data contained 90 samples.

The impact of removing these samples from the dataset is discussed in the conclusions (section 4.6.7).

The statistical analysis was carried out using the R statistics package. R is a language and environment for statistical computing and graphics.

4.6.5.2 Descriptive Statistics

The averages (mean) for accuracy and time to complete across the entire sample is 61.27% and 265 seconds, respectively. In the context of ensuring the evolution of mappings is correct, the goal for accuracy needs to be as close to 100% as possible to ensure correct functioning of the integration system. An error in mapping evolution could lead in the worst case to erroneous data integration.

The standard deviation of the total data set for both accuracy and time indicates that the spread of samples from the mean is wide. This wide spread reflects the difficulty of the

in attempting to carry out dependency analysis manually. In the post exercise interviews, most respondents cited fatigue due to the repetitive nature of the task as a significant factor that affected the performance.

	Mean	St Dev	Min	Max	Median
Accuracy	61.27	29.2	0	100	57.73
Time	265	171	60	900	234

Figure 4-13: Accuracy & Time Means

Breaking the data down by dataset size, it can be seen that the mean accuracy shows only a moderate swing due to dataset size. This is important because it indicates that in spite of the number of mappings in the dataset to be analysed, the accuracy is broadly constant across the datasets. (The small data set had 51 mappings, the medium had 71 mappings and the large had 102 mappings.)

Accuracy	Mean	St. Deviation
Large	50.01858	24.04875
Medium	64.44444	33.07189
Small	63.31845	27.57551

Figure 4-14: Accuracy Means by Dataset size

4.6.5.3 Correlations

The correlations presented below are for the user population that were given the same mapping format (i.e. Group 1 and Group 2 from Table 4-3). Group 3 was excluded since they were given a simpler mapping format.

Each correlation is broken down by the number of functions, overlaps, nodes and levels in the answer. Each of these measurements represented a different aspect of the complexity of the answer to each question in the exercise.

Accuracy Correlations

The data in Figure 4-15 demonstrate a moderately strong negative correlation between accuracy and various measures of answer complexity. This indicates that as the complexity of the answer increases, the accuracy of the answer reduces. This correlation also holds true across the small dataset and so indicates that in spite of a smaller number of mappings in the mappings file, the accuracy still suffers.

Referring to Figure 4-15, the correlation for the "Levels" metric is stronger than the "Node" metric. The "Levels" metric is a measure of the depth (number of levels) in the dependency. The "Nodes" metric is a simple count of the number of nodes in the answer. The stronger correlation for levels indicates that it is the depth of the answer that impacts accuracy more than the number of nodes in the answer. This is important because it indicates that a small mapping set with complex dependencies can still be difficult to evolve.

The correlation for the "Functions" metric is stronger than the correlation for "Overlaps" metric. The "Functions" metric is a simple count of the number of dependencies that arise due to functions in the answer. The "Overlaps" metric is a simple count of the number of dependencies that arise due to "overlapping" nodes in the answer. This indicates that the dependencies that arise due to function overlap are more difficult for the manual process to detect accurately. As function overlaps were processed as the last step in the manual process, this may impact this accuracy of this measurement due to the combined of effects of the "answer review" and "fatigue" problems pointed out by most users. These effects are discussed in the conclusions section of this experiment.

Accuracy Correlations					
Dataset Size	**ALL**	**Small**	**Large**	**Small+Large**	**ALL**[19]
Nodes	-0.56	-0.71	-0.83	-0.68	-0.66
Levels	-0.66	-0.75	-0.85	-0.80	-0.78
Overlaps	-0.29	-0.08	-0.61	-0.36	-0.36
Function	-0.64	-0.73	-0.81	-0.77	-0.73

Figure 4-15: **Accuracy Correlations**

The statistical significance (p-value) for the all the accuracy correlations was less than 0.001.

Time Correlations

The data in Figure 4-16 demonstrate a moderately strong positive correlation between time taken to answer each questions and various measures of answer complexity. This indicates that as the complexity of the answer increased, the time taken to find the dependencies also increased.

[19] Data associated with question 1 and question 2 removed as noted in data summary.

The correlation figures for each of the different measures of complexity (levels, nodes etc) indicate no definite correlation preference. Therefore, from a time perspective it appears that there is no advantage in having mappings that have fewer levels - as is the case for accuracy correlations. The post exercise interviews provide evidence towards the cause for this behaviour. Most respondents indicated that they needed to "redo" certain steps in the process as they were inclined to lose track of what nodes in the spreadsheet had been checked already. This behaviour would effectively add more time to the answer for each question but would not necessarily improve the accuracy.

Time Correlations					
Dataset Size	**ALL**	**Small**	**Large**	**Small+Large**	**ALL[20]**
Nodes	0.36	0.388	0.66	0.40	0.42
Levels	0.36	0.4	0.68	0.47	0.49
Overlaps	0.29	0.03	0.67	0.31	0.31
Function	0.22	0.38	0.46	0.35	0.38

Figure 4-16: Time Correlations

The statistical significance (p-value) for the all the time correlations was less than 0.001.

4.6.5.4 Impact Logistics Expertise and Simplified Mapping Format

Three groups of users were tested during this experiment. The second group (Group 2 from Table 4-3) was comprised of logistics experts who work within the Alcatel-Lucent supply chain. They work day to day with the logistics data using in the mapping exercise and thus are domain experts. The third group (Group 3 from Table 4-3) consists of integration specialists who were given a simplified mapping file format. This mapping file format simplified the search required to find matches on any given row by joining three columns of data together into one column.

Logistics User Group

Breaking down by experience level, Figure 4-17 shows that the mean accuracy for the entire group, logistics and integration professionals. Logistics professions performed better than the full population as indicated by mean answer accuracy of 77% for the logistics user group.

[20] Data associated with question 1 and question 2 removed as noted in data summary

Accuracy	Mean	St. Deviation
Full Sample	61.3	29.2
Logistics Professionals	77.18254	19.63985
Integration Professionals	56.20316	28.66833

Figure 4-17: Group Analysis (Accuracy)

In the post exercise interviews with the logistics professional, most of this user group indicated that while they recognised most data terms in the spreadsheets, they felt it did not help them complete the task any better. It is also noted that two of the logistics respondents used the advanced excel feature of column colouring, auto filtering and Vlookup[21] feature. This could have been a contributing factor for the improved accuracy.

The analysis for the mean time to complete the answers is less clear (Figure 4-18). The Logistics professionals show a marginally smaller mean time. However, because the standard deviation is large, it is difficult to draw conclusive result in relation to time for this user group. This behaviour may be a function of the "answer review" and "fatigue" problems that most respondents highlighted in their post exercise interviews.

Time	Mean	St. Deviation
Full Sample	265	171
Logistics Professionals	218	111

Figure 4-18: Group Analysis (Time)

Simplified Mapping File User Group

The mean accuracy for this control group was 63%. This indicated that there is very little difference in accuracy between this group and the larger population. The post exercise interview provides a hint to understanding this behaviour because most respondents felt the exercise was very difficult.

	Mean	St. Deviation
Accuracy	63	35
Time	338	215

Figure 4-19: Control Group Accuracy and Time

[21] Vlookup is an advanced lookup feature of Microsoft Excel Spreadsheet

4.6.5.5 Performance of the automatic approach

In the current implementation of the TomE tool, all dependencies are pre-computed as described in the design and implementation chapter. Once the processing stage is complete, the time to search and query using the functionality of the tool is bound only by the speed of the user.

Figure 4-20 shows the processing time for each mapping file used in the evaluation.

No of Mappings	Processing time (Seconds)
51	127s
71	160s
102	205s

Figure 4-20: Automatic Approach Processing Time

The tests were run on a low end machine with 3 GB RAM, 2 GHz Dual Core Processor running Windows XP Professional.

The answers from the automated approach were desk checked for accuracy and the system achieved 100% accuracy.

The runtime performance of the every user interface function was not tested, however the user interface performance of the test platform described above was adequate. Node expansion and collapse was of the order of 1-2 second response time. The initial loading of the full dependency graph took in the order of 3-4 seconds for the largest mapping file with 102 mappings.

4.6.5.6 Collation of Post Exercise Interviews

This section contains a summary of the answers to the user questionnaires which was attached to the question and answer book (Appendix). After the dependency analysis exercise, each user was asked to complete the questionnaire. The answers given to each question are described below.

Q1: "How do you find the task?"

7 users rated the task as Hard, 8 users rated the task as Very Hard and 3 users rated the task as impossible,

Q2: "Which part of the process was the hardest?"

9 users rated Step 4 of the process as the hardest while 9 users rated step 5 as the hardest.

Q3: "Rate the hardest and easiest dataset?"

Dataset 3 (largest) was rated as the most difficult. Dataset 2(Smallest) was rated as the easiest.

Q4: "Rate the hardest and easiest question?"

Question 9 was rated as the most difficult question. Questions 2, 3 and 5 were rated as among the easiest to answer.

Summary of findings from the Interviews.

These four questions were used as the context for a general discussion with each user to gather more detailed information about the issues encountered while carrying out the task. Two main themes emerged from the interviews. The first and biggest issue that users encountered was described as the confusion the user experienced in remembering which step of the process they were executing. This was described by some as "excel overload" or "snow blindness". To resolve this confusion, some parts of the process were repeated or rechecked by the users. This was called the "Answer Review" problem and is discussed in Section 4.6.6. The second issue that was highlighted by half of the respondents was that "fatigue" set in during the exercise. The exercise lasted 60 minutes that was deemed to be "intensive", "busy", "heavy going" by respondents. Some respondents noted that the fatigue was more prevalent in complex questions and became more progressive as the exercise progressed. This was called the "Answer Fatigue" problem and is also discussed in Section 4.6.6.

4.6.6 Discussion of Experimental Results

The statistical analysis above allows some conclusions to be drawn from the data. These are summarised below.

4.6.6.1 Performance of Manual Approach

Accuracy

The strongest negative correlation is between answer complexity and accuracy. In particular, the number of levels in the answer is the dominant correlation. This finding

has an interesting impact on the evolution of mappings in ontology-based integration system. Even integration systems with small number of mappings can still prove a challenging to evolve the mappings.

This finding could be further analysed to develop some design patterns for the creation of mappings and ontological concepts to minimize the number of overlapping mappings. One way that this can be achieved is to limit the number of generalization/specialization concepts in the integration ontology (as they lead to overlapping dependencies).

Time

The correlation picture for time is less clear. While a positive correlation exists in the data between time and the answer complexity measures, the data does not allow for a clear distinction to be made. The positive correlation indicates that more complex answers will take longer to complex. The correlation is weaker than for accuracy. It is clear from the post exercise interview and that fact that no respondent completed all questions that the task is time consuming and performance is likely worsen the longer the task is persisted.

Impact of Logistics Expertise and Simplified Mapping format.

A deep knowledge of the domain data (as was the case for the logistics control group) allows for a small improvement in answer accuracy but does not improve the time to complete. Providing a simple data format for the mappings did not influence the accuracy of the answers (63% for the control group, 61% for the full group).

The "answer review" problem.

During the post exercise interview, many (12 of 18) users described the biggest issue that they encountered as the confusion, described by some as "excel overload" or "snow blindness", as to which step of the process they were currently working on. To resolve this confusion, some parts of the process were repeated or rechecked by the users.

The "answer fatigue" problem.

Half of the respondents noted that “fatigue” set in during the exercise. The exercise lasted 60 minutes that was deemed to be “intensive”, “busy”, “heavy going” by respondents. Some respondents noted that the fatigue was more prevalent in complex questions and became more progressive as the exercise progressed.

4.6.6.2 Performance of Automatic Approach

An exhaustive runtime performance test was run performed for the TomE tool. However the discussion in this section provided an indication of the overall runtime performance of the system and the main processing functions.

The answers from the automated system achieved 100% accuracy and completed the exercises in 127, 160 and 205 seconds respectively.

As described in the design chapter (Figure 3-7), the TomE tool has four functional areas (Mapping Factory, Model Factory, Dependency Factory, Visualisation). The majority of the processing time is spent in the Model and Dependency Factories. The model factory is responsible for the creation of the in memory ontology model (using the Jena API [Jena]) and validating of the instances of the model using the Pellet reasoner. Model validation is carried out twice to enable easier debug of the model should an error occur – once after the dependency model instances are added and once after dependency axioms are added. Finally, for each dependency axiom, the reasoner is invoked to compute the dependencies associated with the axiom. These functions account for approximately 70% of the processing time.

The dependency factory is responsible for creating the dependency graphs. In the current TomE implementation both in memory and GraphML file dependency graphs are maintained. These functions account for approximately 30% of the processing time.

These performance results cleared demonstrate the performance and accuracy advantages of the automatic approach.

4.6.7 Summary of Conclusions, Open Issues & Limitations

As expected the results of this experiment show that the ontology-based dependency model significantly outperforms the manual process for both accuracy and time measurements. More significant from the experiment however is the clear indication of the complexity involved in a manual processes and the difficulty in identification of dependencies in ontology-based integration systems without tool support. With the

theoretical set of evolution needs, the ontology-based dependency model provides fast, accurate and automatic support of the first step of mapping evolution.

This experiment used a group of ontology and information systems specialists to represent the system integrators who would be the final users of a dependency analysis system (within an integration system). However, the technical background of the group that was selected would be very similar to database and system integrators. They might also be reasonably expected to work in that field and thus represent an excellent proxy for the system integrators.

While a large sample of data was collected in the experiment (i.e. 216 samples), the data collected was noisy. In particular, 55 samples were removed because the user ran out of time while answering the question. Note that all users were given explicit direction at the beginning of each session to move the next question if more than 10 minutes was spent on any given question. 45 samples were removed as they users answer contained both valid and invalid nodes. Both these effects are representative of the complex and time consuming nature of the exercise.

By removing the invalid, out of time and missing data the remaining samples represented the absolute best case performance of the manual approach and as such provide a very conservative basis with which to compare to the automatic approach. An alternative approach would be to use a precision and recall calculation rather than simple accuracy; however this is likely to lower the accuracy levels of the exercise.

As noted in the state of the art review, most current data integration frameworks tend not to provide mapping management functionality. Therefore the OBDM was compared with a manual dependency analysis approach that was designed as part of this experiment. To mitigate any risk that the manual approach is not representative, the data used in the experiment is based on real industrial data and the manual process was designed using the expertise of the integration and logistics specialists. Furthermore two different mapping formats were tested to ensure that the format of the mapping file did not impact the results.

4.7 Next Steps in Action Methodology

At this point in the action based research process, the dependencies that mappings exhibit has been identified as the focus point for this research, an ontology-based test system and ontology-based dependency model has been designed. The performance of the OBDM has been verified using a comprehensive but synthetic set of mappings.

The next iteration focused on testing the ontology-based dependency model as a new data source was introduced into an existing dataset and examined how the resultant set of evolution needs were coped with.

4.8 Experiment Four – OBDM Performance

4.8.1 Overview

Experiment three demonstrated the performance of the dependency modelling approach using a synthetic set of mappings. This experiment was designed to demonstrate the capability of the ontology-based dependency model when presented with a set of non-synthetic evolution needs.

A new data source that represented a new logistics service provider was added to the ontology-based integration system used in experiment two. The ontology-based dependency model and TomE tool was used to support the identification of which mappings were impacted by the introduction of the new data source.

The experiment shows that the ontology-based dependency model and TomE tool enables the integration/ontology designer to quickly locate the impacted areas and allow analysis of the changes to process in an ordered fashion. The approach supports the mapping evolution process by providing global dependency views of the mappings that allow the user to focus in on areas of high dependence initially and then to drill down progressively to the detail to understand what impact of each computed dependency. As noted in [Halevy et al. 2005, Zhou et al. 2006], this is one of the key challenges facing enterprise integration systems.

Section 4.2.2 describes in the objectives of the experiment in the context of the research question.

Section 4.2.3 provides the background to the supply chain based use case that was used for this experiment.

Section 4.2.5 describes in the detail the approach taken for this experiment.

Section 4.2.6 and 4.2.7 describe the results and conclusions of this experiment.

4.8.2 Objectives & Hypotheses

In the Introduction chapter, the fourth objective that was derived to evaluate the research question was stated as:

- "Evaluate the dependency model and tool using a concrete industrial use case."

To address this objective, experiment four derived the following sub-objective:

> i) To demonstrate the capability and relevance of the ontology-based dependency model when presented with a set of non-synthetic evolution needs.

Hypothesis

Evolution of the mappings in an ontology-based integration system is difficult to identify without tool support due to the difficultly in finding which mappings are impacted when the data sources are updated.

4.8.3 Use Case Background

For this experiment, a new logistics carrier was introduced that provides transportation services by sea and thus providing a new dataset for the ontology-based integration system. The generalised ontology-based integration system was populated with the logistics domain ontology, mappings and data sources from experiment two (Section 4.4.4).

The existing data sources came from logistics carriers that provide air transportation services. For some forward logistics business where fast delivery time is not required, sea transportation can provide very much reduced costs.

The main areas of difference in the data between air and sea logistics services originate from the descriptions of services, surcharges associated with the services and the package types.

4.8.4 Experimental Approach

The ontology-based dependency model and TomE tool were used to carry out an evolution of the mappings of the generalised ontology-based integration system used in

experiment two. The TomE supported the evolution (in step 2 below) by identifying which mappings were impacted for each ontological concept in the integration system.

This analysis provided by the TomE tool developed an understanding of which mappings needed to be changed and what new mappings were required.

This following approach was used to apply the TomE tool to this task:

- **Step 1: Load the new logistics data set.**

  The new logistics data set was loaded and the lower ontologies for this data source was generated as described in the design chapter (Section 3.3).

- **Step 2: Run Dependency Analysis on existing mappings.**

  The TomE tool was run using the current mapping file to identify which data items and mappings are used for each concept in the logistics domain ontology. This step provided detailed graphs of the dependencies for each ontological concept including the mappings and data sources elements.

- **Step 3: Identify candidate mapping updates from the new data sources.**

  Using the output from step (2), for each concept in the logistics ontology (e.g. service name), identify similar data items in the new data sources. The TomE tool provides the view of the current data sources that are mapped to this concept and this can be used to find similar items in the new data sources.

- **Step 4: Identify Missing Ontological Concepts.**

  Identify any data source items that are not modelled by the logistics domain ontology that would require new mappings to be created. This enabled the identification of new mappings that need to be added. This step enables the identification of new mappings that need to be added.

4.8.5 Experimental Setup

This experiment was conducted by the author of the thesis and required the setup of the new logistics data sources and execution of the TomE tool.

Database setup

The new logistics database represented sea transportation rates. The logistics company for the sea rates also provided air transportation rates for experiment two and thus the

database schema was very similar to the air transportation data source. The key differences in the database schema occurred in the service descriptions, rates and surcharges tables. The key differences are described below:

- **Service descriptions:** The descriptions of services for sea transportation added new concepts related to containers types that needed to be incorporated into the existing logistics ontology.
- **Rates:** The descriptions of rates for sea transportation added new concepts related to the fact that rates are based on a per container basis that needed to be incorporated into the existing logistics ontology.
- **Surcharges:** The descriptions of surcharges for sea transportation added new concepts related to the fact that rates are based on a per container basis that needed to be added to the surcharges descriptions currently handled by logistics ontology.

TomE Dependency Analysis Execution

The mapping file for experiment two was loaded into the TomE tool and the dependency graph generation was carried out using the steps as described in the design chapter (Section 3.2.6).

4.8.6 Experimental Results

For this experiment, a new logistics carrier was introduced that provides transportation services by sea. This requires the analysis and update of the semantic mappings used in experiment two.

Of the 92 mappings in the original integration system, it was found that 23 mappings needed to be updated and 17 new mappings needed to be added (for the surcharges concepts related to sea transportation).

4.8.6.1 Analysis of new mappings

The new logistics data represents costs associated with sea transportation. The existing data sources all represent air transportation. The new data source required some updates to the logistics ontology to incorporate new concepts related to the surcharges associated with sea (for example port charges) that are not present in air transportation.

These new mappings were simple and exhibited no overlapping or function-based dependencies.

4.8.6.2 Analysis of updated mappings

23 mappings were updated. These mappings cover the core concepts in the logistics ontology that represent the logistics carrier information, rate information, basic service information.

Using the TomE tool, it was found that 13 (of 23) mappings exhibited simple dependencies that simply required the update of the lower entity part. 10 mappings exhibited complex dependencies that require further analysis to ensure updates are applied correctly. The following sections describe a sample of the simple and complex dependencies that were found by the dependency management tool.

Simple Dependencies

The example in Figure 4-21 shows the dependency graph for the "carriers-name" concept from the integration ontology. This concept has a mapping that needs to collect the logistics carrier name from the databases and is dependent from two database elements (GE). The update required to this mapping can be achieved by adding the new GE reference to the existing GE references in the mapping. This is a relatively simple and self-contained update because the mapping does not have a function associated with it and does not overlap with other mappings. Therefore, the impact of the change is localised to this mapping.

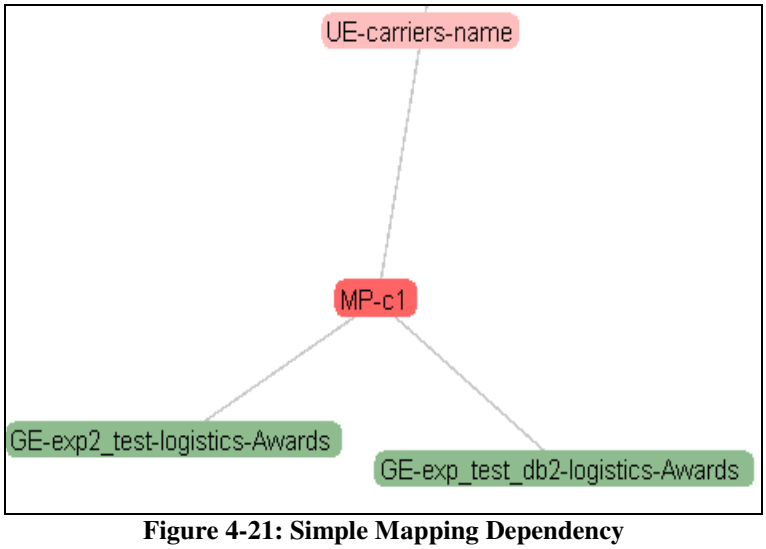

Figure 4-21: Simple Mapping Dependency

Complex Dependencies

10 mappings exhibited complex dependencies because they either overlapped with other dependencies or have functions associated with them or overlapped and had function associated. A representative selection of these mappings is discussed below.

The example in Figure 4-22 shows the dependency graph for the "services-commodity" concept from the integration ontology. This concept has a mapping (MP-s4) that collects the commodity description (name) from the databases and has dependent relations with three other mappings.

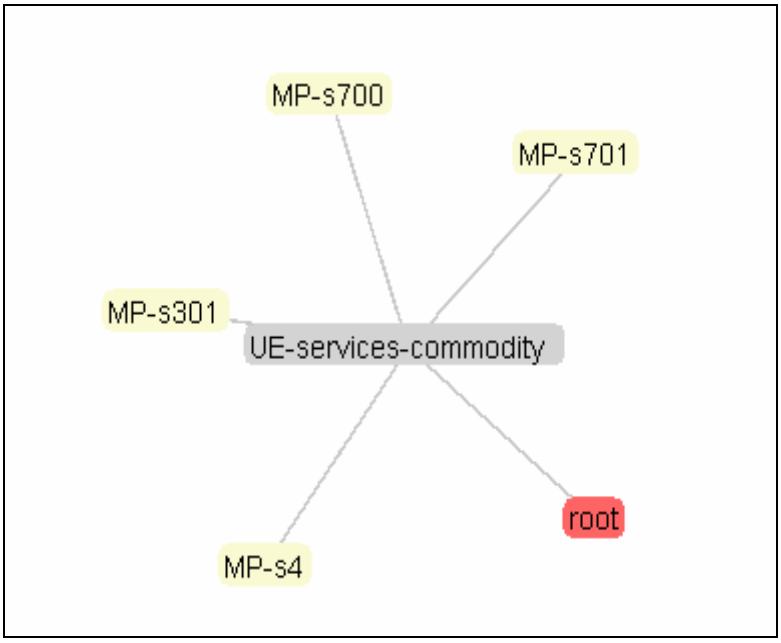

Figure 4-22: Services Dependency

Figure 4-22 isolates the concept under investigation and show the hierarchy of mappings impacted. On first view, there appears to be a complex set of dependencies coming appearing. The Dependency Management tool provides the level view of the dependencies to support the user in the mapping update decision making process.

The levels view, in Figure 4-23, shows the dependency levels and types. In this view the edge highlighted in green is the direct dependency and is assigned level 1. The dependency type and level are identified using the following syntax:

- Overlapping dependencies are identified by "o".
- Function-based dependencies are identified by "f".

- Level is identified by the number that preceded the dependency type (e.g. 2:o).

The graph shows that at level 2, there is an overlapping dependency with MP-s301 and a function based dependency with MP-s700. Finally, MP-s700 has a function-based dependency on MP-s701.

Armed with this additional information, the user can check the other mappings to see if updates are needed to these also. In this case only updates to MP-s4 are required as MP-S301 is a mapping that is used as part of the specification of packages concept that requires access to commodity data also. MP-s700 and MP-s701 are mappings that are used by the commodity concept. The commodity concept is a standalone concept used to describe different types of commodity and there unique reference number that describes various commodity types (e.g. dutiable or non-dutiable) and rules associated with the types.

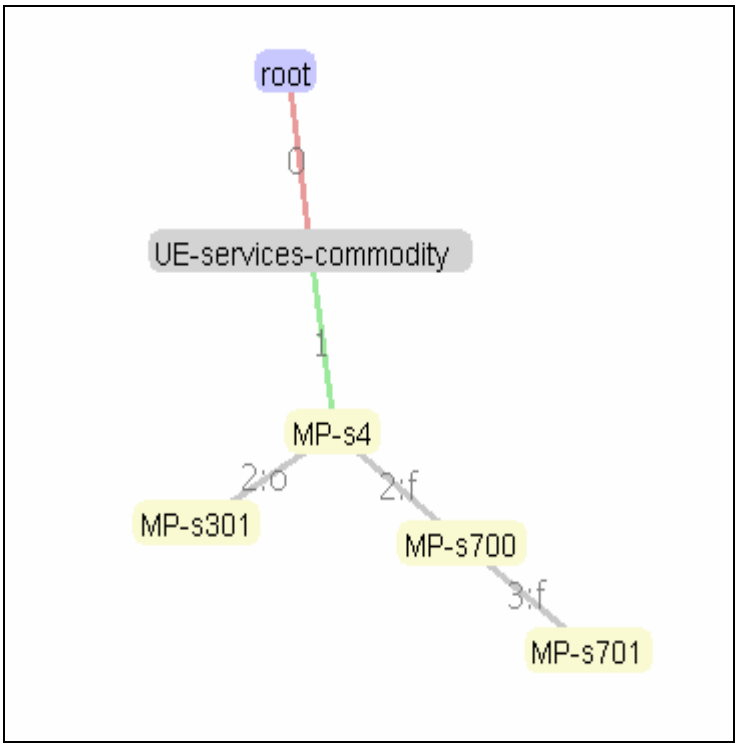

Figure 4-23: Level and Types view

Very complex dependencies

One mapping exhibited very complex set of dependencies. Figure 4-24 below indicates that 32 other mappings exhibit some dependency relationship with "UE-services-servicename". This indicates that the change to this concept could have far reaching impacts. The discussion below shows how these dependencies developed and thus gives the ontology designer useful insight in the underlying database design.

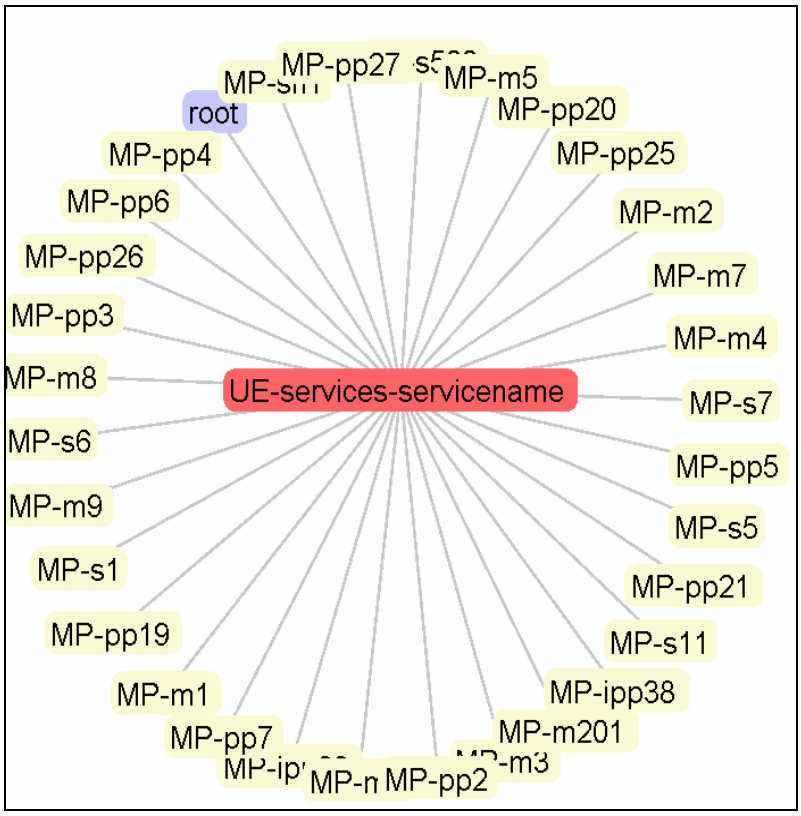

Figure 4-24: Very Complex Dependency

The integration ontology concept "services-servicename" provides a simple abstraction of the service names that are used by different logistics carriers (e.g. Express, Express Saver, and Expedited). Unfortunately one of the underlying logistics databases uses the poor database design practice of encoding both service name and weight category in the schema (i.e. in a column names of tables. The ontological mapping in this case needs to access each column name to extract the service name. The actual column (instance data) for these columns contains the rates information but the service and weight category is encoded in the column name. This creates overlapping dependencies with other ontological concepts for weight, cost and service.

The dependency levels are shown in Figure 4-25.

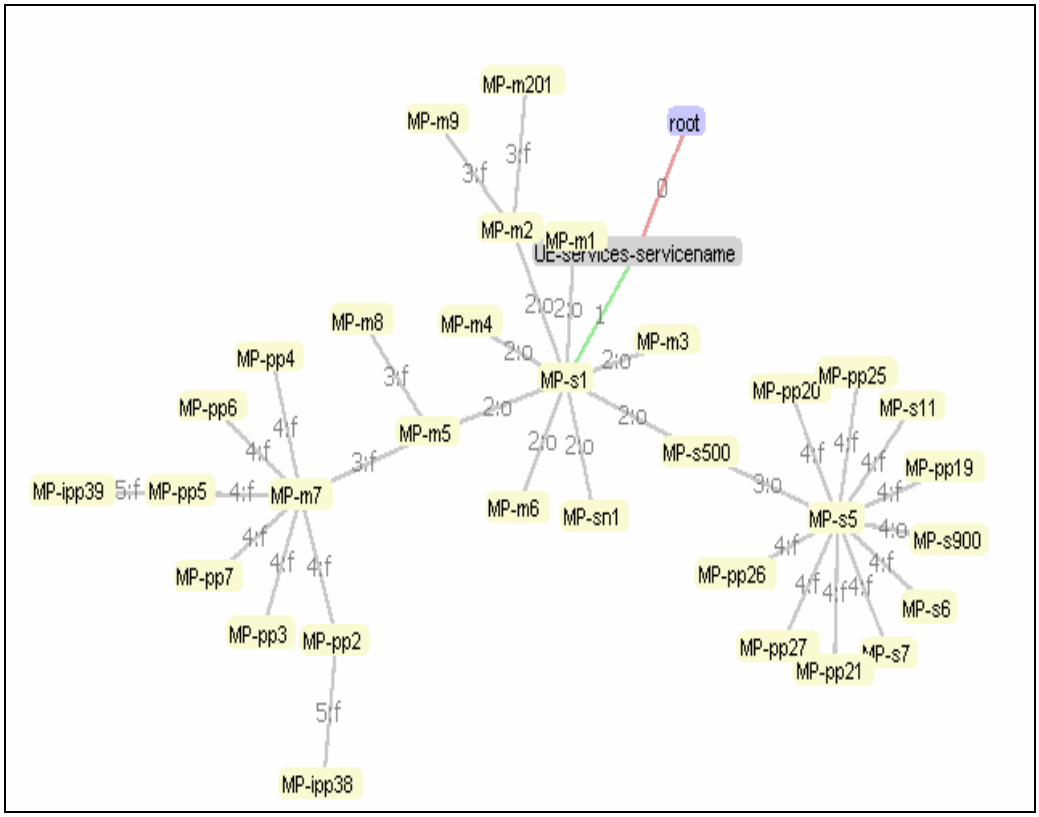

Figure 4-25: Levels and Types Dependency

In the case of the update to “service-name” concept, only the first level dependencies are impacted by the change and the deeper level dependencies do not require update. The mapping associated with “service-name” (MP-s1) does not have a function associated with it and therefore there is no function based dependency identified at level 2 in the graph.

This example highlights a potential issue with the aggression of the dependency algorithm when, as currently implemented, it is tasked with computing all dependent elements. In the conclusions section, a user driven throttling mechanism for the algorithm is discussed.

4.8.7 Discussion of Experimental Results

The ontology-based dependency model and TomE tool provided important advantages to the evolution process carried out in this experiment.

The flexible visualisation capability of the computed dependencies allowed three views of the systems dependencies. This provided quick and accurate computation and visualization of the full impact of the dependencies in the integration system.

This output from the TomE tool supported the evolution of mappings by:

- Identifying which mappings are impacted by changes (Step 2, Section 4.8.4).
- Helping to identify which parts of the new data sources to look at to update the mappings (Step 3, Section 4.8.4).

The TomE tool enabled the integration/ontology designer to quickly localise the impacted areas and allow analysis of the changes to proceed in an ordered fashion. The approach supports the mapping evolution process by providing global dependency views that allow the user to focus in on areas of high dependence initially and then to progressively drill down to the detail to understand what impact of each computed dependency.

This case study has shown that for the update case, the direct (first level) dependencies are the most critical to understand and evolve. This is a feature of this particular use case that focused on the addition of new data source that required mainly updates of existing mappings. However, it can be expected that full range of CRUD[22] operations will come into play when other data sources changes are made. In particular the deletion of some fields from the data sources (perhaps to enable a cleanup or evolution to a more complex schema) will require close study of the indirect (deeper level) dependencies because a delete operation will remove the GE which other indirect mappings depend on and thus break the integration.

Finally, the dependency modelling approach provides could be used to support of verification and testing of the updated system as noted in the state of the art review for dependency (Section 2.3.1). This can be achieved using the full dependency graph for

[22] Create, Request, Update and Delete

any given change because it provides a set of candidate areas to verify or regression test.

4.8.8 Summary of Conclusions, Open Issues and Limitations

This experiment shows that the ontology-based dependency model and TomE tool enables the integration/ontology designer to quickly localise the impacted areas and allow analysis of the changes to process in an ordered fashion. The approach supports the mapping evolution process by providing global dependency views that allow the user to focus in on areas of high dependence initially and then to progressively drill down to the detail to understand the impact of each computed dependency. As noted in [Halevy et al. 2005, Zhou et al. 2006], this is one of the key challenges facing enterprise integration systems.

The data used in the experiment came from the logistics based use case from experiment two. While this data set may not be representative of every mapping evolution task because this experiment focused on updating mappings (and not creating new mappings or deleting existing mappings), the process that would be used to carry out dependency analysis in the delete and new mappings cases is the same. This means once the dependency model can accurately find all the dependencies then the delete and new mappings cases can be accommodated by adding or deleting mappings and rerunning the dependency analysis. A detailed process for the usage of the TomE tool for these cases has not been defined but has been included in the future work (Section 5.3).

The aggression of the dependency algorithm could be throttled by changing the dependency relations that are used to compute dependency graphs. For example, the algorithm could be limited to look only for overlapping type dependencies or to compute to a certain depth. Note that this adjustment capability is not available in the TomE tool and would require changes to the dependency factory code of the TomE tool. This update to TomE has been added to the future work (Section 5.3).

4.9 Next Steps in Action Methodology

The final iteration of the action based research carried out a corroborative study into the genericity of the dependency metamodel that was used to build the ontology-based dependency model (OBDM).

4.10 Corroborative Study – Genericity of the Dependency Metamodel

4.10.1 Overview

This corroborative study applied the dependency metamodel from the design chapter (Section 3.2.3) in a new domain. This provided an extra indication of the genericity of the ontology-based metamodel

The dependency metamodel has already been applied to datasets from both a Product Line Management (Experiment one) and Logistics domains (Experiment two). In these domains, the dependency metamodel was used to support the management of dependencies between mappings in an integration system. In this study, the dependency metamodel is not used to support mappings – rather it is used to support the dependencies that might arise in a domestic electrical circuit.

In this study, a domestic electrical circuit was selected as the application domain because it provided a different set of dependencies from the ontology-based integration system where the metamodel was previously applied. In this domain, the dependency model was used to localise faults in an electrical circuit.

A domain expert on electrical engineering was coached through an eight-step process to build a dependency model, using the metamodel, of an electrical circuit and to carry out a dependency analysis exercise using the model. The eight-step process used Protégé [Protégé] and Pellet [Pellet] to support the dependency model development and the dependency analysis exercise. The dependency analysis exercise was carried out using the model based on the requirements of the electrical engineer.

After the eight-step process was completed, the engineer was interviewed to document the issues that were encountered during the experiment.

4.10.2 Objectives & Hypotheses

This research has developed an approach for the management and evolution of mappings in an ontology-based integration system. The approach taken to achieve this

developed an ontology-based dependency metamodel that defined the basic building blocks of dependencies that can be applied in any domain.

The dependency metamodel has already been applied to the management and evolution of mappings in ontology-based integration in Experiments two and four (Section 4.4 and Section 4.8). This experiment tested the application of the dependency metamodel in a new domain. The metamodel is used in here to describe the dependencies between electrical components in a scoped electrical circuit. The usage of the metamodel in this new domain provides an indication of the genericity of the dependency metamodel.

The aims of the fifth experiment were:

i) To apply ontology-based dependency metamodel developed as part of this research, in another domain to study the ability of the metamodel to be used in other domains.
ii) To discover the issues when applying the metamodel in a second domain.
iii) To provide an indication of the genericity of the metamodel.

4.10.3 Experimental Approach

An eight-step process was defined to support the electrical engineer on the steps required to create a dependency model for an electrical circuit using the ontology-based dependency metamodel. The Protégé Ontology development tool [Protégé] was used instead of using the TomE tool as the TomE tool would have required updating of the mapping factory code. As only a small number of instances would be loaded into the model and visualisations of the dependencies were not required, the Protégé tool was used.

Protégé was used to import the metamodel, to build the ontology-based dependency model for this domain and to run the dependency analysis using the Pellet reasoner.

The electrical engineer was supported on the usage of Protégé by the thesis author. This involved the thesis author carrying out one example of each step in the process and then allowing the electrical engineer to complete the step.

Any errors made while inputting data into Protégé by the electrical engineer were corrected while the data was being input. For example, if an invalid instance was entered (Section 4.10.4 step 7) then this was corrected before moving to the step 8.

After the exercise, an interview was conducted to understand the key issues in carrying out the steps in the process.

4.10.4 Experimental Setup

The following eight-step process was used to setup the experiment. The process was executed over the course of three meetings that were held with the electrical engineer as described below.

Step 1 – Development of electrical circuit domain

The electrical engineer was asked to draw an electrical circuit that represents the main circuits used in a domestic setting based on his expert understanding of the domain. To support the electrical engineer in this task, a meeting (30 minutes duration) was held to present an overview of dependency analysis. The general approach to dependency analysis was described to the electrical engineer using simple examples based on family relations (i.e. Son depends on Father) and automotive engine (i.e. Engine depends on Fuel Supply and Ignition System).

The electrical engineer was asked to draw a domestic electrical circuit that would cover the basic elements of each type of circuit in the home. The electrical engineer was asked to focus on the different types of circuit rather than the different appliances.

The electrical engineer without input from the thesis author created the domestic circuit diagram over a two-day period.

Step 2 – Creation of main circuit components

Using the output of step 1, the electrical engineer was asked to select the major component of the circuit he wished to model from the diagram. These components form the architectural entities of the dependency model for this domain. To support the electrical engineer in this task, a second meeting (90 minutes duration) was held to define which elements of the circuit diagram were to be modelled. These elements were selected based on the requirements of the electrical engineer, who wished to carry out a dependency analysis of each circuit to compute which components were in each circuit and which components depended on each switchboard fuse.

Step 3 – Dependency relation creation
With the output of step 2, the electrical engineer was asked to define the major dependencies between the components. This step was completed during the second meeting.

Step 4 – Generate graph
The output of steps 2 and 3 were used to create a scoped electrical diagram. This step creates the basic dependency model (on paper) for this domain. This step was completed during the second meeting.

Step 5 – Define dependency attributes
The electrical engineer was asked to specify the attributes (transitivity, symmetry etc.) of each dependency relation. The engineer was coached on the meaning of each dependency attribute. This step was completed during the second meeting.

Step 6 – Dependency Model input to Protégé
The author demonstrated the addition of one architectural entity (component) and one dependency relation using Protégé. The electrical engineer was asked to enter the rest of the architectural entities. This step was completed during the second meeting.

Step 7 – Instance input to Protégé
For each circuit type identified by the electrical engineer in step 1, instance data was entered into Protégé. The author demonstrated the addition of one instance of an architectural entity (component) and one dependency relation using Protégé. The electrical engineer was asked to enter the rest of the instance data. This step was completed during the second meeting.

Step 8 – Dependency Analysis
A dependency analysis was run for each circuit using the Protégé and Pellet tools. The author of the thesis ran this step in conjunction with the electrical engineer during the third and final meeting (duration 45 minutes).

4.10.5 Experimental Results

The outputs of each step in the eight-step process are described below.

4.10.5.1 Process Outputs

Output of Step 1

The first step of the process created a scoped domestic electrical circuit with the following components: "Main Switch Board", "Switch", "Light" and "Consumer Device". The electrical circuit created by the electrical engineer for step 1 is shown in Figure 4-26 below. Figure 4-26 contains three circuits (ring, lighting and single appliance) that represent the major circuits in a domestic environment.

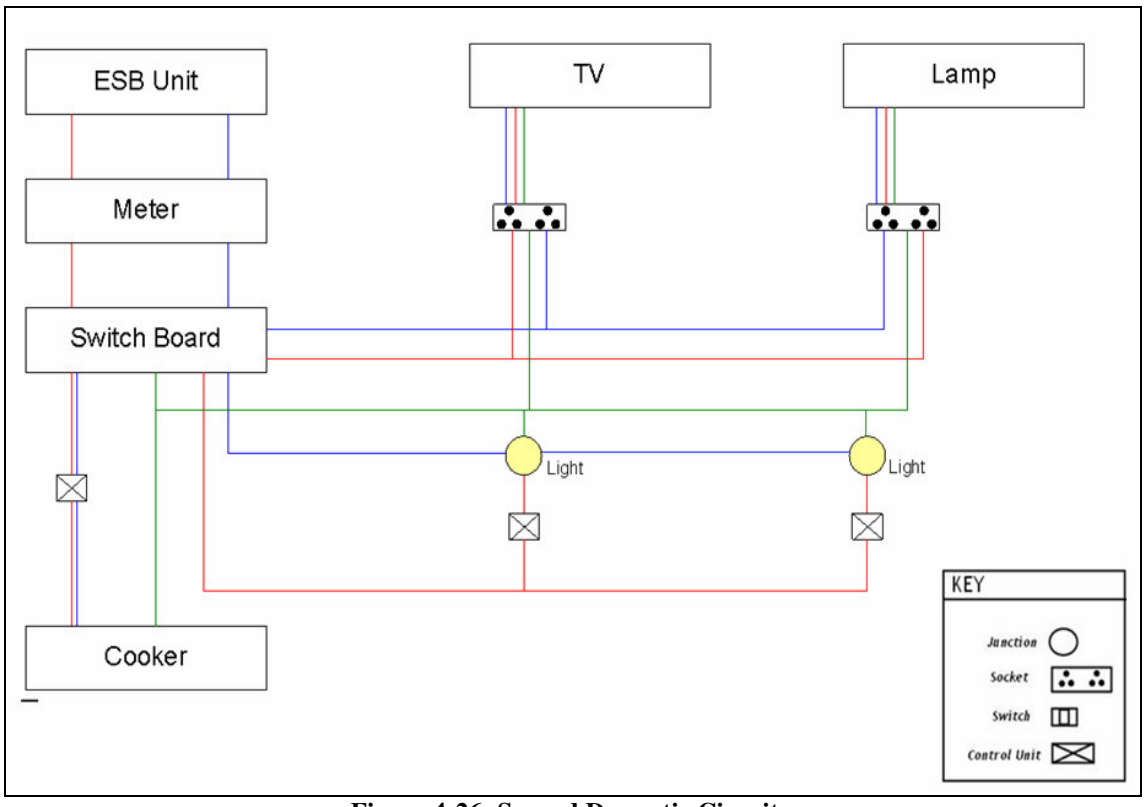

Figure 4-26: Scoped Domestic Circuit

Output of Steps 2, 3, 4, 5, 6

The dependency model created using the process had the following architectural entities:

- **Appliance:** This architectural entity was created as a container concept for the different types of electrical appliances in the home. It has "Light", "Cooker" and "TV" subclasses in this experiment. In Figure 4-26, "Light" and "Lamp" were deemed to be the same by the electrical engineer.

- **Socket:** This architectural entity was created to represent the electrical sockets that are part of the standard domestic ring circuit.
- **SWFUSE:** This architectural entity was created to represent the main fuse board in the home. This represents a simple abstraction of Supplier Unit, Meter and SwitchBoard entities in Figure 4-26. Each "SWFUSE" entity serves a single circuit.
- **Switch:** This represents a switch of any kind on a circuit (e.g. a light switch).
- **Junction:** This represents electrical junctions that are typically used in lighting circuits.
- **ControlUnit:** This represents a control unit that are typically connected to domestic appliances that draw heavy electrical load (e.g. cooker).

Figure 4-27 below shows the concepts that were created in Protégé

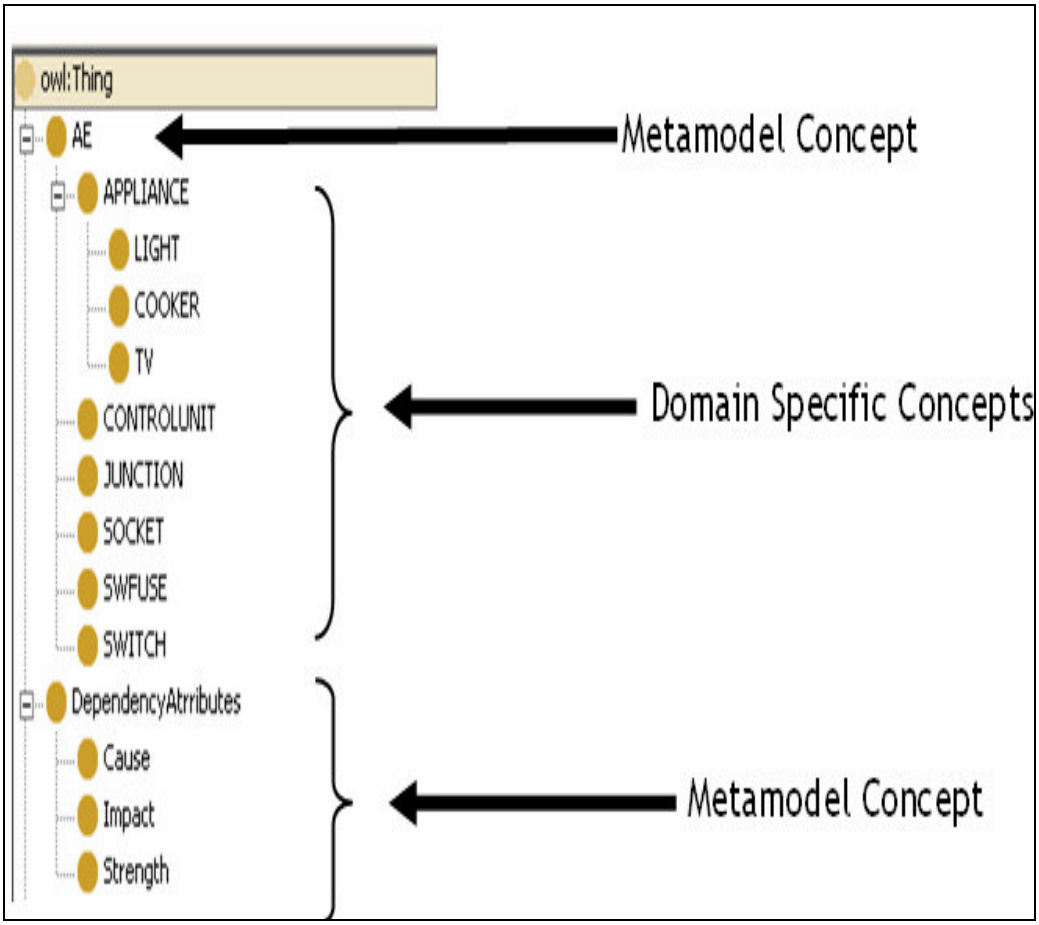

Figure 4-27: Excerpt from the Domain Specific Model (from Protege).

The dependency model created using the process had the following dependency relations:

- **Light2switch:** This dependency relation was created between the "Light" and the "Switch" architectural entities. This relation had the transitive and symmetric attribute set.

- **Switch2junction:** This dependency relation was created between the "Switch" and the "Junction" architectural entities. This relation had the transitive and symmetric attribute set.

- **Junction2swfuse:** This dependency relation was created between the "Junction" and the SwitchBoardFuse" architectural entities. This relation had the transitive and symmetric attribute set.

- **Junc2junc:** This dependency relation was created between the "Junction" architectural entities. This relation had the symmetric attribute set.

- **App2socket:** This dependency relation was created between the "Appliance" and "Socket" architectural entities. This relation had the transitive and symmetric attribute set.

- **Cu2swfuse:** This dependency relation was created between the "Control Unit" and "SWFUSE" architectural entities. This relation had the transitive and symmetric attribute set.

- **Socket2swfuse:** This dependency relation was created between the "Socket" and "SWFUSE" architectural entities. This relation had the transitive and symmetric attribute set.

- **App2cu:** This dependency relation was created between the "Appliance" and "Control Unit" architectural entities. This relation had the transitive and symmetric attribute set.

The dependency attributes (Cause, Impact and Strength) were not applied in this model as the electrical engineer felt that they were not required for the analysis of this scoped domestic circuits because the dependency analysis exercise was to focus on the elements in each circuit and not the attributes of the dependencies between them.

Domain specific models were created to represent each circuit as shown Figure 4-28.

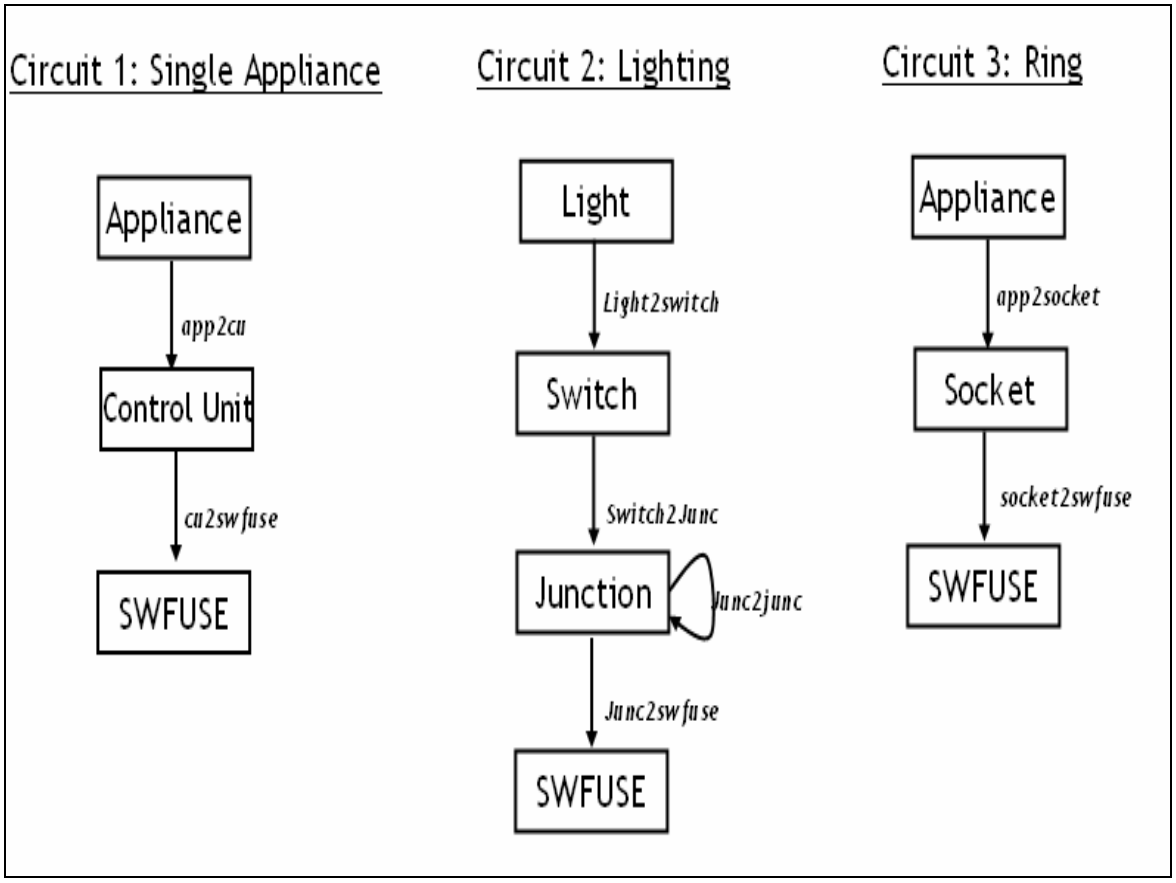

Figure 4-28: Domain Specific Models for each circuit

Output of Steps 7

Based on the electrical components in Figure 4-26, the following instances were created in Protégé to populate the model (Table 4-4: Domain Specific Model Instances).

Instance Name	Model Concept	Circuit #
SWFUSE1	SWFUSE	Circuit 1
SWFUSE2	SWFUSE	Circuit 2
SWFUSE3	SWFUSE	Circuit 3
Socket 1	Socket	Circuit 2
Socket 2	Socket	Circuit 2
TV1	Appliance	Circuit 2
Lamp1	Appliance	Circuit 2
Junction1	Junction	Circuit 3
Junction 2	Junction	Circuit 3
Switch 1	Switch	Circuit 3
Switch 2	Switch	Circuit 3
Light 1	Light	Circuit 3

Light 2	Light	Circuit 3
CU1	ControlUnit	Circuit 1
Cooker1	Appliance	Circuit 1

Table 4-4: Domain Specific Model Instances

During this process step, the electrical engineer was asked to enter the data into Protégé. While the data was being entered, two types of error were corrected as described below:

- Associating the wrong instance name with a Model concept. This error occurred due to the instance creation panel in Protégé that must have the correct concept name highlighted before the create instance operation is selected.
- Creating dependency relations between the wrong instances. This error occurred because of the Electrical Engineers lack of familiarity with the names used to identify the dependency relations (app2cu, app2socket) and instances (light 1, CU1). The domain model constraints will not allow the wrong type of concept to be entered but will allow any instance name to be entered, even if a dependency relation does not exist in reality between those elements.

4.10.5.2 Outputs of Dependency Analysis Exercise

The Electrical engineer wished to test the system by requesting which elements were dependent on each "SWFUSE" element specified in the system because this would effectively find all elements in each circuit.

To achieve this, the thesis author created an OWL axiom for each "SWFUSE" element and entered it into Protégé as shown in Table 4-5. The Pellet reasoner [Pellet] was used to infer the dependency elements in each circuit by computing the dependency elements for each "SWFUSE" instances.

Circuit #	*Axiom*	*Result*

1	`<owl:Class rdf:ID="Axiom_FUSE1">`
	`<owl:equivalentClass>`
	`<owl:Restriction>`
	`<owl:onProperty rdf:resource="#DependencyRelation"/>`
	`<owl:hasValue rdf:resource="#SWFUSE1_CT1"/>`
	`</owl:Restriction>`
	`</owl:equivalentClass>`
	`</owl:Class>`
		COOKER1
		CU1
		SWFUSE1_CT1

2	`<owl:Class rdf:ID="Axiom_FUSE2">`
	`<owl:equivalentClass>`
	`<owl:Restriction>`
	`<owl:onProperty rdf:resource="#DependencyRelation"/>`
	`<owl:hasValue rdf:resource="#SWFUSE2_CT2"/>`
	`</owl:Restriction>`
		LIGHT1
		SWITCH1
		SWFUSE2_CT2
		LIGHT2

	</owl:equivalentClass> </owl:Class>	SWITCH2 JUNCTION1 JUNCTION2
3	<owl:Class rdf:ID="Axiom_FUSE3"> <owl:equivalentClass> <owl:Restriction> <owl:onProperty rdf:resource="#DependencyRelation"/> <owl:hasValue rdf:resource="#SWFUSE3_CT3"/> </owl:Restriction> </owl:equivalentClass> </owl:Class>	TV1 SWFUSE3_CT3 LAMP1 SOCKET1 SOCKET2

Table 4-5: Reasoning Axioms for the domestic electrical circuits.

Given that the number of instances in the dependency model is small, the inferences above executed in 1-2 seconds for each case.

The electrical engineer checked the output of each inference and agreed that it was consistent with what he would expect.

4.10.5.3 Discussion of results

The issues that were identified during the process of applying the dependency metamodel to this new domain are discussed below.

Instance Data Entry

The data entry of instances into the model was identified as an issue by the electrical engineer. The process selected for this experiment used the Protégé tool to load instance data. Two distinct problems are discussed below.

The first problem concerned the instance-loading screen in Protégé. The user interface in Protégé provides all the available relationships for any defined concept (e.g. dependency attribute of level, strength, impact). It is not clear when using Protégé which attributes are mandatory and which are optional. During the exercise, the thesis author needed to instruct the electrical engineer on the meanings of each attribute.

The second problem concerned the time taken to load instances. Even with a small number of instances in this exercise, considerable time was spent on this step to ensure correct and consistent data entry. A number of errors in the data entry needed to be corrected as discussed earlier.

Separation of the Ontological Construct from the Model

The electrical engineer felt that the Protégé tool was not the most appropriate way to present the domain specific model as it contained many non-essential features that are related to building ontologies rather than the electrical domain.

The key issue was in the separation between the domain level (i.e. electrical domain) that the electrical engineer wishes to work at and the low level modelling constructs that are visible in Protégé.

Application of Dependency Attributes.

Two aspects of the application of the dependencies attributes were discussed.

The first aspect related to the dependency attributes for "Level", "Strength" and "Impact". The electrical engineer who wished to focus on the fault isolation in the scoped example did not use these attributes. It was noted that this information tends not to be formally represented in circuit diagrams and would be the subjective view of the circuit designer.

The second aspect related to dependency attributes concerns the usage of the "junc2junc" dependency relationship. Circuit three (Lighting circuit) contains symmetric and transitive/symmetric dependency relations. The "junc2junc" dependency relationship is not transitive so the dependencies will not propagate across this relationship. This means that a dependency analysis axiom for "LIGHT2" will yield the dependent elements LIGHT2, JUNCTION2, SWITCH2.

```
<owl:Class rdf:ID="Axiom_LIGHT2">
<owl:equivalentClass>
<owl:Restriction>
<owl:onProperty rdf:resource="#DependencyRelation"/>
<owl:hasValue rdf:resource="#LIGHT2 "/>
</owl:Restriction>
</owl:equivalentClass>
</owl:Class>
```

Code 13: Axiom for "LIGHT2"

The axiom for "LIGHT1" will yield the dependent elements "LIGHT1", "SWITCH1", "JUNCTION1" and "SWFUSE2_CT2".

In the electrical circuit domain, this behaviour is correct and as required by the electrical engineer because a failure on the LIGHT1 is unlikely to be caused by components associated with LIGHT2.

Multiple Domain Models

In this experiment, the metamodel was used to represent three types of circuit as shown in Figure 4-28. This approach was taken to limit the time spent by the electrical engineer during the exercise. The dependency metamodel provided sufficient

flexibility to design these three models. An alternative approach would be to create a single overall model to represent a more generalised model of an electrical circuit.

4.10.6 Discussion of Experimental Results

The ontology-based dependency metamodel enabled an electrical engineer to quickly create three domain specific models for a scoped domestic electrical circuit.

The automatic inference provided fast and accurate insight in the dependencies for any point in the electrical circuit. This information is useful when both attempting to isolate faults in the circuit as well as when adding new components to the circuit.

This study has highlighted the importance of tool support for the loading of instance data. However, this tool support requires access to a structured information store from where to load the instance data. In the case of the ontology-based information integration system, the mapping file provided an excellent structured source. In other domains, especially outside of the information technology space, this information may not be easily available.

4.10.7 Summary of Conclusions, Open Issues and Limitations

The dependency metamodel was used, under supervision, by an electrical engineer to carry out a dependency analysis of a scoped electrical circuit. The study showed that the metamodel can be applied in a relatively short time (2-3 hours). The study showed how different electrical circuit types can be supported by the metamodel.

The dependency axioms that were constructed to infer dependencies were similar to other use cases tested in this research. This is due to the abstract nature of the dependency metamodel that provides for dependency reasoning over architectural entities (AE). In this domain, the Architectural Entities represent points in the circuit from where a dependency analysis can be carried out.

Three areas of improvement have been identified as discussed below.

- More instructional information should have been provided on which concepts and attributes of the dependency metamodel are mandatory and which are optional. It was initially assumed by the electrical engineer that all concepts and attributes were mandatory.

- The metamodel could be enhanced to include more support for representing the different types of models. In this experiment, three models were created. Currently the metamodel only supports the concept of a graph, that has name and type attributes.
- The construction of axioms to infer dependencies is left to the domain specific model. The process to construct the domain specific models could be enhanced to provide a framework to support the creation of inferences related to the dependencies.

4.11 Summary of Evaluation

This research has been carried out in an iterative manner using an action-based methodology. Action based research involves an iterative inquiry process that leads to a refinement of the research question. The inquiry process was conducted using a series of experiments.

The aim of the first experiment was to discover the key issues related to integration performance when applying an ontology-based integration approach in an industrial context. The results of experiment one showed that while there is advantage to be gained by using the ontology-based approach, because the solution can cope with semantic heterogeneity using mappings, it is not easy to identify which mappings need to change when one of the underlying data sources changes. This was due to the complex nature of some of the mappings and the complex coupling between different parts of the integration system that the mappings create. Experiment one noted that approximately 33% of the mappings contained complex mappings functions. Following analysis of the results from experiment one, a hypothesis was developed that suggested that the complexity and coupling of the mappings would make the mappings difficult to evolve and that support for understanding the complexity and couplings would bring benefits to the integration system.

When a data source changes, the first step in evolving the mappings is to understand which parts of the system are impacted. Experiment two evaluates the hypothesis that this can be achieved by modelling the dependencies that exist between the parts of the ontology-based integration system.

The second experiment developed and evaluated an ontology-based dependency model (OBDM) that would support understanding of the complex coupled nature of mappings. The results of this experiment showed that a significant proportion, approximately 30%, of the mappings exhibit complicated dependency relationships. It was hypothesised that these mapping relationships are difficult to identify without tool support and thus makes the first step of mapping evolution difficult for integrators.

Experiment three confirmed the hypothesis by demonstrating the time consuming and error prone nature of this first step of mapping evolution (i.e. identification of mapping dependencies) through the use of a synthetic set of evolution needs. This was achieved by comparing the performance of the OBDM with a manual dependency analysis

process that was carried out by 18 users. The results of the experiment show that the ontology-based dependency model significantly outperforms the manual process in both accuracy and time. With the synthetic set of evolution needs, the ontology-based dependency model provides fast, accurate and automatic support for the first step of mapping evolution.

The fourth experiment tested the ontology-based dependency model as a new data source was introduced into an existing dataset and examined how the resultant set of evolution needs were coped with. The set of evolution needs arising was more unpredictable in comparison to the synthetic set designed for use in experiment three as the data set was taken from an industrial context. The experiment shows that the ontology-based dependency model and toolset enables the integration/ontology designer to quickly localise the impacted areas and allows analysis of the changes to proceed in an ordered fashion. The approach supports the mapping evolution process by providing global dependency views that allow the user to focus in on areas of high dependence initially and then to progressively drill down to the detail to understand the impact of each computed dependency. As noted in [Halevy et al. 2005, Zhou et al. 2006], this is one of the key challenges facing enterprise integration systems.

A corroborative study applied the ontology-based dependency metamodel that was created as part of experiment two, to build a dependency model for a domestic electrical circuit. The key ideas concerning dependency analysis and dependency models were presented to an electrical engineer who was asked to create a dependency model for an electrical circuit from a domestic setting. A domain model was created by the electrical engineer for the electrical circuit that contained four elements (Main Switch Board, Switch, Light and Consumer Device). The engineer was asked to build a model on paper using the dependency metamodel and domain elements. The engineer was coached by the thesis author during this process to ensure that the experiment focused on the metamodelling constructs and not on the Protégé [Protégé] or Pellet [Pellet] toolset.

5 CONCLUSIONS

This chapter describes how well the objectives of the thesis were achieved (Section 5.1), summarises the contributions made (Section 5.2), describes work that may be undertaken in the future (Section 5.3) and concludes with some final remarks (Section 5.4).

5.1 Objectives & Achievements

The research question in this thesis was defined in Chapter 1 as "*How and to what extent can a dependency model enhance integration performance by allowing for the identification of and support for the management of the mapping dependencies of an integration system?"*

Four objectives were derived to address the research question:

1) Perform a state of the art review of approaches for semantically linking local[23] schema and aggregate or global schema[24].

2) Research and develop a model to define the dependencies that arise when creating semantic links between schemas to support an ontology-based integration approach between local schemas and global schemas.

3) Research and develop a prototype tool capable of supporting this dependency modelling approach.

4) Evaluate the dependency model and tool using industrial use cases.

Each of these objectives and associated achievements are discussed in the following sections.

5.1.1 Objective One - State of the Art Review

The state of the art chapter was divided into three sections. Before the state of the art a background review of current information integration approaches and technologies was undertaken.

The first part of the state of the art reviewed the prior art in dependency and dependency analysis. The second part looked at approaches to schema and ontology

[23] Local schema refers to a schema that represents the local sources to be integrated.
[24] Global schema refers to a common view of sources to be integrated.

mapping management as they apply to management of semantic mappings. The third part reviewed the state of the art in ontology-based integration systems.

Review of Information Integration

The background review into information integration illustrated that research has been ongoing for at least 30 years in various forms but is as relevant today as ever. The detailed state of the art for integration focused on ontology-based approaches to data integration by providing a review of the fundamental ways to apply ontologies to the integration problem and then reviewing several recent ontology-based integration frameworks against these fundamentals.

The review showed how information integration is often cited as the biggest and most expensive challenge that information-technology organisations face and how information integration is thought to consume about 40% of their budget [Bernstein and Haas 2008]. In spite of many successes in information integration (e.g. relational databases, ETL [25]techniques, data federation techniques), the state of the art review illustrates the relevance of research in data integration today [Bernstein and Melnik 2007, Lowell Database Report 2003, IBM 2004, Halevy et al. 2005, and Zhou et al. 2006].

The role that ontologies play in supporting the resolution of semantic heterogeneity [Pollock 2002, Cruz and Xiao 2005, Calvanese et al. 2001, Noy 2004 and Wache et al. 2001] and how semantic mappings are used to create relationships between the ontologies and data sources of the systems to enable integration [Cruz and Xiao 2005, Noy 2004, Wache et al. 2001] was described. The review showed that as the ontology-based systems are scaled up, semantic mappings also need to grow and evolve [Bernstein and Melnik 2007, Velegrakis et al. 2003, Yu and Popa 2005, and Halevy et al. 2005]. Despite the broad usage of mappings across these approaches, it was found that there is little commonality in the approach to the management of the mappings [Bernstein and Melnik 2007, Doan and Halevy 2005, Halevy et al. 2005].

To investigate this management gap in the state of the art, experiment one (Section 4.2) was developed to discover the key issues related to integration performance when applying an ontology-based integration approach in an industrial context. This led to a number of important achievements that are described below.

[25] Extract, Transform and Load (ETL) is a data integration technique.

A test bed that represented a generalised ontology-based integration system using the hybrid ontology approach was developed as described in the design chapter. The test bed enabled the exploration of the semantic mappings that are at the heart of the ontology-based system in experiment one. The "Integration Quality" metric of the system was measured using the THALIA integration benchmark [Stonebraker 2005]. From the analysis of the results of the experiment a hypothesis was developed concerning the complex nature of the mappings and the complex coupling between different parts of the integration system that the mappings create. The test bed that was created in this thesis adhered to the fundamental approaches for using ontologies for integration as described in the state of the art review.

Mappings in Schema and Ontology Evolution

The state of the art in the management of schema and ontology mapping was reviewed by first looking at mapping usage in schema evolution and then reviewing the state of the art in ontology evolution.

While the mappings play a key role in the approaches to schema evolution, there are fundamental reasons why the approaches are not easily transferable [Kondylakis et al. 2009, Noy and Klein 2002].

Among these fundamental differences is that ontologies themselves are data that can be reasoned over to an extent that schemas cannot (e.g. a query on a database schema will usually result in a set of instance data, while a query on an ontology can result in both instance data and elements of the ontologies itself). Furthermore ontologies themselves incorporate explicit semantics of a domain that in the case of schema based systems tend to be incorporated into the application itself. The extra expressivity of the ontological domain descriptions means the mappings in the ontological domain contain semantic information themselves as illustrated by the fact that mappings are sometimes represented using ontological languages.

The author of this thesis believes that, based on the evidence from the state of the art review that the mappings in the ontology-based integrations systems are sufficiently different from the schema approaches that the mappings would benefit from an independent management approach. The ontology-based dependency modelling approach proposed in this thesis provides an approach for the management of mappings in the ontology-based integration domain in which the mappings are seen as

fundamental parts of the integration system that needs to be evolved when the data sources change.

The review showed described two of the most recent tools (PRISM workbench [Curino et al. 2008], Clio project [Miller et al. 2001]) to support schema evolution.

In the context of ontology-based integration systems, it was noted that these approaches, while relevant, may not be directly applicable due to the differences in both the usage and nature of mappings in the ontology-based integration domain. A number of differences were noted as follows:

- The more expressive nature of the ontology languages made it unclear if the approaches that use schema matching operators defined in [Curino et al. 2008] are relevant to the ontology domain.
- The process for schema mappings and schema evolution tends to be coupled and the lifecycle of each is not identified or managed separately.
- The formal semantics of ontology-based languages allow for the use of reasoning that can be used for consistency checking of evolved ontologies.

The ontology-based dependency modelling approach proposed in this thesis provides a new approach for the management of mappings in the ontology-based integration domain that is not covered by the state of the art.

The state of art review noted that the development of ontology is a complex process and recently much fruitful research has been carried out [Hepp et al. 2008] and is beginning to be realised in excellent tools such as the NeOn project [NeOn].

The review highlighted that the ongoing maintenance and evolution of the ontologies is also of critical importance for any industrial deployment of an ontology-based integration solution as noted in [Wache et al. 2001, Uschold and Gruniger 2004, Hepp et al. 2008]. The NeOn project [Hepp et al. 2008] provides an excellent, extensible framework for the development and management of ontologies.

The complex nature of mapping evolution was described [An and Topaloglou 2007] and the review revealed that the evolution of semantic mappings is still in its early stages [Hepp et al. 2008]. This was further confirmed by the review of current frameworks that use ontologies to support integration.

The ontology-based dependency modelling approach proposed in this thesis can support the ontology alignment lifecycle proposed in [Hepp et al. 2008] by automatically providing the candidate mappings that are dependent on the part of the ontology that is evolving.

Dependency

In the state of the art review on dependency, it was shown that dependencies and dependency analysis has been used across many domains such as distributed service management, fault management and software configuration management [Borner and Paech 2009, Varol and Bayrak 2010, Luo and Diao 2009, Drabble et al. 2009, Wang and Capretz 2009 and Maddox and Shin 2009]. The approach enabled valuable insight into the management of their respective systems by providing impact analysis caused by changes (e.g. faults or data updates) in the underlying systems.

Very few approaches presented in the state of the art provide formal representations of dependency that can be used to reason about dependencies. Most representations of dependency are based on simple notions of dependency without any behaviour aspects modelled as in the approach taken in this thesis. The models proposed in [Keller et al. 2000] and [Cox et al. 2001] provide useful insight into the descriptive attributes of dependency that are useful in the service management domain.

The ontology-based dependency modelling approach presented in this thesis describes two different types of dependency attribute i.e. behavioural attributes and descriptive attributes. While the descriptive attributes of the model are important, it is the behavioural attributes that enable the automatic reasoning over the ontology-based dependency model and thus provide the dependency analysis with the capability to automatically build chains of dependencies.

It was noted in the review that the processes to acquire instances to populate the dependency model are not explicitly specified and tend to use bespoke coded solutions to acquire the instance data [Ensel and Keller 2002, Keller et al. 2000, Borner and Paech 2009 and Drabble et al. 2009]. This makes any generalisation of the approaches difficult.

Summary

From the discussion above, the state of the art review identified the different approaches that can be taken to use ontologies to support semantic integrations. These

fundamental approaches were applied in the construction of the generalised ontology-based integration test bed used in experiment one and two.

The review highlights the hypothesis that semantic mappings can pose problems in ontology-based data integration systems due to the difficulty in evolving them when data sources change. This problem is exacerbated by the lack of mapping management approaches.

The review showed the value of dependency analysis as it has been applied in other domains but noted that the approaches are tightly coupled to the domain under test. This thesis has developed a dependency modelling approach to support the management of mappings in ontology-based integration systems as the data sources evolve.

5.1.2 Objective Two - Design of Ontology-Based Dependency Model

Following analysis of the results of experiment one, a hypothesis was developed that stated that the complexity and coupling of the mappings would make the mappings difficult to evolve. Furthermore support for understanding the mapping complexity and coupling would bring benefits to the integration system when the mappings need to be updated.

This thesis has demonstrated the complexity associated with mappings in the ontology-based integration systems in experiment two. This was achieved by using a model of dependencies to explicitly show the relationships between mappings and the rest of the integration system.

This was achieved by the development of a domain specific model in OWL [OWL] to represent the dependencies in the ontology-based integration system. This is called the ontology-based dependency model (OBDM).

The ontology-based dependency model was created using a metamodelling approach. The dependency metamodel that was created provided an extensible set of concepts related to modelling of dependencies and can be reused to build other dependency models. The dependency metamodel moved past the state of the art in dependency modelling due to its support for dependency attributes (behavioural and descriptive) and in its ability to enable reasoning about dependencies. The compact nature of the

metamodel enabled its application in a new domain as shown in the corroborative study in the evaluation chapter.

The selection of OWL to create the ontology-based dependency model and metamodel enabled automated reasoning about dependencies based on the formal semantics of the OWL constructs used in the dependency metamodel and model. This automated reasoning approach was used in the TomE tool described in Section 3.2.6.3 of the design chapter.

5.1.3 Objective Three - Design of Ontology-Based Dependency Model Tool (TomE)

A tool called TomE (Towards Ontology Mapping Evolution) was developed to instantiate the OBDM and to support the analysis of dependencies in the ontology-based integration system.

The TomE tool automatically computes the dependencies arising from the semantic mappings in the ontology-based integration test system. The tool was used to support experiment two, three and four.

The tool provides strong visualisation of the automatically computed dependencies by providing three separate graphical representations of the dependencies. The tool automatically populates the dependency model by reading the semantic mapping file from the ontology-based integration system.

The tool is an important achievement because it abstracts the ontological aspects of the dependency model from the user. This thesis has shown how the tool ensured fast and accurate computation of the dependencies across a range of different semantic mappings files in experiment two, three and four.

5.1.4 Objective Four - Evaluation of Dependency Modelling Approach

The performance of the dependency modelling approach that uses an ontology-based metamodel was measured in experiment two, three and four.

Experiment two demonstrated the three different types (non-overlapping, overlapping, function-based) of dependencies that can arise when semantic mappings are used in the

generalised ontology-based system. The existence of different types of dependencies supports the hypothesis that mappings are difficult to evolve because they exhibit complex dependency relations with other parts of the system.

Experiment three demonstrated that the automated dependency approach will significantly outperform manual process based approaches. Furthermore, the results of experiment three show that even the dependencies in a small number of mappings can present considerable difficulty in the absence of tool support. Knowledge of the underlying data set did not significantly improve the performance of the manual approach.

Experiment four demonstrated how the dependency modelling approach and the TomE tool can be used to support the evolution of mappings when a data source changes. The dependency modelling approach and TomE tool enabled the fast and accurate identification of the mappings that were impacted by the introduction of a new data source in the generalised ontology-based integration system.

The corroborative study provided an indication of the genericity of the dependency metamodel by applying the ontology-based metamodel in a different domain. The study showed that an electrical engineer could create a dependency model and carry out dependency analysis using the metamodel. This is important because it provides evidence of the straight forward approach that can be taken to apply the metamodel in a new domain.

5.2 Contribution

The major contribution of this work is the ontology-based dependency model (OBDM) that can represent the dependencies that occur between mappings, ontologies and databases in an ontology-based integration system. The ontology-based dependency model will be beneficial to system integrators when developing approaches to improve the ability of the enterprise integration systems to evolve when data sources change.

In the context of the generalised ontology-based integration system, the dependency modelling approach is automatic since it can decompose the mapping file, compute and visualise dependencies without human intervention. As shown in experiment three, it significantly outperforms manual process oriented approaches for both accuracy and time measurements. The approach provides useful insight into the mapping evolution in a fast and reliable way by providing three levels of dependency analysis, complete

with visualisation and navigation of the dependency graphs. A case study (experiment four) that introduced a new data source to the integration system demonstrated the relevance of the dependency model and toolset to the evolution problem by providing analysis of the dependencies. The approach supports the evolution process by providing global dependency views that allow the user to focus in on areas of high dependence initially and then to progressively drill down to the detail to understand the impact of each computed dependency. The dependency model is novel since it automatically computes the dependency relationships. The automation is achieved through the instrumental usage of ontological reasoning over different forms of dependency relation (e.g. transitive, symmetric). This approach requires coding only to invoke the ontological reasoner. To the authors knowledge, an ontology-based dependency metamodel has not been published before that has support for both behavioural and descriptive attributes and that can enable reasoning over the dependency relationships in the model to enable automatic computation of dependencies.

The dependency modelling approach makes the dependencies that exist in the system explicit thus making analysis of dependencies and mapping evolution easier.

The approach does not require instrumentation of the integration system and thus does not impact the processing of the integration system while the dependency analysis is taking place. The ontology-based dependency model (OBDM) was case studied against industrial data from real systems from the Alcatel-Lucent supply chain that provided a challenging set of requirements for the system. The results of the experiments indicate how the ontology-based dependency model and tool enable the integration specialist to quickly identify all the impacts of a complex set of changes to the data sources. By providing progressive detail of the dependencies, the integration specialist can quickly focus and assess what needs to be changed in the system. The results show that dependencies found can also be used to develop targeted regression testing after the integration system has been updated. This analysis is useful for integration systems developers who wish to understand the complexity involved in the evolution of mappings in an industrial context.

A minor contribution is the ontology-based dependency metamodel from which the domain specific dependency model was created. The ontology-based dependency metamodel could be beneficial to management systems (e.g. service and fault

management) which need to model dependencies between parts of the system as described in the state of the art review of dependency (Section 2.3.1). The genericity of the metamodel has been tested across two large industrial datasets that originated from a dynamic industrial environment with multiple IT systems and multiple processes. A corroborative study was carried out to demonstrate the application of the metamodel in an entirely different domain (i.e. dependency analysis in a domestic electrical circuit). The compact nature of the metamodel facilitates design flexibility, behaviour reuse and scalability. Design flexibility is achieved since the metamodel enables domain specific models to select those features of the metamodel it wishes to realise. Reuse is achieved because the domain specific models inherit the important formal semantics associated with dependency relations (e.g. transitivity). The metamodel and domain specific model can be independently evolved with care. A process has been defined that describes the steps required to create domain specific models from the dependency metamodel. This ensures that the system is extensible because the technique and model to manage mapping evolution can be adapted to cater for other mapping formats by simply decomposing the mapping format into the core architectural entities. The decomposition process requires the model creator to encode only the first level of dependency for each node thus reducing the breadth of domain knowledge any single model creator requires.

Peer review publications

The design of the generalised ontology-based integration test system and the setup, results and conclusions of experiment one were published in:

> Aidan Boran, Declan O'Sullivan and Vincent Wade, A Case Study of an Ontology-Driven Dynamic Data Integration in a Telecommunications Supply Chain. *Proceedings of the Workshop on the First Industrial Results of Semantic Technologies (FIRST2007) at ISWC/ASWC2007, Busan, South Korea,* 2007.

The design of the ontology-based dependency model and the result of experiment two were published in:

> Aidan Boran, Declan O'Sullivan and Vincent Wade, Managing Ontology Based Integration Systems using Dependencies. *Proceedings of the Workshop on the Managing of Ubiquitous Communications and Services Workshop (MUCS) at PerCom 2010, Mannheim,Germany ,* 2010.

The design of the ontology-based dependency metamodel, model and toolset was published in:

> Aidan Boran, Declan O'Sullivan and Vincent Wade, A Dependency Modelling Approach for the Management of Ontology Based Integration systems. Network Operations and Management Symposium (NOMS), Osaka, Japan, *2010*.

It is planned to submit the ontology-based dependency modelling approach using dependency metamodel to selected journals in the service and data management areas.

5.3 Future Work

The experiments that were carried out in this research highlighted a number of limitations as discussed at the end of each experiment. These limitations afford the opportunity for further research. This further research is classified here according to whether they impact the performance or the functionality of the dependency modelling approach.

5.3.1 Future work related to the performance of the dependency model

This section describes future work that could be undertaken to improve or further verify the performance of the dependency modelling approach.

Runtime Performance of Ontology-Based Integration Approaches

The THALIA benchmark provided a simple measure (i.e. a score out of twelve) of the ability of the system to perform integrations across the twelve types of heterogeneity. A more comprehensive suite of performance measurements (e.g. runtime performance) would be needed to confirm the integration systems suitability for industrial deployment. These aspects of performance were not tested in this research as the focus was to investigate the complexity of the mappings.

The THALIA benchmark system does not provide quantitative data on how much effort is needed to run each test. This is important because, while a THALIA integration test may pass, it may require costly manual intervention (e.g. mapping updates) that would impact the scalability of the system. To address this in experiment one an effort classification was developed and used that provides qualitative estimation

of the effort needed for each test in THALIA. Further research could be undertaken to develop a more sophisticated quantitative measure of the effort for each THALIA test.

Mapping Formats

Only one mapping format (INRIA [Euzenat 2004]) was tested as part of the experiments. Other mappings formats could cause dependencies between different parts of the integration system that were not tested in this experiment. However, the approach taken in the design of the dependency metamodel and model creation process means that irrespective of the mapping format, once the mapping decomposition process is carried out, the dependency model will be able to support other mapping formats. Further research could be undertaken to verify the performance of the dependency modelling approach using other mapping formats.

5.3.2 Future work related to the functionality of the dependency model

This section describes future work that could be undertaken to improve the functionality of the dependency modelling approach.

TomE Tool Implementation

The current implementation of the visualisation of the dependency chain in TomE does not display a graphical representation of the function associated with each mapping point. This could be improved by updating the dependency factory code in TomE to add appropriate GraphML nodes for functions.

In the current implementation of the TomE tool, the function names need to be manually extracted from the function descriptions in the generalised ontology-based integration system. While the TomE tool loads the function descriptions automatically from a user specified file (Section 3.2.6), the file has to be prepared manually by examining the mapping file and the code for each mapping function. Further work could be carried out here to automate the collection of the function names and the parameter names.

Rule Enhancement for the Dependency Model

The OBDM currently does not support the automatic classification of the dependency types found. The addition of a rule capability to the domain specific model would

provide the dependency modeller with the ability to define rules to support this classification. This could be achieved by research into the application of Semantic Web Rule Language [SWRL] to the dependency model.

The Role of Dependency Analysis in the Mapping Evolution Process

The industrial data used in the experiment four came from the logistics based use case and focused on updating mappings rather than the creation of new mappings or the deletion of existing mappings. A process was defined to describe the usage of the TomE tool for the update case. A detailed process for the usage of the TomE tool for all cases (update, new, delete mapping) should be defined that will cover the sequence of tasks needed to carry out the dependency analysis and will define how the dependency analysis interacts with the mapping evolution process.

Application of Dependency Model to other domains

The dependency modelling approach and dependency metamodel is proposed to be used in a number of other application areas. The FAME Strategic Research Cluster [FAME] in Ireland will use the dependency modelling approach as part of the strategy to manage ontology mappings for the FAME architecture. Within a research project in Bell Labs, the ontology-based dependency metamodel is under investigation to support the management of dependencies between web service invocations. The dependency model may be included in a larger data management ontology which includes concepts to represent provenance of the data sources which are represented by the domain ontology.

5.4 Final Remarks

The explosion of information that is available in the internet and the enterprise has created the need for dynamic data integration technologies that can evolve as the information evolves. The emerging approaches to data integration that use ontologies and mappings promise to make data integration systems more flexible in the face of evolving data sources.

The author believes that the ontology-based dependency model described in this research provides a framework that can be used in data integration toolsets to support the data integration industry as it takes the first steps towards full mapping management.

6 Bibliography

Abels et al. 2008 Sven Abels, Stuart Campbell and Hamzeh Sheikhhasan. Stasis - Creating an Eclipse Based Semantic Mapping Platform. In eChallenges 2008.

An and Topaloglou 2007 Yuan An and Thodoros Topaloglou. Maintaining Semantic Mappings between Database Schemas and Ontologies. In the Proceedings of the Joint ODBIS and SWDB Workshop on Semantic Web, Ontologies, Databases 2007 in conjunction with VLDB07, Vienna, Austria.

Beneventano et al. 2003 Domenico Beneventano, Sonia Bergamaschi, Francesco Guerra and Maurizio Vincini. Synthesizing an Integrated Ontology. IEEE Internet Computing 7(5):42–51, 2003.

Beneventano et al. 2009 Domenico Beneventano, Mirko Orsini, Laura Po, Antonio Sala and Serena Sorrentino. "An Ontology-Based Data Integration System for Data and Multimedia Sources". pp. 606-611. 2009 IEEE International Conference on Semantic Computing, 2009.

Bernstein and Melnik 2007 Philip A. Bernstein and Sergey Melnik. Model Management 2.0: Manipulating Richer Mappings. International Conference on Management of Data. Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data.

Bernstein and Haas 2008 Philip A. Bernstein and Laura M. Haas 2008. Information Integration in the Enterprise. Communications of the ACM. Volume 51, Issue 9 72-79. (September 2008). DOI= http://doi.acm.org/10.1145/1378727.1378745

Biffl et al. 2010 Stefan Biffl, Wikan Danar Sunindyo and Thomas Moser. "Semantic Integration of Heterogeneous Data Sources for Monitoring Frequent-Release Software Projects". pp. 360-367. 2010 International Conference on Complex, Intelligent and Software Intensive Systems, 2010.

Böhm et al 2008 Matthias Böhm, Dirk Habich, Wolfgang Lehner and Uwe Wloka. "DIPBench: An Independent Benchmark for Data-Intensive Integration Processes". Data Engineering Workshop, 2008. ICDEW 2008. IEEE 24th International Conference. Vol., No. pp.214-221, 7-12 April 2008. DOI: 10.1109/ICDEW.2008.4498321.

Böhme and Rahm 2001 Timo Böhme and Erhard Rahm."Xmach-1: A Benchmark for XML Data Management." in BTW, 2001, pp. 264–273.

Borner and Paech 2009 Lars Borner and Barbara Paech. "Using Dependency Information to Select the Test Focus in the Integration Testing Process". Practice and Research Techniques, Testing: Academic and Industrial Conference pp. 135-143, 2009 Testing: Academic and Industrial Conference - Practice and Research Techniques, 2009.

Brown et al. 2001 Aaron Brown, Guatam Kar, Alexander Keller. An Active Approach to Characterizing Dynamic Dependencies for Problem Determination in a Distributed Environment. Proc 7th IFIP/IEEE International Symposium on Integrated Network Management (IM VII), Seattle, WA, May 2001.

Calvanese et al. 2001 Diego Calvanese, Guiseppe De Giacomo and Maurizio Lenzerini. Ontology of Integration and Integration of Ontologies. In Description Logic Workshop (DL 2001), pages 10–19, 2001.

Choi et al. 2006 Namyoun Choi, Il-Yeol Song and Hyoil Han. A Survey on Ontology Mapping, ACM SIGMOD Record, Volume 35 Number 3, pp.34-41, September 2006.

Cleve and Hainaut 2006 Anthony Cleve and Jean Luc Hainaut. Co-Transformations in Database Applications Evolution. Generative and Transformational Techniques in Software Engineering, pages 409–421, 2006.

Corcho and Gomez-Perez 2000 Oscar Corcho and Asuncion Gomez-Perez. Evaluating Knowledge Representation and Reasoning Capabilities of Ontology Specification Languages. In Proceedings of the ECAI 2000 Workshop on Applications of Ontologies and Problem-Solving Methods, Berlin, 2000.

Cox et al. 2001 Lisa Cox, Dr. David Skipper and Dr. Harry S. Delugach. Dependency Analysis Using Conceptual Graphs. In Proceedings of the 9th International Conference on Conceptual Structures, ICCS 2001.

Crubezy and Musen 2003 Monica Crubezy and Mark A. Musen. Ontologies in Support of Problem Solving. S. Staab and R. Studer (eds). Handbook on Ontologies. Pages 321–342. Springer, 2003.

Cruz and Xiao 2005 Isabel F. Cruz and Huiyong Xiao. "The Role of Ontologies in Data Integration". Journal of Engineering Intelligent Systems, Vol. 13 (4), pp. 245- 252, 2005.

Cruz et al. 2004 Isabel F. Cruz, Huiyong Xiao and Feihong Hsu. "An Ontology-Based Framework for XML Semantic Integration". pp. 217-226. International Database Engineering and Applications Symposium (IDEAS'04) 2004.

Curino et al. 2008 Carlo A. Curino, Hyun J. Moon and Carlo Zaniolo. Graceful Database Schema Evolution: The PRISM Workbench. Proc. VLDB Endow. 1, 1 (Aug. 2008) 761-772. DOI= http://doi.acm.org/10.1145/1453856.1453939

Deng et al. 2004 Yu Deng, Harumi Kuno and Kevin Smathers. Managing the Evolution of Simple and Complex Mappings between Loosely-Coupled Systems. http://www.hpl.hp.com/techreports/2004/HPL-2004-68.html

Doan and Halevy 2005 AnHai Doan and Alon Y. Halevy. Semantic Integration Research in the Database Community, A Brief Survey. A.I. Magazine, Volume 26, Issue 1 (March 2005). Special issue on Semantic Integration. Pages: 83 – 94. Year of Publication: 2005.

Dong and Linpeng 2008 Li Dong and Huang Linpeng. "A Framework for Ontology-Based Data Integration" pp. 207-214. 2008 International Conference on Internet Computing in Science and Engineering.

Dou et al. 2003 Dejing Dou, Drew McDermott and Peishin Qi. Ontology Translation on the Semantic Web. In the International Conference on Ontologies, Databases and Applications of Semantics 2003.

Drabble et al. 2009 Brian Drabble, Tim Black, Chris Kinzig and Gary Whitted. "Ontology Based Dependency Analysis: Understanding the Impacts of Decisions in a Collaborative Environment". International Symposium on Collaborative Technologies and Systems, 2009. pp. 10-17.

Dreo Rodosek and Lewis 2001 Gabi Dreo Rodosek and Lundy Lewis. Dynamic Service Provisioning: A User- Centric Approach. O. Festor and A. Pras (eds.), In Proceedings of the 12th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management (DSOM 2001) INRIA Press, Nancy, France. pp. 37–48, October 2001.

D2RQ API D2RQ API. http://sites.wiwiss.fuberlin.de/suhl/bizer/D2RQ/

Eclipse Eclipse. Integrated Development Environment. http://www.eclipse.org/

Ensel 2001 Christian Ensel. A Scalable Approach to Automated Service Dependency Modeling in Heterogeneous Environments. In the 5th International Enterprise Distributed Object Computing Conference (EDOC 2001). IEEE Publishing, IEEE, Seattle, USA, September, 2001.

Ensel and Keller 2002 Christian Ensel and Alexander Keller. An Approach for Managing Service Dependencies with XML and the Resource Description Framework. Journal of Network Systems Management. Volume 10 Issue 2 (June 2002) Pages: 147- 170.DOI= http://dx.doi.org/10.1023/A:1015902715532

Euzenat 2004 Jérôme Euzenat. INRIA, A Format for Ontology Alignment. An API for Ontology Alignment. The 3rd Conference on International Semantic Web Conference (ISWC), Hiroshima (Japan) 2004. Lecture notes in Computer Science 3298:698-712, 2004. http://alignapi.gforge.inria.fr/format.html

FAME Federated, Autonomic Management of End-to-End Communications Services (FAME). Strategic Research Cluster (SRC). http://www.fame.ie

Fisher 2004 Dr. Robert J. Fisher. What is Action Research? An Introduction to Action Research for Community Development. Paper prepared for Working Party Meeting on Action Research for Integrated Community Development, 5-8 April 2004, Tehran, Islamic Republic of Iran.

Fraissé 1986 Roland Fraissé. Theory of Relations, First Edition, 1986. North-Holland. ISBN 988044 4505422 044 4505423

Fu et al. 2008 Kui Fu, Guihua Nie, Donglin Chen and Huimin Wang. "A Semantic Integration Framework for E-Business and Logistics Systems". pp. 394-397. 2008 International Conference on Computer Science and Software Engineering.

Gilliland 2002 Michael Gilliland. Is Forecasting a Waste of Time? Supply Chain Management Review, July/August 2002.

GMF Eclipse Graphical Modeling Framework. http://www.eclipse.org/gmf/

Gomez-Perez 1998 Asunción Gomez-Perez. Knowledge Sharing and Reuse. The Handbook on Applied Expert Systems. ED CRC Press 1998.

GraphML GraphML, An XML format for Graphs. http://graphml.graphdrawing.org/

Gruber 1993 Thomas R. Gruber. A Translation Approach to Portable Ontology Specifications. Knowledge Acquisition. Volume 5, Issue 2 (Jun. 1993) Pages 199-220. DOI= http://dx.doi.org/10.1006/knac.1993.1008

Gruschke 1998 Boris Gruschke. Integrated Event Management: Event Correlation Using Dependency Graphs. International Workshop on Distributed Systems: Operations and Management 1998 (DSOM 98).

Halevy 2001 Alon Y. Halevy. "Answering Queries Using Views: A Survey," The VLDB Journal, Vol. 10 (4), pp. 270-294, 2001.

Halevy 2005 Alon Y. Halevy. Why Your Data Won't Mix. Queue 3, 8 (October 2005), 50- 58. DOI= http://doi.acm.org/10.1145/1103822.1103836

Halevy et al. 2005 Alon Y. Halevy, N. Ashish, D. Bitton, M. Carey , D. Draper, J. Pollock, A. Rosenthal and V. Sikka. Enterprise Information Integration: Successes, Challenges and Controversies, ACM SIGMOD International Conference on Management of Data. A.C.M., Baltimore, 2005. pp. IIS-ISI. ISBN:1-59593-060-4

Halevy et al. 2006 A.Y. Halevy, A. Rajaraman and J. Ordille 2006. Data Integration: The Teenage Years. In Proceedings of the 32nd International Conference on Very Large Data Bases (Seoul, Korea, September 12 - 15, 2006). U. Dayal, K. Whang, D. Lomet, G. Alonso, G. Lohman, M. Kersten, S. K. Cha, and Y. Kim, Eds. Very Large Data Bases. VLDB Endowment, 9-16.

Harth et al. 2004 F. Martin-Recuerda Harth, A. Harth et al. D2.1 Report on Requirements Analysis and State of the Art (WP2-Ontology Management Version 1.00), FP6 DIP Project, FP-507483. http://dip.semanticweb.org/deliverables.html August 31st, 2004

Haas 2007 Laura M. Haas. Beauty and The Beast: The Theory and Practice of Information Integration. International Conference on Database Theory. Barcelona, Spain, January 2007. 28–43.

Hendrik et al 2009 Hendrik Thomas, Declan O'Sullivan and Rob Brennan , Ontology Mapping Representations: a Pragmatic Evaluation , International Conference on Software Engineering and Knowledge Engineering, Boston, USA, July 1-3, 2009, Knowledge Systems Institute Graduate School, 2009, pp228-232

Hepp et al. 2008 M. Hepp, P. Leenheer, A. Moor and Y. Sure (Eds.). Ontology Management Semantic Web, Semantic Web Services and Business Applications. Springer Books 2008.

Huynh et al. 2007 David Huynh, Robert Miller and David Karger. Potluck: Data Mash-Up Tool for Casual Users. ISWC 2007-11.

IBM 2004 IBM Business Consulting Services: Your Turn. The Global CEO Study 2004. Available from http://www.bitpipe.com/detail/RES/1129048329_469.html

ITU-T TMN ITU-T TMN. Telecommunication Management Network Standardisation. http://www.itu.int/ITU-T/

Jena Jena Semantic Web Framework. http://jena.sourceforge.net/

Kar et al. 2000 Guatam Kar, Alexander Keller and S. Calo. Managing Application Services over Service Provider Networks: Architecture and Dependency Analysis. Proceedings of the Seventh IEEE/IFIP Network Operations and Management Symposium (NOMS 2000), Honolulu, HI, 2000.

Kalfoglou and Schorlemmer 2003 Y. Kalfoglou and M. Schorlemmer. Ontology Mapping: The State of the Art. The Knowledge Engineering Review, 18(1):1–31, 2003.

Katker and Paterok 1997 S. Katker and M. Paterok. Fault Isolation and Event Correlation for Integrated Fault Management. A.A. Lazar, R. Saracco and R. Stadler (eds.) In Proceedings of the 5th IFIP/IEEE International Symposium on Integrated Network Management, Chapman and Hall, San Diego, California. pp. 583–596, May 1997.

KAON KAON. The Karlsruhe Ontology and Semantic Web Tool Suite. An open-source ontology management infrastructure from the University of Karlsruhe. http://kaon.semanticweb.org/

Keller et al. 2000 Alexander Keller, U. Blumenthal and Guatam Kar. Classification and Computation of Dependencies for Distributed Management. In Proceedings of the 5th IEEE Symposium on Computers and Communications (ISCC 2000) (July 04 - 06, 2000). ISCC. IEEE Computer Society, Washington, DC, 78.

Kondylakis et al. 2009 H. Kondylakis, G. Flouris and D. Plexousakis. Ontology and Schema Evolution in Data Integration: Review and Assessment. In Proceedings of the Confederated International Conferences, Coopis, Doa, Is, and ODBASE 2009. On the Move to Meaningful Internet Systems: Part II (Vilamoura, Portugal, November 01 - 06, 2009). R. Meersman, T. Dillon, and P. Herrero, Eds. Lecture Notes in Computer Science, Vol. 5871. Springer-Verlag, Berlin, Heidelberg, 932-947. DOI= http://dx.doi.org/10.1007/978-3-642-05151-7_14

Kwak and Yong 2008 Jung-Ae Kwak and Hwan-Seung Yong. "An Approach to Ontology-Based Semantic Integration for PLM Object". pp. 19-26. 2008 IEEE International Workshop on Semantic Computing and Applications.

Lenzerini 2002 Maurizio Lenzerini. Data Integration: A Theoretical Perspective. In Proceedings of the 21st ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (Madison, Wisconsin, June 03 - 05, 2002). PODS 2002. ACM, New York, NY, 233-246. DOI= http://doi.acm.org/10.1145/543613.543644

Luo and Diao 2009 Daizhong Luo and Shanhui Diao. "Feature Dependency Modeling for Software Product Line". International Conference on Computer Engineering and Technology, 2009. pp. 256-260.

Lowell Database Report 2003 Lowell Database Report. Attendees at the Lowell Workshop were: Serge Abiteboul, Rakesh Agrawal, Phil Bernstein, Mike Carey, Stefano Ceri, Bruce Croft, David DeWitt, Mike Franklin, Hector Garcia Molina, Dieter Gawlick, Jim Gray, Laura Haas, Alon Halevy, Joe Hellerstein, Yannis Ioannidis, Martin Kersten, Michael Pazzani, Mike Lesk, David Maier, Jeff Naughton, Hans Schek, Timos Sellis, Avi Silberschatz, Mike Stonebraker, Rick Snodgrass, Jeff Ullman, Gerhard Weikum, Jennifer Widom, and Stan Zdonik. Slides and some detailed notes from the event are at http://research.microsoft.com/~gray/lowell/.

Maddox and Shin 2009 Jeffrey Maddox and Dong-Guk Shin. "Applying Relational Dependency Discovery Framework to Geo-spatial Data Mining". International Conference on Information and Multimedia Technology, 2009. pp. 10-14.

Maedche et al. 2002 A. Maedche, B. Motik, N. Silva and R. Volz. MAFRA - A Mapping Framework for Distributed Ontologies. In the 13th European Conference on Knowledge Engineering and Knowledge Management EKAW, Madrid, Spain, 2002.

Miller et al. 2001 R.J. Miller, M.A. Hernández, L.M. Haas, L. Yan, C.T. Howard Ho, R. Fagin, and L. Popa. 2001. The Clio Project: Managing Heterogeneity. SIGMOD Rec. 30, 1 (Mar. 2001), 78-83. DOI= http://doi.acm.org/10.1145/373626.373713

MySQL MySQL. Open source database. http://www.mysql.com/

NeOn 2005 NeOn Project. NeOn is a project involving 14 European Partners and co- funded by the European Commission's Sixth Framework Programme under grant number IST-2005-027595. http://www.neon-project.org

Noy and Klein 2002 N.F. Noy and M. Klein. Ontology Evolution: Not the Same as Schema Evolution. Smi-2002-0926, University of Stanford, Stanford Medical Informatics, USA, 2002.

Noy 2004 N.F. Noy. "Semantic Integration: A Survey of Ontology Based Approaches" SIGMOD Record, Vol. 33 (4), December 2004.

Noy and Musen 2000 F. Noy and M.A. Musen. PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment. In Proceedings of the 17th National Conference on Artificial Intelligence and 12th Conference on Innovative Applications of Artificial Intelligence (AAAI/IAAI 2000). Pages 450–455, 2000.

O'Brien R. 2001 R. O'Brien 2001. Um Exame da Abordagem Metodológica da Pesquisa Ação. [An Overview of the Methodological Approach of Action Research]. In Roberto Richardson (Ed.), Teoria e Prática da Pesquisa Ação [Theory and Practice of Action Research]. João Pessoa, Brazil: Universidade Federal da Paraíba. (English version) Available: http://www.web.ca/~robrien/papers/arfinal.html (Accessed 20/1/2002)

O'Sullivan D. 2005 Declan O'Sullivan. PhD Thesis. The OISIN Framework: Ontology Interoperability in Support of Semantic Interoperability. Trinity College Dublin. December 2005.

OSI GRM OSI General Relationship Model ISO/IEC CD 10165-7, Information Technology - Open Systems Interconnection - Structure of Management Information - Part 7: General Relationship Model.

Othayoth and Poess 2006 R. Othayoth and M. Poess. "The Making of tpc-ds" in VLDB 2006, pp. 1049–1058.

OWL Web Ontology Language. http://www.w3.org/TR/owl-ref/ W3C Recommendation 10 February 2004

OWL-QL OWL-QL. http://www-ksl.stanford.edu/projects/owl-ql/

Pellet Pellet OWL Reasoner. http://clarkparsia.com/pellet

Pollock 2002 J. Pollock. Integration's Dirty Little Secret: It's a Matter of Semantics" Whitepaper. Modulant, The Interoperability Company; February 2002.

Prefuse Prefuse. Java based visualization toolkit. http://prefuse.org/

Protégé Protégé Ontology Editor and Knowledge-base Framework.http://protege.stanford.edu/

R The R project for Statisical Computing. http://www.r-project.org/

Ra 2005 Young-Gook Ra. Relational Schema Evolution for Program Independency. Intelligent Information Technology, pages 273–281, 2005.

Rahm and Bernstein 2006 E. Rahm and P.A. Bernstein. An Online Bibliography on Schema Evolution. SIGMOD Rec. 35, 4 (Dec. 2006),30-31. DOI=http://doi.acm.org/10.1145/1228268.1228273

RDF Resource Description Framework (RDF): Concepts and Abstract Syntax. W3C Recommendation 10 February 2004 http://www.w3.org/TR/rdf-concepts/

Sangal et al. 2005 Neeraj Sangal, Ev Jordan, Vineet Sinha, and Daniel Jackson. Using Dependency Models to Manage Complex Software Architecture. 20th Annual ACM SIGPLAN Conference on Object-Oriented Programming Systems (OOPSLA 2005).

SAX The Simple API for XML. http://www.saxproject.org/

Seidenberg and Rector 2006 Julian Seidenberg and Alan Rector. Representing Transitive Propagation in OWL. 25th International Conference on Conceptual Modeling, Tuscon, AZ, USA. November 2006. Pages 255-266.

Sheth et al 1999 A.P. Sheth. Changing Focus on Interoperability in Information Systems: From System, Syntax, Structure to Semantics. M.F. Goodchild, M.J. Egenhofer, R. Fegeas, and C.A. Kottman (eds.). KLUWER INTERNATIONAL SERIES IN ENGINEERING AND COMPUTER SCIENCE. 1999, ISSUE 495, pages 5-30

Sjoberg 1993 D. Sjoberg. Quantifying Schema Evolution, Information and Software Technology Journal, Volume 35, Number 1, pp. 35–54, 1993.

SPARQL SPARQL Query Language for RDF W3C Recommendation 15 January 2008. http://www.w3.org/TR/rdf-sparql-query/

Stonebraker 2005 M. Stonebraker. THALIA - Integration Benchmark. Presentation at ICDE 2005, April 6, 2005. http://www.cise.ufl.edu/research/dbintegrate/thalia/

Stojanovic 2002 L. Stojanovic, A. Maedche, B. Motik and N. Stojanovic. User-driven Ontology Evolution Management. In Proceedings of the 13th European Conference on Knowledge Engineering and Knowledge Management EKAW, Volume 2473 of Lecture Notes in Computer Science, pages 285 – 300, Siguenza, Spain, October 1-4 2002.

SWRL SWRL : A Semantic Web Rule Language Combining OWL and RuleML. W3C Member Submission 21 May 2004. http://www.w3.org/Submission/SWRL/

Topic Maps Topic Maps. A Standard for the Representation and Interchange of Knowledge. http://www.isotopicmaps.org/tmrm/

UML The Unified Modeling Language (UML). http://www.uml.org/

Uschold and Gruniger 2004 M. Uschold and M. Gruniger. Ontologies and Semantics for Seamless Connectivity. SIGMOD Record, Vol 33, No. 4, December 2004.

Varol and Bayrak 2010 Cihan Varol and Coskun Bayrak, "Business Process Automation Based on Dependencies," Information, Process, and Knowledge Management, International Conference on, pp. 17-22, 2010 Second International Conference on Information, Process, and Knowledge Management, 2010.

Velegrakis et al. 2003 Yannis Velegrakis, Renee Miller and Lucian Popa. Mapping Adaptation Under Evolving Schemas. Proceedings of the 29th International Conference on Very Large Data Bases - Volume 29, 2003.

Wache et al. 2001 H. Wache et al. Ontology-Based Integration of Information – A Survey of Existing Approaches. In Proceedings of the IJCAI-01 Workshop on Ontologies and Information Sharing, 2001.

Wang and Capretz 2009 Shuying Wang and Miriam A.M. Capretz. "A Dependency Impact Analysis Model for Web Services Evolution". Web Services. 2009 IEEE International Conference on Web Services, 2009. pp. 359-365.

WSML Web Service Modelling Language (WSML). http://www.wsmo.org/wsml/wsml-syntax.

WSMT Web Service Modelling Toolkit (WSMT). A development environment for

Wu et al. 2006 Z. Wu, H. Chen, H. Wang, Y. Wang, Y. Mao, J. Tang and C. Zhou. Dartgrid: A Semantic Web Toolkit for Integrating Heterogeneous Relational Databases. In Semantic Web Challenge at 4th International Semantic Web Conference, Athens, USA, November 2006.

XML Path Language XML Path Language (XPath) 2.0 W3C Recommendation 23 January 2007. http://www.w3.org/TR/xpath20/

Yu and Popa 2005 C. Yu and L. Popa. Semantic Adaptation of Schema Mappings when Schemas Evolve. In Proceedings of the 31st International Conference on Very Large Data Bases (Trondheim, Norway, August 30- September 02, 2005). Very Large Data Bases. VLDB Endowment, 1006-1017.

Zablith 2009 F. Zablith. Evolva: A Comprehensive Approach to Ontology Evolution.2009 European Semantic Web Conference (ESWC) PhD Symposium, Crete, Greece. Proceedings of the 6th European Semantic Web Conference, LNCS 5554, (eds.) L. Aroyo et al., pp. 944-948, Springer-Verlag, Berlin, Heidelberg.

Zhou and Wang 2006 Jingtao Zhou and Mingwei Wang 2006. Semantic Integration of Enterprise Information: Challenges and Basic Principles. Lecture Notes in Computer Science. Springer Berlin / Heidelberg ISBN978-3-540-38329-1 Pages 219-233. September 01, 2006.

Zhou et al. 2006 Zhou, Jingtao, Wang, Mingwei, Zhao, Han, 2006, in International Federation for Information Processing (IFIP), Volume 207, Knowledge Enterprise: Intelligent Strategies In Product Design, Manufacturing, and Management, eds. K. Wang, Kovacs G., Wozny M., Fang M., (Boston: Springer), pp. 847-852.

APPENDICES

The appendices present support information for the thesis.

- Appendix I provides the OWL code for the ontology-based dependency metamodel and the ontology-based dependency model (OBDM)
- Appendix II provides the data associated with the experiments carried out in this thesis.
- Appendix III provides a simple worked example of the inputs and outputs for the TomE tool.
- Appendix IV provides the overview of the directory structure for the code for HotFusion and TomE tools that is supplied on DVD with this thesis.

APPENDIX I

This appendix contains the OWL code for the ontology-based dependency metamodel and ontology-based dependency model that was created during this research.

Ontology-Based Dependency Metamodel

```
<?xml version="1.0"?>


<!DOCTYPE rdf:RDF [
    <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
    <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
    <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
    <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
]>


<rdf:RDF xmlns="http://www.owl-ontologies.com/Ontology1270901584.owl#"
     xml:base="http://www.owl-ontologies.com/Ontology1270901584.owl"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:owl="http://www.w3.org/2002/07/owl#">
    <owl:Ontology rdf:about=""/>
    <owl:Class rdf:ID="ArchitecturalEntities"/>
    <owl:Class rdf:ID="Cause">
        <rdfs:subClassOf
rdf:resource="#DescriptiveDependencyAtrributes"/>
    </owl:Class>
    <owl:DatatypeProperty rdf:ID="cause_dst">
        <rdf:type rdf:resource="&owl;FunctionalProperty"/>
        <rdfs:domain rdf:resource="#Cause"/>
        <rdfs:range rdf:resource="&xsd;string"/>
    </owl:DatatypeProperty>
    <owl:DatatypeProperty rdf:ID="cause_src">
        <rdf:type rdf:resource="&owl;FunctionalProperty"/>
        <rdfs:domain rdf:resource="#Cause"/>
        <rdfs:range rdf:resource="&xsd;string"/>
    </owl:DatatypeProperty>
    <owl:DatatypeProperty rdf:ID="cause_value">
        <rdf:type rdf:resource="&owl;FunctionalProperty"/>
        <rdfs:domain rdf:resource="#Cause"/>
        <rdfs:range rdf:resource="&xsd;string"/>
    </owl:DatatypeProperty>
    <owl:Class rdf:ID="DependencyGraph"/>
    <owl:ObjectProperty rdf:ID="DependencyRelation"/>
    <owl:Class rdf:ID="DescriptiveDependencyAtrributes"/>
    <owl:DatatypeProperty rdf:ID="domainname">
        <rdf:type rdf:resource="&owl;FunctionalProperty"/>
        <rdfs:range rdf:resource="&xsd;string"/>
    </owl:DatatypeProperty>
    <owl:ObjectProperty rdf:ID="functional_dependency_relation">
        <rdf:type rdf:resource="&owl;FunctionalProperty"/>
        <rdfs:subPropertyOf rdf:resource="#DependencyRelation"/>
    </owl:ObjectProperty>
    <owl:DatatypeProperty rdf:ID="graphname">
```

```
        <rdf:type rdf:resource="&owl;FunctionalProperty"/>
        <rdfs:domain rdf:resource="#DependencyGraph"/>
        <rdfs:range rdf:resource="&xsd;string"/>
    </owl:DatatypeProperty>
    <owl:DatatypeProperty rdf:ID="graphtype">
        <rdf:type rdf:resource="&owl;FunctionalProperty"/>
        <rdfs:domain rdf:resource="#DependencyGraph"/>
        <rdfs:range rdf:resource="&xsd;string"/>
    </owl:DatatypeProperty>
    <owl:ObjectProperty rdf:ID="hascauseattribute">
        <rdfs:domain rdf:resource="#ArchitecturalEntities"/>
        <rdfs:range rdf:resource="#Cause"/>
    </owl:ObjectProperty>
    <owl:ObjectProperty rdf:ID="hasimpactattribute">
        <rdfs:domain rdf:resource="#ArchitecturalEntities"/>
        <rdfs:range rdf:resource="#Impact"/>
    </owl:ObjectProperty>
    <owl:ObjectProperty rdf:ID="hasstrenghtattribute">
        <rdfs:domain rdf:resource="#ArchitecturalEntities"/>
        <rdfs:range rdf:resource="#Strength"/>
    </owl:ObjectProperty>
    <owl:Class rdf:ID="Impact">
        <rdfs:subClassOf
rdf:resource="#DescriptiveDependencyAtrributes"/>
    </owl:Class>
    <owl:DatatypeProperty rdf:ID="impact_dst">
        <rdf:type rdf:resource="&owl;FunctionalProperty"/>
        <rdfs:domain rdf:resource="#Impact"/>
        <rdfs:range rdf:resource="&xsd;string"/>
    </owl:DatatypeProperty>
    <owl:DatatypeProperty rdf:ID="impact_src">
        <rdf:type rdf:resource="&owl;FunctionalProperty"/>
        <rdfs:domain rdf:resource="#Impact"/>
        <rdfs:range rdf:resource="&xsd;string"/>
    </owl:DatatypeProperty>
    <owl:DatatypeProperty rdf:ID="impact_value">
        <rdf:type rdf:resource="&owl;FunctionalProperty"/>
        <rdfs:domain rdf:resource="#Impact"/>
        <rdfs:range rdf:resource="&xsd;string"/>
    </owl:DatatypeProperty>
    <owl:ObjectProperty rdf:ID="inverse_function_dependency_relation">
        <rdf:type rdf:resource="&owl;InverseFunctionalProperty"/>
        <rdfs:subPropertyOf rdf:resource="#DependencyRelation"/>
    </owl:ObjectProperty>
    <owl:DatatypeProperty rdf:ID="lvl_dst"/>
    <owl:DatatypeProperty rdf:ID="lvl_level"/>
    <owl:DatatypeProperty rdf:ID="lvl_src"/>
    <owl:Class rdf:ID="Strength">
        <rdfs:subClassOf
rdf:resource="#DescriptiveDependencyAtrributes"/>
    </owl:Class>
    <owl:DatatypeProperty rdf:ID="strength_dst">
        <rdf:type rdf:resource="&owl;FunctionalProperty"/>
        <rdfs:domain rdf:resource="#Strength"/>
        <rdfs:range rdf:resource="&xsd;string"/>
    </owl:DatatypeProperty>
    <owl:DatatypeProperty rdf:ID="strength_src">
        <rdf:type rdf:resource="&owl;FunctionalProperty"/>
        <rdfs:domain rdf:resource="#Strength"/>
        <rdfs:range rdf:resource="&xsd;string"/>
    </owl:DatatypeProperty>
```

```
    <owl:DatatypeProperty rdf:ID="strength_value">
        <rdf:type rdf:resource="&owl;FunctionalProperty"/>
        <rdfs:domain rdf:resource="#Strength"/>
        <rdfs:range rdf:resource="&xsd;int"/>
    </owl:DatatypeProperty>
    <owl:ObjectProperty rdf:ID="symmetric_dependency_relation">
        <rdf:type rdf:resource="&owl;SymmetricProperty"/>
        <owl:inverseOf rdf:resource="#symmetric_dependency_relation"/>
        <rdfs:subPropertyOf rdf:resource="#DependencyRelation"/>
    </owl:ObjectProperty>
    <owl:ObjectProperty rdf:ID="transitive_dependency_relation">
        <rdf:type rdf:resource="&owl;TransitiveProperty"/>
        <rdfs:subPropertyOf rdf:resource="#DependencyRelation"/>
    </owl:ObjectProperty>
    <owl:ObjectProperty
rdf:ID="transitive_symmetric_dependency_relation">
        <rdf:type rdf:resource="&owl;SymmetricProperty"/>
        <rdf:type rdf:resource="&owl;TransitiveProperty"/>
        <owl:inverseOf
rdf:resource="#transitive_symmetric_dependency_relation"/>
        <rdfs:subPropertyOf rdf:resource="#DependencyRelation"/>
    </owl:ObjectProperty>
    <owl:DatatypeProperty rdf:ID="type">
        <rdf:type rdf:resource="&owl;FunctionalProperty"/>
        <rdfs:domain rdf:resource="#ArchitecturalEntities"/>
        <rdfs:range rdf:resource="&xsd;string"/>
    </owl:DatatypeProperty>
    <owl:DatatypeProperty rdf:ID="version">
        <rdf:type rdf:resource="&owl;FunctionalProperty"/>
        <rdfs:domain rdf:resource="#DependencyGraph"/>
        <rdfs:range rdf:resource="&xsd;string"/>
    </owl:DatatypeProperty>
</rdf:RDF>
```

Ontology-Based Dependency Model (OBDM)

```
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [
    <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
    <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
    <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
    <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
    <!ENTITY protégé
"http://protege.stanford.edu/plugins/owl/protege#" >
 <!ENTITY p1 "http://www.owl-ontologies.com/Ontology1270901584.owl#" >
]>

<rdf:RDF xmlns="http://www.owl-ontologies.com/Ontology1275558355.owl#"
     xml:base="http://www.owl-ontologies.com/Ontology1275558355.owl"
     xmlns:p1="http://www.owl-ontologies.com/Ontology1270901584.owl#"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:protege="http://protege.stanford.edu/plugins/owl/protege#"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:owl="http://www.w3.org/2002/07/owl#">
    <owl:Ontology rdf:about="">
        <owl:imports rdf:resource="http://www.owl-
ontologies.com/Ontology1270901584.owl"/>
    </owl:Ontology>
    <owl:ObjectProperty rdf:ID="executes">
        <rdfs:domain rdf:resource="#MP"/>
        <rdfs:range rdf:resource="#FN"/>
        <rdfs:subPropertyOf rdf:resource="&p1;DependencyRelation"/>
    </owl:ObjectProperty>
    <owl:Class rdf:ID="FN">
        <rdfs:subClassOf rdf:resource="&p1;ArchitecturalEntities"/>
    </owl:Class>
    <owl:Class rdf:ID="GE">
        <rdfs:subClassOf rdf:resource="&p1;ArchitecturalEntities"/>
    </owl:Class>
    <owl:ObjectProperty rdf:ID="hasinputparams">
        <rdfs:domain rdf:resource="#FN"/>
        <rdfs:range rdf:resource="#IP"/>
        <rdfs:subPropertyOf rdf:resource="&p1;DependencyRelation"/>
    </owl:ObjectProperty>
    <owl:ObjectProperty rdf:ID="haslocalparams">
        <rdfs:domain rdf:resource="#FN"/>
        <rdfs:range rdf:resource="#LP"/>
        <rdfs:subPropertyOf rdf:resource="&p1;DependencyRelation"/>
    </owl:ObjectProperty>
    <owl:ObjectProperty rdf:ID="hasoutputparams">
        <rdfs:domain rdf:resource="#FN"/>
        <rdfs:range rdf:resource="#OP"/>
        <rdfs:subPropertyOf rdf:resource="&p1;DependencyRelation"/>
    </owl:ObjectProperty>
    <owl:ObjectProperty
rdf:ID="inverse_of_transitive_symmetric_dependency_relation_11">
        <rdfs:domain rdf:resource="#MP"/>
        <rdfs:range rdf:resource="#UE"/>
        <owl:inverseOf rdf:resource="#ue2mp"/>
        <rdfs:subPropertyOf
rdf:resource="&p1;transitive_symmetric_dependency_relation"/>
    </owl:ObjectProperty>
    <owl:ObjectProperty
rdf:ID="inverse_of_transitive_symmetric_dependency_relation_13">
```

```
        <rdfs:domain rdf:resource="#LE"/>
        <rdfs:range rdf:resource="#MP"/>
        <owl:inverseOf rdf:resource="#mp2le"/>
        <rdfs:subPropertyOf
rdf:resource="&p1;transitive_symmetric_dependency_relation"/>
    </owl:ObjectProperty>
    <owl:ObjectProperty
rdf:ID="inverse_of_transitive_symmetric_dependency_relation_14">
        <rdfs:domain rdf:resource="#GE"/>
        <rdfs:range rdf:resource="#LE"/>
        <owl:inverseOf rdf:resource="#le2ge"/>
        <rdfs:subPropertyOf
rdf:resource="&p1;transitive_symmetric_dependency_relation"/>
    </owl:ObjectProperty>
    <owl:Class rdf:ID="IP">
        <rdfs:subClassOf rdf:resource="&p1;ArchitecturalEntities"/>
    </owl:Class>
    <owl:Class rdf:ID="LE">
        <rdfs:subClassOf rdf:resource="&p1;ArchitecturalEntities"/>
    </owl:Class>
    <owl:ObjectProperty rdf:ID="le2ge">
        <rdfs:domain rdf:resource="#LE"/>
        <rdfs:range rdf:resource="#GE"/>
        <owl:inverseOf
rdf:resource="#inverse_of_transitive_symmetric_dependency_relation_14"
/>
        <rdfs:subPropertyOf
rdf:resource="&p1;transitive_symmetric_dependency_relation"/>
    </owl:ObjectProperty>
    <owl:Class rdf:ID="LP">
        <rdfs:subClassOf rdf:resource="&p1;ArchitecturalEntities"/>
    </owl:Class>
    <owl:Class rdf:ID="MP">
        <rdfs:subClassOf rdf:resource="&p1;ArchitecturalEntities"/>
    </owl:Class>
    <owl:ObjectProperty rdf:ID="mp2le">
        <rdfs:domain rdf:resource="#MP"/>
        <rdfs:range rdf:resource="#LE"/>
        <owl:inverseOf
rdf:resource="#inverse_of_transitive_symmetric_dependency_relation_13"
/>
        <rdfs:subPropertyOf
rdf:resource="&p1;transitive_symmetric_dependency_relation"/>
    </owl:ObjectProperty>
    <owl:Class rdf:ID="OP">
        <rdfs:subClassOf rdf:resource="&p1;ArchitecturalEntities"/>
    </owl:Class>
    <rdf:Description rdf:about="&p1;ArchitecturalEntities"/>
    <owl:Class rdf:ID="UE">
        <rdfs:subClassOf rdf:resource="&p1;ArchitecturalEntities"/>
    </owl:Class>
    <owl:ObjectProperty rdf:ID="ue2mp">
        <rdfs:domain rdf:resource="#UE"/>
        <rdfs:range rdf:resource="#MP"/>
        <owl:inverseOf
rdf:resource="#inverse_of_transitive_symmetric_dependency_relation_11"
/>
        <rdfs:subPropertyOf
rdf:resource="&p1;transitive_symmetric_dependency_relation"/>
    </owl:ObjectProperty>
</rdf:RDF>
```

APPENDIX II

This appendix contains the data associated with each of the experiments in this research.

Experimental Data for Experiment One

Upper Ontology

```
<?xml version="1.0"?>
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns="http://www.owl-ontologies.com/Ontology1172143263.owl#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
  xml:base="http://www.owl-ontologies.com/Ontology1172143263.owl">
  <owl:Ontology rdf:about=""/>
  <owl:Class rdf:ID="sales_item">
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
          <owl:ObjectProperty rdf:ID="hasforecastitems2"/>
        </owl:onProperty>
        <owl:someValuesFrom>
          <owl:Class rdf:ID="forecasted_item"/>
        </owl:someValuesFrom>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:someValuesFrom>
          <owl:Class rdf:ID="sales_rev"/>
        </owl:someValuesFrom>
        <owl:onProperty>
          <owl:ObjectProperty rdf:ID="haveSalesRev"/>
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:someValuesFrom>
          <owl:Class rdf:about="#forecasted_item"/>
        </owl:someValuesFrom>
        <owl:onProperty>
          <owl:ObjectProperty rdf:ID="hasforecastitems"/>
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Class rdf:ID="products"/>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
          <owl:DatatypeProperty rdf:ID="products_sales_names"/>
        </owl:onProperty>
        <owl:someValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
```

```
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
          <owl:DatatypeProperty rdf:ID="products_sales_id"/>
        </owl:onProperty>
        <owl:someValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
      </owl:Restriction>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="revenue"/>
  <owl:Class rdf:ID="opportunity">
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
          <owl:TransitiveProperty rdf:ID="hasProducts"/>
        </owl:onProperty>
        <owl:someValuesFrom>
          <owl:Class rdf:about="#products"/>
        </owl:someValuesFrom>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:someValuesFrom>
          <owl:Class rdf:ID="customers"/>
        </owl:someValuesFrom>
        <owl:onProperty>
          <owl:TransitiveProperty rdf:ID="hasCustomer"/>
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
          <owl:ObjectProperty rdf:about="#haveSalesRev"/>
        </owl:onProperty>
        <owl:someValuesFrom>
          <owl:Class rdf:about="#sales_rev"/>
        </owl:someValuesFrom>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:someValuesFrom>
          <owl:Class rdf:ID="forecast_rev"/>
        </owl:someValuesFrom>
        <owl:onProperty>
          <owl:ObjectProperty rdf:ID="haveForecastRev"/>
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
          <owl:DatatypeProperty rdf:ID="opportunity_id"/>
        </owl:onProperty>
        <owl:someValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
```

```
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
          <owl:DatatypeProperty rdf:ID="opportunity_name"/>
        </owl:onProperty>
        <owl:someValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf
rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
  </owl:Class>
  <owl:Class rdf:about="#customers">
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:someValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
        <owl:onProperty>
          <owl:DatatypeProperty rdf:ID="customers_region"/>
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:someValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
        <owl:onProperty>
          <owl:DatatypeProperty rdf:ID="customers_name"/>
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
          <owl:DatatypeProperty rdf:ID="customers_id"/>
        </owl:onProperty>
        <owl:someValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf
rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:someValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
        <owl:onProperty>
          <owl:DatatypeProperty rdf:ID="customers_accountexec"/>
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
          <owl:DatatypeProperty rdf:ID="customers_tier1support"/>
        </owl:onProperty>
        <owl:someValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
      </owl:Restriction>
```

```
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
          <owl:DatatypeProperty rdf:ID="customers_tier2support"/>
        </owl:onProperty>
        <owl:someValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
      </owl:Restriction>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:about="#products">
    <rdfs:subClassOf
rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:someValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
        <owl:onProperty>
          <owl:DatatypeProperty rdf:ID="products_type"/>
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:about="#sales_rev">
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
          <owl:DatatypeProperty rdf:ID="sales_rev_sales_q4_rev"/>
        </owl:onProperty>
        <owl:someValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:someValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
        <owl:onProperty>
          <owl:DatatypeProperty rdf:ID="sales_rev_sales_q3_rev"/>
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:someValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
        <owl:onProperty>
          <owl:DatatypeProperty rdf:ID="sales_rev_sales_q2_rev"/>
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
          <owl:DatatypeProperty rdf:ID="sales_rev_sales_q1_rev"/>
        </owl:onProperty>
        <owl:someValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
      </owl:Restriction>
    </rdfs:subClassOf>
```

```
    <rdfs:subClassOf rdf:resource="#revenue"/>
  </owl:Class>
  <owl:Class rdf:about="#forecast_rev">
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:someValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
        <owl:onProperty>
          <owl:DatatypeProperty rdf:ID="forecast_rev_q4_rev"/>
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
          <owl:DatatypeProperty rdf:ID="forecast_rev_q3_rev"/>
        </owl:onProperty>
        <owl:someValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
          <owl:DatatypeProperty rdf:ID="forecast_rev_q2_rev"/>
        </owl:onProperty>
        <owl:someValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
          <owl:DatatypeProperty rdf:ID="forecast_rev_q1_rev"/>
        </owl:onProperty>
        <owl:someValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf rdf:resource="#revenue"/>
  </owl:Class>
  <owl:Class rdf:about="#forecasted_item">
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:someValuesFrom rdf:resource="#forecast_rev"/>
        <owl:onProperty>
          <owl:ObjectProperty rdf:about="#haveForecastRev"/>
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:someValuesFrom rdf:resource="#sales_item"/>
        <owl:onProperty>
          <owl:ObjectProperty rdf:ID="hasparentsalesitem"/>
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf rdf:resource="#products"/>
    <rdfs:subClassOf>
      <owl:Restriction>
```

```
        <owl:someValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
        <owl:onProperty>
          <owl:DatatypeProperty rdf:ID="products_fi_name"/>
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
          <owl:DatatypeProperty rdf:ID="products_fi_id"/>
        </owl:onProperty>
        <owl:someValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
      </owl:Restriction>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:ObjectProperty rdf:about="#hasforecastitems">
    <rdfs:range rdf:resource="#forecasted_item"/>
    <rdfs:domain rdf:resource="#sales_item"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="#hasparentsalesitem">
    <rdfs:domain rdf:resource="#forecasted_item"/>
    <rdfs:range rdf:resource="#sales_item"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="#hasforecastitems2">
    <rdfs:domain rdf:resource="#sales_item"/>
    <rdfs:range>
      <owl:Restriction>
        <owl:onProperty rdf:resource="#products_fi_id"/>
        <owl:someValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
      </owl:Restriction>
    </rdfs:range>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="#haveSalesRev">
    <rdfs:domain>
      <owl:Class>
        <owl:unionOf rdf:parseType="Collection">
          <owl:Class rdf:about="#opportunity"/>
          <owl:Class rdf:about="#sales_item"/>
        </owl:unionOf>
      </owl:Class>
    </rdfs:domain>
    <rdfs:range rdf:resource="#sales_rev"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="#haveForecastRev">
    <rdfs:domain>
      <owl:Class>
        <owl:unionOf rdf:parseType="Collection">
          <owl:Class rdf:about="#opportunity"/>
          <owl:Class rdf:about="#forecasted_item"/>
        </owl:unionOf>
      </owl:Class>
    </rdfs:domain>
    <rdfs:range rdf:resource="#forecast_rev"/>
  </owl:ObjectProperty>
  <owl:DatatypeProperty rdf:about="#opportunity_id">
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:about="#customers_tier2support">
```

```
    <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    <rdfs:domain rdf:resource="#customers"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:about="#customers_accountexec">
    <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    <rdfs:domain rdf:resource="#customers"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:about="#customers_region">
    <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:about="#customers_name">
    <rdfs:domain rdf:resource="#customers"/>
    <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:about="#products_sales_names">
    <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:about="#opportunity_name">
    <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:about="#products_type">
    <rdfs:range>
      <owl:DataRange>
        <owl:oneOf rdf:parseType="Resource">
          <rdf:first
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
          >SALES</rdf:first>
          <rdf:rest rdf:parseType="Resource">
            <rdf:first
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
            >FORECAST</rdf:first>
            <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-
syntax-ns#nil"/>
          </rdf:rest>
        </owl:oneOf>
      </owl:DataRange>
    </rdfs:range>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:about="#products_sales_id">
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:about="#customers_id">
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:about="#customers_tier1support">
    <rdfs:domain rdf:resource="#customers"/>
    <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:DatatypeProperty>
  <owl:TransitiveProperty rdf:about="#hasCustomer">
    <rdfs:range rdf:resource="#customers"/>
    <rdf:type
rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
    <rdfs:domain rdf:resource="#opportunity"/>
  </owl:TransitiveProperty>
```

```
  <owl:TransitiveProperty rdf:about="#hasProducts">
    <rdf:type
rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
    <rdfs:range rdf:resource="#products"/>
    <rdfs:domain rdf:resource="#opportunity"/>
  </owl:TransitiveProperty>
  <customers rdf:ID="test">
    <customers_id rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >0</customers_id>
    <customers_region
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >emea</customers_region>
    <customers_name
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    ></customers_name>
  </customers>
  <sales_rev rdf:ID="sales_rev_8">
    <sales_rev_sales_q3_rev
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    ></sales_rev_sales_q3_rev>
    <sales_rev_sales_q4_rev
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >4</sales_rev_sales_q4_rev>
    <sales_rev_sales_q2_rev
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    ></sales_rev_sales_q2_rev>
    <sales_rev_sales_q1_rev
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    ></sales_rev_sales_q1_rev>
  </sales_rev>
  <forecast_rev rdf:ID="forecast_rev_10">
    <forecast_rev_q2_rev
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    ></forecast_rev_q2_rev>
    <forecast_rev_q1_rev
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    ></forecast_rev_q1_rev>
    <forecast_rev_q3_rev
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    ></forecast_rev_q3_rev>
    <forecast_rev_q4_rev
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >4</forecast_rev_q4_rev>
  </forecast_rev>
  <sales_item rdf:ID="sales_item_7">
    <haveSalesRev rdf:resource="#sales_rev_8"/>
    <products_sales_id
rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >0</products_sales_id>
    <products_sales_names
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    ></products_sales_names>
    <hasforecastitems>
      <forecasted_item rdf:ID="forecasted_item_9">
        <products_type
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >FORECAST</products_type>
        <products_fi_name
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >prod1name</products_fi_name>
        <hasparentsalesitem rdf:resource="#sales_item_7"/>
```

```
        <products_fi_id
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        ></products_fi_id>
        <haveForecastRev rdf:resource="#forecast_rev_10"/>
      </forecasted_item>
    </hasforecastitems>
    <products_type
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >SALES</products_type>
  </sales_item>
  <opportunity rdf:ID="opp2">
    <hasCustomer rdf:resource="#test"/>
    <opportunity_id
rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >0</opportunity_id>
    <hasProducts rdf:resource="#sales_item_7"/>
    <opportunity_name
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    ></opportunity_name>
    <haveSalesRev rdf:resource="#sales_rev_8"/>
    <haveForecastRev rdf:resource="#forecast_rev_10"/>
  </opportunity>
</rdf:RDF>

<!-- Created with Protege (with OWL Plugin 3.2, Build 355)
http://protege.stanford.edu -->
```

Mapping file

```
<?xml version="1.0" encoding="UTF-8"?>
<Mappings>
 <mapping>
        <mapping_number>3</mapping_number>
        <source_type>p</source_type>
        <source_name>customers_id</source_name>
        <source_expansion>s</source_expansion>
        <dest_db>db2</dest_db>
        <dest_type>p</dest_type>
        <dest_prop_name>id</dest_prop_name>
        <dest_table_name>custs</dest_table_name>
        <dest_pkey>id</dest_pkey>
        <dest_pkey_type>int</dest_pkey_type>
        <source_expansion_db>null</source_expansion_db>
        <source_expansion_class>null</source_expansion_class>
  </mapping>
 <mapping>
        <mapping_number>4</mapping_number>
        <source_type>p</source_type>
        <source_name>customers_id</source_name>
        <source_expansion>s</source_expansion>
        <dest_db>db1</dest_db>
        <dest_type>p</dest_type>
        <dest_prop_name>id</dest_prop_name>
        <dest_table_name>customers</dest_table_name>
        <dest_pkey>id</dest_pkey>
        <dest_pkey_type>int</dest_pkey_type>
        <source_expansion_db>null</source_expansion_db>
        <source_expansion_class>null</source_expansion_class>
  </mapping>

 <mapping>
        <mapping_number>5</mapping_number>
        <source_type>p</source_type>
        <source_name>customers_id</source_name>
        <source_expansion>p</source_expansion>
        <dest_db>db1</dest_db>
        <dest_type>p</dest_type>
        <dest_prop_name>customer_id</dest_prop_name>
        <dest_table_name>opps</dest_table_name>
        <dest_pkey>oppid</dest_pkey>
        <dest_pkey_type>String</dest_pkey_type>
        <source_expansion_db>db1</source_expansion_db>
        <source_expansion_class>customers</source_expansion_class>
  </mapping>

 <mapping>
         <mapping_number>6</mapping_number>
        <source_type>p</source_type>
        <source_name>customers_name</source_name>
        <source_expansion>s</source_expansion>
        <dest_db>db2</dest_db>
        <dest_type>p</dest_type>
        <dest_prop_name>name</dest_prop_name>
        <dest_table_name>custs</dest_table_name>
        <dest_pkey>id</dest_pkey>
        <dest_pkey_type>int</dest_pkey_type>
    <source_expansion_db>null</source_expansion_db>
```

```
    <source_expansion_class>null</source_expansion_class>
  </mapping>

 <mapping>
        <mapping_number>7</mapping_number>
        <source_type>p</source_type>
        <source_name>customers_name</source_name>
        <source_expansion>s</source_expansion>
        <dest_db>db1</dest_db>
        <dest_type>p</dest_type>
        <dest_prop_name>name</dest_prop_name>
        <dest_table_name>customers</dest_table_name>
        <dest_pkey>id</dest_pkey>
        <dest_pkey_type>int</dest_pkey_type>
        <source_expansion_db>null</source_expansion_db>
        <source_expansion_class>null</source_expansion_class>
  </mapping>

 <mapping>
        <mapping_number>8</mapping_number>
        <source_type>p</source_type>
        <source_name>customers_region</source_name>
        <source_expansion>s</source_expansion>
        <dest_db>db2</dest_db>
        <dest_type>p</dest_type>
        <dest_prop_name>region</dest_prop_name>
        <dest_table_name>custs</dest_table_name>
        <dest_pkey>id</dest_pkey>
<dest_pkey_type>int</dest_pkey_type>
        <source_expansion_db>null</source_expansion_db>
        <source_expansion_class>null</source_expansion_class>
  </mapping>

 <mapping>
         <mapping_number>9</mapping_number>
        <source_type>p</source_type>
        <source_name>customers_region</source_name>
        <source_expansion>s</source_expansion>
        <dest_db>db1</dest_db>
        <dest_type>p</dest_type>
        <dest_prop_name>region</dest_prop_name>
        <dest_table_name>customers</dest_table_name>
        <dest_pkey>id</dest_pkey>
        <dest_pkey_type>int</dest_pkey_type>
        <source_expansion_db>null</source_expansion_db>
        <source_expansion_class>null</source_expansion_class>
  </mapping>

  <mapping>
         <mapping_number>9.1</mapping_number>
        <source_type>p</source_type>
        <source_name>customers_accountexec</source_name>
        <source_expansion>s</source_expansion>
        <dest_db>db2</dest_db>
        <dest_type>p</dest_type>
        <dest_prop_name>accountexec</dest_prop_name>
        <dest_table_name>custs</dest_table_name>
        <dest_pkey>id</dest_pkey>
        <dest_pkey_type>int</dest_pkey_type>
        <source_expansion_db>null</source_expansion_db>
        <source_expansion_class>null</source_expansion_class>
```

```
 </mapping>

 <mapping>
        <mapping_number>9.2</mapping_number>
       <source_type>p</source_type>
       <source_name>customers_tier1support</source_name>
       <source_expansion>s</source_expansion>
       <dest_db>db2</dest_db>
       <dest_type>p</dest_type>
       <dest_prop_name>tier1support</dest_prop_name>
       <dest_table_name>custs</dest_table_name>
       <dest_pkey>id</dest_pkey>
       <dest_pkey_type>int</dest_pkey_type>
       <source_expansion_db>null</source_expansion_db>
       <source_expansion_class>null</source_expansion_class>
 </mapping>

 <mapping>
        <mapping_number>9.3</mapping_number>
       <source_type>p</source_type>
       <source_name>customers_tier2support</source_name>
       <source_expansion>s</source_expansion>
       <dest_db>db2</dest_db>
       <dest_type>p</dest_type>
       <dest_prop_name>tier2support</dest_prop_name>
       <dest_table_name>custs</dest_table_name>
       <dest_pkey>id</dest_pkey>
       <dest_pkey_type>int</dest_pkey_type>
       <source_expansion_db>null</source_expansion_db>
       <source_expansion_class>null</source_expansion_class>
 </mapping>

<mapping>
        <mapping_number>12</mapping_number>
       <source_type>p</source_type>
       <source_name>products_fi_name</source_name>
       <source_expansion>s</source_expansion>
       <dest_db>db2</dest_db>
       <dest_type>p</dest_type>
       <dest_prop_name>longcode</dest_prop_name>
       <dest_table_name>prods</dest_table_name>
       <dest_pkey>id</dest_pkey>
       <dest_pkey_type>int</dest_pkey_type>
       <source_expansion_db>null</source_expansion_db>
       <source_expansion_class>null</source_expansion_class>
 </mapping>

<mapping>
        <mapping_number>13</mapping_number>
       <source_type>p</source_type>
       <source_name>products_sales_name</source_name>
       <source_expansion>s</source_expansion>
       <dest_db>db1</dest_db>
       <dest_type>p</dest_type>
       <dest_prop_name>name</dest_prop_name>
       <dest_table_name>products</dest_table_name>
       <dest_pkey>id</dest_pkey>
       <dest_pkey_type>int</dest_pkey_type>
       <source_expansion_db>null</source_expansion_db>
       <source_expansion_class>null</source_expansion_class>
 </mapping>
```

```
<mapping>
        <mapping_number>14</mapping_number>
       <source_type>p</source_type>
       <source_name>products_fi_id</source_name>
       <source_expansion>p</source_expansion>
       <dest_db>db2</dest_db>
       <dest_type>p</dest_type>
       <dest_prop_name>prodid</dest_prop_name>
       <dest_table_name>forecasted_items</dest_table_name>
       <dest_pkey>prodid</dest_pkey>
       <dest_pkey_type>int</dest_pkey_type>
       <source_expansion_db>db2</source_expansion_db>
       <source_expansion_class>prods</source_expansion_class>
 </mapping>



<mapping>
        <mapping_number>15</mapping_number>
       <source_type>p</source_type>
       <source_name>products_sales_id</source_name>
       <source_expansion>p</source_expansion>
       <dest_db>db1</dest_db>
       <dest_type>p</dest_type>
       <dest_prop_name>id</dest_prop_name>
       <dest_table_name>products</dest_table_name>
       <dest_pkey>id</dest_pkey>
       <dest_pkey_type>int</dest_pkey_type>
       <source_expansion_db>null</source_expansion_db>
       <source_expansion_class>null</source_expansion_class>
 </mapping>


<mapping>
       <mapping_number>16</mapping_number>
       <source_type>p</source_type>
       <source_name>sales_rev_sales_q1_rev</source_name>
       <source_expansion>p</source_expansion>
       <dest_db>db1</dest_db>
       <dest_type>p</dest_type>
       <dest_prop_name>revq1</dest_prop_name>
       <dest_table_name>opps</dest_table_name>
       <dest_pkey>oppid</dest_pkey>
       <dest_pkey_type>string</dest_pkey_type>
       <source_expansion_db>null</source_expansion_db>
       <source_expansion_class>null</source_expansion_class>
 </mapping>

<mapping>
        <mapping_number>17</mapping_number>
       <source_type>p</source_type>
       <source_name>sales_rev_sales_q2_rev</source_name>
       <source_expansion>p</source_expansion>
       <dest_db>db1</dest_db>
       <dest_type>p</dest_type>
       <dest_prop_name>revq2</dest_prop_name>
       <dest_table_name>opps</dest_table_name>
       <dest_pkey>oppid</dest_pkey>
       <dest_pkey_type>string</dest_pkey_type>
       <source_expansion_db>null</source_expansion_db>
```

```
        <source_expansion_class>null</source_expansion_class>
  </mapping>

<mapping>
        <mapping_number>18</mapping_number>
        <source_type>p</source_type>
        <source_name>sales_rev_sales_q3_rev</source_name>
        <source_expansion>p</source_expansion>
        <dest_db>db1</dest_db>
        <dest_type>p</dest_type>
        <dest_prop_name>revq3</dest_prop_name>
        <dest_table_name>opps</dest_table_name>
        <dest_pkey>oppid</dest_pkey>
        <dest_pkey_type>string</dest_pkey_type>
        <source_expansion_db>null</source_expansion_db>
        <source_expansion_class>null</source_expansion_class>
  </mapping>

<mapping>
         <mapping_number>19</mapping_number>
        <source_type>p</source_type>
        <source_name>sales_rev_sales_q4_rev</source_name>
        <source_expansion>p</source_expansion>
        <dest_db>db1</dest_db>
        <dest_type>p</dest_type>
        <dest_prop_name>revq4</dest_prop_name>
        <dest_table_name>opps</dest_table_name>
        <dest_pkey>oppid</dest_pkey>
        <dest_pkey_type>string</dest_pkey_type>
        <source_expansion_db>null</source_expansion_db>
        <source_expansion_class>null</source_expansion_class>
  </mapping>

<mapping>
        <mapping_number>20</mapping_number>
        <source_type>p</source_type>
        <source_name>forecast_rev_q1_rev</source_name>
        <source_expansion>p</source_expansion>
        <dest_db>db2</dest_db>
        <dest_type>p</dest_type>
        <dest_prop_name>revm1</dest_prop_name>
        <dest_table_name>forecasted_items</dest_table_name>
        <dest_pkey>opp</dest_pkey>
        <dest_pkey_type>string</dest_pkey_type>
        <source_expansion_db>null</source_expansion_db>
        <source_expansion_class>null</source_expansion_class>
  </mapping>

<mapping>
         <mapping_number>21</mapping_number>
        <source_type>p</source_type>
        <source_name>forecast_rev_q2_rev</source_name>
        <source_expansion>p</source_expansion>
        <dest_db>db2</dest_db>
        <dest_type>p</dest_type>
        <dest_prop_name>revm4</dest_prop_name>
        <dest_pkey>opp</dest_pkey>
        <dest_pkey_type>string</dest_pkey_type>
        <dest_table_name>forecasted_items</dest_table_name>
        <source_expansion_db>null</source_expansion_db>
        <source_expansion_class>null</source_expansion_class>
```

```
  </mapping>

<mapping>
        <mapping_number>22</mapping_number>
        <source_type>p</source_type>
        <source_name>forecast_rev_q3_rev</source_name>
        <source_expansion>p</source_expansion>
        <dest_db>db2</dest_db>
        <dest_type>p</dest_type>
        <dest_prop_name>revm7</dest_prop_name>
        <dest_table_name>forecasted_items</dest_table_name>
        <dest_pkey>opp</dest_pkey>
        <dest_pkey_type>string</dest_pkey_type>
        <source_expansion_db>null</source_expansion_db>
        <source_expansion_class>null</source_expansion_class>
  </mapping>

<mapping>
        <mapping_number>23</mapping_number>
        <source_type>p</source_type>
        <source_name>forecast_rev_q4_rev</source_name>
        <source_expansion>p</source_expansion>
        <dest_db>db2</dest_db>
        <dest_type>p</dest_type>
        <dest_prop_name>revm10</dest_prop_name>
        <dest_table_name>forecasted_items</dest_table_name>
        <dest_pkey>opp</dest_pkey>
        <dest_pkey_type>string</dest_pkey_type>
        <source_expansion_db>null</source_expansion_db>
        <source_expansion_class>null</source_expansion_class>
  </mapping>


<mapping>
        <mapping_number>24</mapping_number>
        <source_type>p</source_type>
        <source_name>opportunity_name</source_name>
        <source_expansion>p</source_expansion>
        <dest_db>db1</dest_db>
        <dest_type>p</dest_type>
        <dest_prop_name>oppname</dest_prop_name>
        <dest_table_name>opps</dest_table_name>
        <dest_pkey>oppid</dest_pkey>
        <dest_pkey_type>string</dest_pkey_type>
        <source_expansion_db>null</source_expansion_db>
        <source_expansion_class>null</source_expansion_class>
  </mapping>

<mapping>
        <mapping_number>25</mapping_number>
        <source_type>p</source_type>
        <source_name>product_fi_id</source_name>
        <source_expansion>p</source_expansion>
        <dest_db>db2</dest_db>
        <dest_type>p</dest_type>
        <dest_prop_name>prodid</dest_prop_name>
        <dest_table_name>forecasted_items</dest_table_name>
        <dest_pkey>prodcat</dest_pkey>
        <dest_pkey_type>null</dest_pkey_type>
        <source_expansion_db>null</source_expansion_db>
        <source_expansion_class>null</source_expansion_class>
```

```
  </mapping>

<mapping>
         <mapping_number>26</mapping_number>
        <source_type>link</source_type>
        <source_name>db1,customers,id</source_name>
        <source_expansion>p</source_expansion>
        <dest_db>db1</dest_db>
        <dest_type>p</dest_type>
        <dest_prop_name>oppid</dest_prop_name>
        <dest_table_name>opps</dest_table_name>
        <dest_pkey>oppid</dest_pkey>
        <dest_pkey_type>int</dest_pkey_type>
        <source_expansion_db>null</source_expansion_db>
<source_expansion_class>customer_id</source_expansion_class>
  </mapping>
<mapping>
         <mapping_number>27</mapping_number>
        <source_type>link</source_type>
        <source_name>db1,customers,id</source_name>
<source_expansion>p</source_expansion>
        <dest_db>db1</dest_db>
        <dest_type>p</dest_type>
        <dest_prop_name>oppname</dest_prop_name>
        <dest_table_name>opps</dest_table_name>
        <dest_pkey>null</dest_pkey>
        <dest_pkey_type>null</dest_pkey_type>
        <source_expansion_db>null</source_expansion_db>
        <source_expansion_class>null</source_expansion_class>
  </mapping>

<mapping>
        <mapping_number>28</mapping_number>
        <source_type>link</source_type>
        <source_name>db1,customers,id</source_name>
        <source_expansion>p</source_expansion>
        <dest_db>db1</dest_db>
        <dest_type>p</dest_type>
        <dest_prop_name>productid</dest_prop_name>
        <dest_table_name>opps</dest_table_name>
        <dest_pkey>null</dest_pkey>
        <dest_pkey_type>null</dest_pkey_type>
        <source_expansion_db>null</source_expansion_db>
        <source_expansion_class>null</source_expansion_class>
  </mapping>

<mapping>
        <mapping_number>29</mapping_number>
        <source_type>link</source_type>
        <source_name>db1,products,id</source_name>
        <source_expansion>p</source_expansion>
        <dest_db>db1</dest_db>
        <dest_type>p</dest_type>
        <dest_prop_name>oppid</dest_prop_name>
        <dest_table_name>opps</dest_table_name>
        <dest_pkey>oppid</dest_pkey>
        <dest_pkey_type>null</dest_pkey_type>
        <source_expansion_db>null</source_expansion_db>
     <source_expansion_class>productid</source_expansion_class>
  </mapping>
```

```
<mapping>
         <mapping_number>30</mapping_number>
        <source_type>link</source_type>
        <source_name>db2,forecasted_items,prodcat</source_name>
        <source_expansion>p</source_expansion>
        <dest_db>db1</dest_db>
        <dest_type>p</dest_type>
        <dest_prop_name>id</dest_prop_name>
        <dest_table_name>products</dest_table_name>
        <dest_pkey>id</dest_pkey>
        <dest_pkey_type>null</dest_pkey_type>
        <source_expansion_db>null</source_expansion_db>
      <source_expansion_class>productid</source_expansion_class>
  </mapping>

<mapping>
        <mapping_number>31</mapping_number>
        <source_type>link</source_type>
        <source_name>db1,opps,oppid</source_name>
        <source_expansion>p</source_expansion>
        <dest_db>db1</dest_db>
        <dest_type>p</dest_type>
        <dest_prop_name>oppid</dest_prop_name>
        <dest_table_name>opps</dest_table_name>
        <dest_pkey>oppid</dest_pkey>
        <dest_pkey_type>null</dest_pkey_type>
        <source_expansion_db>null</source_expansion_db>
        <source_expansion_class>oppid</source_expansion_class>
  </mapping>

    <mapping>
         <mapping_number>32</mapping_number>
        <source_type>link</source_type>
        <source_name>db2,forecasted_items,revm1</source_name>
        <source_expansion>p</source_expansion>
        <dest_db>db2</dest_db>
        <dest_type>p</dest_type>
        <dest_prop_name>opp</dest_prop_name>
        <dest_table_name>forecasted_items</dest_table_name>
        <dest_pkey>opp</dest_pkey>
        <dest_pkey_type>null</dest_pkey_type>
        <source_expansion_db>null</source_expansion_db>
        <source_expansion_class>opp</source_expansion_class>
  </mapping>
</Mappings>
```

Experimental Data for Experiment Two

Upper Ontology for Experiment two (Logistics).

```
<?xml version="1.0"?>
<rdf:RDF
    xmlns:xsp="http://www.owl-ontologies.com/2005/08/07/xsp.owl#"
    xmlns:swrlb="http://www.w3.org/2003/11/swrlb#"
    xmlns:swrl="http://www.w3.org/2003/11/swrl#"
    xmlns:protege="http://protege.stanford.edu/plugins/owl/protege#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns="http://www.owl-ontologies.com/Ontology1225382715.owl#"
    xmlns:assert="http://www.owl-ontologies.com/assert.owl#"
  xml:base="http://www.owl-ontologies.com/Ontology1225382715.owl">
  <owl:Ontology rdf:about="">
    <owl:imports rdf:resource="http://www.owl-
ontologies.com/assert.owl"/>
  </owl:Ontology>
  <owl:Class rdf:ID="duties"/>
  <owl:Class rdf:ID="exportduties">
    <rdfs:subClassOf rdf:resource="#duties"/>
  </owl:Class>
  <owl:Class rdf:ID="lot"/>
  <owl:Class rdf:ID="shipmentinformation"/>
  <owl:Class rdf:ID="services"/>
  <owl:Class rdf:ID="ratesheets">
    <rdfs:subClassOf
rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
          <owl:ObjectProperty rdf:ID="haveratecosts"/>
        </owl:onProperty>
        <owl:someValuesFrom>
          <owl:Class rdf:ID="weightcosts"/>
        </owl:someValuesFrom>
      </owl:Restriction>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="surcharges"/>
  <owl:Class rdf:ID="packages"/>
  <owl:Class rdf:ID="importduties">
    <rdfs:subClassOf rdf:resource="#duties"/>
  </owl:Class>
  <owl:Class rdf:ID="carriers"/>
  <owl:Class rdf:ID="zone">
    <owl:disjointWith>
      <owl:Class rdf:ID="origin"/>
    </owl:disjointWith>
  </owl:Class>
  <owl:Class rdf:ID="lane"/>
  <owl:Class rdf:ID="irc">
    <rdfs:subClassOf rdf:resource="#surcharges"/>
  </owl:Class>
  <owl:Class rdf:ID="route">
    <owl:equivalentClass>
```

```
    <owl:Class>
      <owl:intersectionOf rdf:parseType="Collection">
        <owl:Restriction>
          <owl:onProperty>
            <owl:ObjectProperty rdf:ID="containlots"/>
          </owl:onProperty>
          <owl:someValuesFrom rdf:resource="#lot"/>
        </owl:Restriction>
        <owl:Restriction>
          <owl:allValuesFrom rdf:resource="#lot"/>
          <owl:onProperty>
            <owl:ObjectProperty rdf:about="#containlots"/>
          </owl:onProperty>
        </owl:Restriction>
      </owl:intersectionOf>
    </owl:Class>
  </owl:equivalentClass>
</owl:Class>
<owl:Class rdf:ID="fuel">
  <rdfs:subClassOf rdf:resource="#surcharges"/>
</owl:Class>
<owl:Class rdf:about="#origin">
  <owl:disjointWith rdf:resource="#zone"/>
</owl:Class>
<owl:Class rdf:ID="destination"/>
<owl:Class rdf:ID="weight_types"/>
<owl:ObjectProperty rdf:ID="hasZone">
  <rdfs:range rdf:resource="#zone"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="usespackages">
  <rdfs:range rdf:resource="#packages"/>
  <rdfs:domain rdf:resource="#services"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="havesurcharges">
  <rdfs:range rdf:resource="#surcharges"/>
  <rdfs:domain rdf:resource="#services"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="startAt">
  <rdfs:domain rdf:resource="#route"/>
  <rdfs:range rdf:resource="#origin"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="sell">
  <rdfs:domain rdf:resource="#carriers"/>
  <rdfs:range rdf:resource="#services"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="endAt">
  <rdfs:domain rdf:resource="#route"/>
  <rdfs:range>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class rdf:about="#destination"/>
        <owl:Class rdf:about="#zone"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:range>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="haveratecostsin">
  <rdfs:domain rdf:resource="#services"/>
  <rdfs:range rdf:resource="#ratesheets"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasCountry"/>
```

```
  <owl:ObjectProperty rdf:ID="operate">
    <rdfs:range rdf:resource="#route"/>
    <rdfs:domain rdf:resource="#carriers"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="#haveratecosts">
    <rdfs:domain rdf:resource="#ratesheets"/>
    <rdfs:range rdf:resource="#weightcosts"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="hasAirport"/>
  <owl:ObjectProperty rdf:ID="containslanes">
    <rdfs:range rdf:resource="#lane"/>
    <rdfs:domain rdf:resource="#route"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="haveduties">
    <rdfs:domain rdf:resource="#services"/>
    <rdfs:range rdf:resource="#duties"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="logsshipmentinfo">
    <rdfs:range rdf:resource="#shipmentinformation"/>
    <rdfs:domain rdf:resource="#services"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="#containlots">
    <rdfs:domain rdf:resource="#lane"/>
    <rdfs:range rdf:resource="#lot"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="hasCity"/>
  <owl:DatatypeProperty rdf:ID="ratestructure">
    <rdfs:domain rdf:resource="#carriers"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="ed_othercharges">
    <rdfs:domain rdf:resource="#exportduties"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="lotid">
    <rdfs:domain rdf:resource="#lot"/>
    <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="p_materialdescription">
    <rdfs:domain rdf:resource="#packages"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="irc_pkg">
    <rdfs:domain rdf:resource="#irc"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="irc_type">
    <rdfs:domain rdf:resource="#irc"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="id_ta_min">
    <rdfs:domain rdf:resource="#importduties"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="laneid">
    <rdfs:domain rdf:resource="#lane"/>
    <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="isocode">
    <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="ed_handlingperkg">
    <rdfs:domain rdf:resource="#exportduties"/>
  </owl:DatatypeProperty>
```

```
  <owl:DatatypeProperty rdf:ID="d_countryname">
    <rdfs:domain rdf:resource="#destination"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="averageweight">
    <rdfs:domain rdf:resource="#shipmentinformation"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="ws_value"/>
  <owl:DatatypeProperty rdf:ID="lang">
    <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="lanedescription">
    <rdfs:domain rdf:resource="#lane"/>
    <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="p_packtype">
    <rdfs:domain rdf:resource="#packages"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="airportcode"/>
  <owl:DatatypeProperty rdf:ID="id_ic5_pos">
    <rdfs:domain rdf:resource="#importduties"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="ed_securityperkg">
    <rdfs:domain rdf:resource="#exportduties"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="range_identifier"/>
  <owl:DatatypeProperty rdf:ID="fuel_min">
    <rdfs:domain rdf:resource="#fuel"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="service_name">
    <rdfs:domain rdf:resource="#ratesheets"/>
    <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="weighttype">
    <rdfs:domain rdf:resource="#weightcosts"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="endweight">
    <rdfs:domain rdf:resource="#weightcosts"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="transit_time_max">
    <rdfs:domain rdf:resource="#services"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="ed_tlcurrency">
    <rdfs:domain rdf:resource="#exportduties"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="weightunit">
    <rdfs:domain rdf:resource="#weightcosts"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="multiplier">
    <rdfs:domain rdf:resource="#weightcosts"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="symbol">
    <rdfs:domain rdf:resource="#weight_types"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="id_leadtime">
    <rdfs:domain rdf:resource="#importduties"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="d_zone"/>
  <owl:DatatypeProperty rdf:ID="fsc_type">
```

```
    <rdfs:domain rdf:resource="#fuel"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="startweight">
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
    <rdfs:domain rdf:resource="#weightcosts"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="servicename">
    <rdfs:domain rdf:resource="#services"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="lotname">
    <rdfs:domain rdf:resource="#lot"/>
    <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="transit_time_min">
    <rdfs:domain rdf:resource="#services"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="id_ta_perkg">
    <rdfs:domain rdf:resource="#importduties"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="lotdescription">
    <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    <rdfs:domain rdf:resource="#lot"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="ed_tlairport">
    <rdfs:domain rdf:resource="#exportduties"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="d_ctylang">
    <rdfs:domain rdf:resource="#destination"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="ed_TI_max">
    <rdfs:domain rdf:resource="#exportduties"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="df_thurs">
    <rdfs:domain rdf:resource="#services"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="id_T1fee">
    <rdfs:domain rdf:resource="#importduties"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="occode">
    <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="servicesummary">
    <rdfs:domain rdf:resource="#carriers"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="ed_exportcustdoc">
    <rdfs:domain rdf:resource="#exportduties"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="region">
    <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="df_sun">
    <rdfs:domain rdf:resource="#services"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="ws_end"/>
  <owl:DatatypeProperty rdf:ID="id_hc_max">
    <rdfs:domain rdf:resource="#importduties"/>
  </owl:DatatypeProperty>
```

```
  <owl:DatatypeProperty rdf:ID="ed_leadtime">
    <rdfs:domain rdf:resource="#exportduties"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="point_identifier">
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="ed_handlingmax">
    <rdfs:domain rdf:resource="#exportduties"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="id_icaddpos">
    <rdfs:domain rdf:resource="#importduties"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="yearlyshipments">
    <rdfs:domain rdf:resource="#shipmentinformation"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="ed_TI_min">
    <rdfs:domain rdf:resource="#exportduties"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="ed_securitymax">
    <rdfs:domain rdf:resource="#exportduties"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="rs_type"/>
  <owl:DatatypeProperty rdf:ID="fuel_max">
    <rdfs:domain rdf:resource="#fuel"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="consolidation_airport">
    <rdfs:domain rdf:resource="#services"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="o_countryname">
    <rdfs:domain rdf:resource="#origin"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="df_wed">
    <rdfs:domain rdf:resource="#services"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="irc_max">
    <rdfs:domain rdf:resource="#irc"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="id_atlasfee">
    <rdfs:domain rdf:resource="#importduties"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="packtype">
    <rdfs:domain rdf:resource="#services"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="cost">
    <rdfs:domain rdf:resource="#weightcosts"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="DatatypeProperty_3"/>
  <owl:DatatypeProperty rdf:ID="id_storageperaddday">
    <rdfs:domain rdf:resource="#importduties"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="o_cityname">
    <rdfs:domain rdf:resource="#origin"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="currency">
    <rdfs:domain rdf:resource="#ratesheets"/>
    <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="d_airportcode">
    <rdfs:domain rdf:resource="#destination"/>
  </owl:DatatypeProperty>
```

```
  <owl:DatatypeProperty rdf:ID="zoneid">
    <rdfs:domain rdf:resource="#zone"/>
    <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="routename">
    <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    <rdfs:domain rdf:resource="#route"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="o_ctycode">
    <rdfs:domain rdf:resource="#origin"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="id_hc_perkg">
    <rdfs:domain rdf:resource="#importduties"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="yearlyweight">
    <rdfs:domain rdf:resource="#shipmentinformation"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="d_ctyisocode">
    <rdfs:domain rdf:resource="#destination"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="df_sat">
    <rdfs:domain rdf:resource="#services"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="costset">
    <rdfs:domain rdf:resource="#ratesheets"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="irc_min">
    <rdfs:domain rdf:resource="#irc"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="ed_TI_perkg">
    <rdfs:domain rdf:resource="#exportduties"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="o_ctyisocode">
    <rdfs:domain rdf:resource="#origin"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="transit_time">
    <rdfs:domain rdf:resource="#services"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="unit">
    <rdfs:domain rdf:resource="#weight_types"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="id_ta_max">
    <rdfs:domain rdf:resource="#importduties"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="ed_securitymin">
    <rdfs:domain rdf:resource="#exportduties"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="name">
    <rdfs:domain rdf:resource="#carriers"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="rs_start"/>
  <owl:DatatypeProperty rdf:ID="o_ctylang">
    <rdfs:domain rdf:resource="#origin"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="airportname"/>
  <owl:DatatypeProperty rdf:ID="id_hc_min">
    <rdfs:domain rdf:resource="#importduties"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="transfer_airport">
```

```
    <rdfs:domain rdf:resource="#services"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="direct_flight">
    <rdfs:domain rdf:resource="#services"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="o_airportcode">
    <rdfs:domain rdf:resource="#origin"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="lanename">
    <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    <rdfs:domain rdf:resource="#lane"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="fuel_pkg">
    <rdfs:domain rdf:resource="#fuel"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="weight">
    <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    <rdfs:domain rdf:resource="#weightcosts"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="ed_handlingmin">
    <rdfs:domain rdf:resource="#exportduties"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="df_tues">
    <rdfs:domain rdf:resource="#services"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="value">
    <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="d_ctycode">
    <rdfs:domain rdf:resource="#destination"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="df_fri">
    <rdfs:domain rdf:resource="#services"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="countryname"/>
  <owl:DatatypeProperty rdf:ID="p_dims">
    <rdfs:domain rdf:resource="#packages"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="id_othercharges">
    <rdfs:domain rdf:resource="#importduties"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="cityname"/>
  <owl:DatatypeProperty rdf:ID="d_cityname">
    <rdfs:domain rdf:resource="#destination"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="id_handoverfee">
    <rdfs:domain rdf:resource="#importduties"/>
  </owl:DatatypeProperty>
</rdf:RDF>

<!-- Created with Protege (with OWL Plugin 3.2, Build 355)
http://protege.stanford.edu -->
```

Mapping file for experiment two (Excerpt from full mapping on DVD)

```
<?xml version="1.0" encoding="UTF-8"?>
<Mappings>
<mapping>
<mapping_number>c1</mapping_number>
<mapping_type>ps</mapping_type>
<source_type>p</source_type>
<source_name>carriers:name</source_name>
<source_expansion>null</source_expansion>
<dest_db>exp2_test:exp_test_db2</dest_db>
<dest_type>p</dest_type>
<dest_prop_name>Awards:Awards</dest_prop_name>
<dest_table_name>logistics:logistics</dest_table_name>
<dest_pkey>null</dest_pkey>
<dest_pkey_type>null</dest_pkey_type>
<source_expansion_db>null</source_expansion_db>
<source_expansion_class>null</source_expansion_class>
<function>c1:null</function>
</mapping>

<mapping>
<mapping_number>s1</mapping_number>
<mapping_type>ps</mapping_type>
<mapping_desc>Need complex func to extract service names from these
dest fields</mapping_desc>
<source_type>p</source_type>
<source_name>services:servicename</source_name>
<source_expansion>null</source_expansion>
<dest_db>exp2_test:exp2_test:exp2_test:exp2_test_db2</dest_db>
<dest_type>p</dest_type>
<dest_prop_name>ALDS_1_44:BLDS_1_44:UDS_1_44:Service</dest_prop_name>
<dest_table_name>rates:rates:rates:rates</dest_table_name>
<dest_pkey>null</dest_pkey>
<dest_pkey_type>null</dest_pkey_type>
<source_expansion_db>null</source_expansion_db>
<source_expansion_class>null</source_expansion_class>
<function>s1:S1_1:s1_2:null</function>
</mapping>

<mapping>
<mapping_number>s3</mapping_number>
<mapping_type>pp</mapping_type>
<mapping_desc>simple point to point </mapping_desc>
<source_type>p</source_type>
<source_name>services:packtype</source_name>
<source_expansion>null</source_expansion>
<dest_db>exp2_test_db2</dest_db>
<dest_type>p</dest_type>
<dest_prop_name>PackType</dest_prop_name>
<dest_table_name>logistics</dest_table_name>
<dest_pkey>null</dest_pkey>
<dest_pkey_type>null</dest_pkey_type>
<source_expansion_db>null</source_expansion_db>
<source_expansion_class>null</source_expansion_class>
<function>null</function>
</mapping>

<mapping>
<mapping_number>s4</mapping_number>
```

```
<mapping_type>pp</mapping_type>
<mapping_desc>simple point to point </mapping_desc>
<source_type>p</source_type>
<source_name>services:commodity</source_name>
<source_expansion>null</source_expansion>
<dest_db>exp2_test_db2</dest_db>
<dest_type>p</dest_type>
<dest_prop_name>commodity</dest_prop_name>
<dest_table_name>rates</dest_table_name>
<dest_pkey>null</dest_pkey>
<dest_pkey_type>null</dest_pkey_type>
<source_expansion_db>null</source_expansion_db>
<source_expansion_class>null</source_expansion_class>
<function>null</function>
</mapping>

<mapping>
<mapping_number>s5</mapping_number>
<mapping_type>ps</mapping_type>
<mapping_desc>simple point to point </mapping_desc>
<source_type>p</source_type>
<source_name>services:transittime</source_name>
<source_expansion>null</source_expansion>
<dest_db>exp2_test_db2:exp2_test:exp2_test:exp2_test</dest_db>
<dest_type>p</dest_type>
<dest_prop_name>transit_time:alds_transit_time:blds_transit_time:uds_t
ransmit_time</dest_prop_name>
<dest_table_name>servicedescriptions:servicedescriptions:servicedescri
ptions:servicedescriptions</dest_table_name>
<dest_pkey>null</dest_pkey>
<dest_pkey_type>null</dest_pkey_type>
<source_expansion_db>null</source_expansion_db>
<source_expansion_class>null</source_expansion_class>
<function>s5:s5:s5:s5</function>
</mapping>

<mapping>
<mapping_number>s6</mapping_number>
<mapping_type>ps</mapping_type>
<mapping_desc>simple point to point </mapping_desc>
<source_type>p</source_type>
<source_name>services:transittimemax</source_name>
<source_expansion>null</source_expansion>
<dest_db>exp2_test_db2:exp2_test:exp2_test:exp2_test</dest_db>
<dest_type>p</dest_type>
<dest_prop_name>transit_time_max:alds_transit_time_max:blds_transit_ti
me_max:uds_transmit_time_max</dest_prop_name>
<dest_table_name>servicedescriptions:servicedescriptions:servicedescri
ptions:servicedescriptions</dest_table_name>
<dest_pkey>null</dest_pkey>
<dest_pkey_type>null</dest_pkey_type>
<source_expansion_db>null</source_expansion_db>
<source_expansion_class>null</source_expansion_class>
<function>s6:s6:s6:s6</function>
</mapping>
.
.
.
</Mappings>
```

Experimental Data for Experiment Three

The runtime performance and accuracy of the OBDM was verified in experiment three. To demonstrate the difficulty of mapping evolution without tool support, experiment three measured the performance and accuracy of a manual approach to dependency analysis and compared this to the OBDM.

Manual Process Definition

Evaluation of Mapping Dependency Discovery

Overview: This evaluation has been setup to measure the performance of a new technique to discover the dependencies that arise in data integration systems. A dependency is a simple relation between two things that are dependent (e.g. A depends on B)

The exercises in this evaluation are based on three samples of data which are provided in three spreadsheets. Each spreadsheet contains columns of data which represents some internal aspects of the integration system. From these spreadsheets, a view of the items that are dependent on each other can be built up (using the process described later). In general, each row in any given spreadsheet represents elements that depend on each other. If two different rows share the same element then the elements in each row can be dependent also. The process below provides a failsafe way to find out the dependencies that exist.

There are 4 questions to be answered on each spreadsheet and must be completed in 20 minutes giving a total time of 1 hour for 12 questions. Each question simply requires the user to run the process below.

In the spreadsheets, the integration system has been divided up into parts as follows:
UPPER ENTITY (UE)
MAPPING (MP)
LOWER ENTITY (LE)
FUNCTION (F)

Each spreadsheet has 6 columns.

COLUMN A = Is the name of the UPPER ENTITY

COLUMN B = Is the name of the MAPPING (MP)

COLUMN C, D, E = Is the name of the LOWER ENTITIES (LE). NB. These columns are colon separated lists of Lower Entities

COLUMN F - Is the name of another UPPER ENTITY that this MAPPING uses.

A row in the excel spreadsheet defines what the UPPER ENTITY on that row depends on.

Process:

For each question below please carry out the following process.

1. **OPEN THE APPROPRIATE DATASET (SMALL, MEDIUM OR LARGE).**

2. **NOTE THE START TIME**

3. **FIND THE ROW WHERE THE UE OR LE SPECIFIED IN THE QUESTION OCCURS AND WRITE DOWN THE MAPPING POINT NAME**

4. **CHECK IF ANY OF THE LE(S) FROM THE ROWS NOTED IN STEP 3 OCCUR IN OTHER ROWS (COLUMN C,D,E). WHERE MATCHES ARE FOUND NOTE DOWN THE MAPPING POINT NAME OF THAT ROW**

5. **FOR EACH MP NAME FOUND SO FAR, CHECK IF A FUNCTION IS SPECIFIED (COLUMN F).**
    a) **IF A FUNCTION IS SPECIFIED THEN FIND THE ROW WHERE THAT FUNCTION NAME APPEARS AS A UE (COLUMN A) OR A FUNCTION (COLUMN F) AND NOTE DOWN THE MP NAME**
    b) **REPEAT STEP 4 FOR ANY ROWS FOUND**

6. **NOTE THE END TIME**
7. **WRITE DOWN DURATION IN THE "TIME TAKEN TO COMPLETE (IN SECONDS)" FIELD**

User Questionnaire.

Evaluation of Mapping Dependency Discovery

NAME:

……………………………………………………………………………………………

A) DATA SET 1 – first.xls

1. **Find DEPENDENTS OF: (LE) "db1:rates:ALDS_500_999"**

START TIME [ ]
END TIME [ ]
DEPENDENT ELEMENTS:

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

2. Find DEPENDENTS OF (LE) "db1:fcs_irc:fcs_min"

START TIME [ ]
END TIME [ ]

DEPENDENT ELEMENTS:
……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

3. Find DEPENDENTS OF (LE) “db2:rates:commodity”

START TIME [ ]
END TIME [ ]

DEPENDENT ELEMENTS:
……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

4. Find DEPENDENTS OF (LE) "ls1:sdescs:uds_transit_time_min"

START TIME [ ]
END TIME [ ]

DEPENDENT ELEMENTS:

B) DATA SET 2 – SECOND.XLS

5. Find DEPENDENTS OF (LE) ds2:logisticdescriptions:PackType

START TIME [ ]
END TIME [ ]

DEPENDENT ELEMENTS:
……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

6. Find DEPENDENTS OF (LE) ds1:sdescs:uds_transit_time_min

START TIME [ ]
END TIME [ ]

DEPENDENT ELEMENTS:
……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

7. Find DEPENDENTS OF (LE) ds2:sdescs:transit_time

START TIME [ ]
END TIME [ ]

DEPENDENT ELEMENTS:
……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

8. Find DEPENDENTS OF (LE) ls1:servdescriptions:uds_transit_time_min

START TIME [ ]
END TIME [ ]

DEPENDENT ELEMENTS:
……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

C) DATA SET 3 –THIRD.XLS

9. Find DEPENDENTS OF (LE) “db1:rates:ALDS_1_44”

START TIME [ ]
END TIME [ ]

DEPENDENT ELEMENTS:

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

10. Find DEPENDENTS OF (LE) "db1:fcs_irc:irc_pkg"

START TIME [ ]
END TIME [ ]
DEPENDENT ELEMENTS:
……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

11. Find DEPENDENTS OF (LE) “db1:lotdescriptions:LotNumber”

START TIME [ ]
END TIME [ ]

DEPENDENT ELEMENTS:
……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

12. Find DEPENDENTS OF (LE) "ls1:sdescs:uds_transit_time_min"

START TIME [ ]
END TIME [ ]

DEPENDENT ELEMENTS:

A.) **Please add further observations about the difficulty of the task**

- Did you find the task ?

Easy	Hard	Very Hard	Impossible
[ ]	[ ]	[ ]	[ ]

- Which part of the process was the hardest

STEP3	STEP4	STEP5
[ ]	[ ]	[ ]

- Rate the easiest and hardest dataset by writing "HARD" and "EASY" in the selection below.

ONE	TWO	THREE
[ ]	[ ]	[ ]

- Rate the easiest and hardest question

Hardest QUESTION NO [ ]
Easiest QUESTION NO [ ]

- How confident are you that your answers are correct (i.e. you have no errors)

Not Confident	Confident	Very Confident
[ ]	[ ]	[ ]

- Any other comments

……
……
……
……

……………………………………………………………………………………………
……

……………………………………………………………………………………………
……

……………………………………………………………………………………………
……

……………………………………………………………………………………………
……

……………………………………………………………………………………………
……

……………………………………………………………………………………………
……

……………………………………………………………………………………………
……

……………………………………………………………………………………………
……

……………………………………………………………………………………………
……

……………………………………………………………………………………………
……

……………………………………………………………………………………………
……

……………………………………………………………………………………………
……

……………………………………………………………………………………………
……

……………………………………………………………………………………………
……

The results of this survey will be used for the PhD project of Aidan Boran only and will be made available for the participant on request.

First Mapping File (MS-Excel Format)

	A	B	C	D	E	F	G	H	I	J
1	**UPPER ENTITY**	**MAPPING**	**LOWER ENTITY ADDRESS 1**	**LOWER ENTITY ADDRESS 2**	**LOWER ENTITY ADDRESS 3**	**FUNCTION**				
2	carriers:name	c1	db1:db2	logistics:logistics	Awards:Awards					
3	services:servicename	s1	db1:db1:db1:db2	rates:rates:rates:rates	ALDS_1_44:BLDS_1_44:UDS_1_44:Service					
4	services:packtype	s3	db2	logistics	PackType					
5	services:commodity	s4	db2	rates	commodity	commodity:name				
6	services:transittime	s5	db2:db1:db1:db1:db1	servicedescriptions:servicedes	transit_time:alds_transit_time:blds_transit_time:uds_transit_time:code					
7	services:transittimemax	s6	db2:db1:db1:db1	servicedescriptions:servicedes	transit_time_max:alds_transit_time_max:blds_transit_time_max:uds_transit_time_max					
8	services:transittimemin	s7	db2:db1:db1:db1	servicedescriptions:servicedes	transit_time_min:alds_transit_time_min:blds_transit_time_min:uds_transit_time_min					
9	services:df_mon	s11	db1:db1:db1	servicedescriptions:servicedes	alds_df_mon:blds_df_mon:uds_df_mon					
10	services:df_tues	s12	db1:db1:db1	servicedescriptions:servicedes	alds_df_tues:blds_df_tues:uds_df_tues					
11	services:df_wed	s13	db1:db1:db1	servicedescriptions:servicedes	alds_df_wed:blds_df_wed:uds_df_wed					
12	services:df_thurs	s14	db1:db1:db1	servicedescriptions:servicedes	alds_df_thu:blds_df_thu:uds_df_thu					
13	services:df_fri	s15	db1:db1:db1	servicedescriptions:servicedes	alds_df_fri:blds_df_fri:uds_df_fri					
14	services:df_sat	s16	db1:db1:db1	servicedescriptions:servicedes	alds_df_sat:blds_df_sat:uds_df_sat					
15	services:df_sun	s17	db1:db1:db1	servicedescriptions:servicedes	alds_df_sun:blds_df_sun:uds_df_sun					
16	services:direct_flight	s18	db1:db1:db1	servicedescriptions:servicedes	alds_direct_flight:blds_direct_flight:uds_direct_flight					
17	services:consolidation_airport	s19	db1:db1:db1	servicedescriptions:servicedes	alds_consolidation_airport:blds_consolidation_airport:uds_consolidation_airport					
18	services:transfer_airport	s20	db1:db1:db1	servicedescriptions:servicedes	alds_transit_airport:blds_transit_airport:uds_transit_airport					
19	ratesheets:service_name	sn1	db1:db1:db1:db2	rates:rates:rates:rates	ALDS_1_44:BLDS_1_44:UDS_1_44:Service					
20	weightcosts:weighttype	m1	db2:db1:db1:db1:db1:db1	rates:rates:rates:rates:rates:rates	weight:ALDS_1_44:ALDS_45_99:ALDS_100_249:ALDS_250_499:ALDS_500_999:ALDS_1000_2999:ALDS_	weight_types:unit				
21	weightcosts:startweight	m2	db2:db1:db1:db1:db1:db1:db1	rates:rates:rates:rates:rates:rates	weight:ALDS_1_44:ALDS_45					
22	weightcosts:endweight	m3	db2:db1:db1:db1:db1:db1:db1	rates:rates:rates:rates:rates:rates	weight:ALDS_1_44:ALDS_45	weight_types:unit				
23	weightcosts:multiplier	m4	db2:db1:db1:db1:db1:db1:db1	rates:rates:rates:rates:rates:rates	weight:ALDS_1_44:ALDS_45	weight_types:unit				
24	weightcosts:cost	m5	db2:db1:db1:db1:db1:db1:db1	rates:rates:rates:rates:rates:rates	costzone:ALDS_1_44:ALDS_45	ratesheets:currency currency:symbol				
25	weightcosts:weight	m6	db2:db1:db1:db1:db1:db1:db1	rates:rates:rates:rates:rates:rates	weight:ALDS_1_44:ALDS_45	weight_types:unit				
26	ratesheets:currency	m7	db2:db1	logistics:logistics	currencyId:currencyId					
27	currency:symbol	m8	db2:db1	currency:currency	symbol:symbol					
28	weight_types:unit	m9	db2:db1	units:weight_types	unit:weight_symbol					
29	zone:zoneid	pp1	db2	zonetable	DestinationZone					
30	fuel:fuel_min	pp2	db1	fcs_irc	fcs_min	fuel:fsc_type				
31	fuel:fuel_max	pp3	db1	fcs_irc	fcs_max	fuel:fsc_type				
32	fuel:fuel_pkg	pp4	db1	fcs_irc	fcs_pkg					

Second Mapping File (MS-Excel Format)

	A	B	C	D	E	F
1	**UPPER ENTITY**	**MAPPING**	**LOWER ENTITY ADDRESS 1**	**LOWER ENTITY ADDRESS 2**	**LOWER ENTITY ADDRESS 3**	**FUNCTION**
2	importduties:id_ta_max	app42	ds1	inland_import	TA_max	
3	fuel:fsc_type	aipp38	ds1	fcs_irc	fsc_type	
4	irc:irc_type	aipp39	ds1	fcs_irc	irc_type	
5	carriers:ratestructure	am201	ds2:ds1	logisticdescriptions:logisticde	RateStructure:RateStructure	
6	packages:p_packtype	as300	ds2:ds2	logisticdescriptions:packages	PackType:id	
7	packages:p_materialdescriptic	as301	ds2	rates	commodity	
8	carriers:servicesummary	as500	ds1:ds1:ds1:ds1:ds1:ds1:ds2	logisticdescriptions:sdescs:sc	RateStructure:alds_transit_time:blds_transit_time:uds_transit_time:ALDS_1_44:BLDS_1_44:UDS_1_44:Se	
9	route:routename	as600	ds1:ds1	lotdescriptions:lanedescriptior	LotNumber:LaneNumber	
10	commodity:name	as700	ds1	commodity	name	commodity:id
11	commodity:id	as701	ds1	commodity	id	
12	ms_services:transittimemin	msas7	ls2:ls1:ls1:ls1	servdescriptions:servdescriptic	transit_time_min:alds_transit_time_min:blds_transit_time_min:uds_transit	ms_carriers:ratestructure
13	ms_carriers:ratestructure	msam201	ls2:ls1	logisticdescriptions:logisticde	RateStructure:RateStructure	
14	ms_carriers:servicesummary	msas500	ls1:ls1:ls1:ls1:ls1:ls1:ls1:ls2:ls2:l	logisticdescriptions:servdescri	RateStructure:alds_transit_time:blds_transit_time:uds_transit_time:ALDS_1_44:BLDS_1_44:UDS_1_44:Se	
15	ms_ratesheets:service_name	msasn1	ls1:ls1:ls1:ls1:ls2	rates:rates:rates:rates	ALDS_1_44:BLDS_1_44:UDS_1_44:Service	
16	ms_services:servicename	msas1	ls1:ls1:ls1:ls2	rates:rates:rates:rates	ALDS_1_44:BLDS_1_44:UDS_1_44:Service	
17	ms_services:transittime	msas5	ls2:ls1:ls1:ls1:ls1	servdescriptions:servdescriptic	transit_time:alds_transit_time:blds_transit_time:uds_transit_time:code	ms_carriers:ratestructure
18	ms_services:transittimemax	msas6	ls2:ls1:ls1:ls1	servdescriptions:servdescriptic	transit_time_max:alds_transit_time_max:blds_transit_time_max:uds_tran	ms_carriers:ratestructure
19	ms_services:a_localcode	ms9000	ls2:ls1	lr_codes:lr_codes	a_lcode:a_lcode	ms_services:a_remotecode
20	ms_services:a_remotecode	ms9001	ls2:ls1	lr_codes:lr_codes	a_rcode:a_rcode	ms_services:b_localcode
21	ms_services:b_localcode	ms9002	ls2:ls1	lr_codes:lr_codes	b_lcode:b_lcode	ms_services:b_remotecode
22	ms_services:b_remotecode	ms9003	ls2:ls1	lr_codes:lr_codes	b_rcode:b_rcode	ms_services:c_localcode
23	ms_services:c_localcode	ms9004	ls2:ls1	lr_codes:lr_codes	c_lcode:c_lcode	
24	carriers:name	ac1	ds1:ds2	logisticdescriptions:logisticde	Awards:Awards	
25	services:servicename	as1	ds1:ds1:ds1:ds2	rates:rates:rates:rates	ALDS_1_44:BLDS_1_44:UDS_1_44:Service	
26	services:packtype	as3	ds2	logisticdescriptions	PackType	
27	services:commodity	as4	ds2	rates	commodity	commodity:name
28	services:transittime	as5	ds2:ds1:ds1:ds1:ds1	sdescs:sdescs:sdescs:sdesc	transit_time:alds_transit_time:blds_transit_time:uds_transit_time:code	carriers:ratestructure
29	services:transittimemax	as6	ds2:ds1:ds1:ds1	sdescs:sdescs:sdescs:sdesc	transit_time_max:alds_transit_time_max:blds_transit_time_max:uds_tran	carriers:ratestructure
30	services:transittimemin	as7	ds2:ds1:ds1:ds1	sdescs:sdescs:sdescs:sdesc	transit_time_min:alds_transit_time_min:blds_transit_time_min:uds_transit	carriers:ratestructure
31	services:consolidation_airport	as19	ds1:ds1:ds1	sdescs:sdescs:sdescs	alds_consolidation_airport:blds_consolidation_airport:uds_consolidation_airport	
32	services:transfer_airport	as20	ds1:ds1:ds1	sdescs:sdescs:sdescs	alds_transit_airport:blds_transit_airport:uds_transit_airport	
33	ratesheets:service_name	asn1	ds1:ds1:ds1:ds2	rates:rates:rates:rates	ALDS_1_44:BLDS_1_44:UDS_1_44:Service	
34	origin:o_airportcode	app20	ds1:ds2	logisticdescriptions:logisticde	OriginAirportCode:OriginAirportCode	
35	origin:o_countryname	app21	ds1:ds2	logisticdescriptions:logisticde	OriginCountryName:OriginCountryName	
36	origin:o_ctyisocode	app22	ds1	countrycodes	isocode	
37	origin:o_ctycode	app23	ds1	countrycodes	occcode	
38	origin:o_lang	app24	ds1	countrycodes	lang	
39	destination:d_cityname	app25	ds1:ds2	logisticdescriptions:logisticde	DestCityName:DestCityName	
40	destination:d_airportcode	app26	ds1:ds2	logisticdescriptions:logisticde	DestAirportCode:DestAirportCode	
41	destination:d_countryname	app27	ds1:ds2	logisticdescriptions:logisticde	DestCountryName:DestCountryName	
42	destination:d_ctyisocode	app28	ds1	countrycodes	isocode	

Third Mapping File (MS-Excel Format)

	A	B	C	D	E	F	G
1	**UPPER ENTITY**	**MAPPING**	**LOWER ENTITY ADDF**	**LOWER ENTITY ADDRESS 2**	**LOWER ENTITY ADDRESS 3**	**FUNCTION**	
2	importduties:id_atlasfee	pp31	db1	inland_import	Atlasfee		
3	importduties:id_handowerfee	pp32	db1	inland_import	handowerfee		
4	importduties:id_hc_max	pp33	db1	inland_import	HC_max		
5	importduties:id_hc_min	pp34	db1	inland_import	HC_min		
6	importduties:id_hc_perkg	pp35	db1	inland_import	HC_perkg		
7	importduties:id_ic5_pos	pp36	db1	inland_import	IC_5pos		
8	importduties:id_icaddpos	pp37	db1	inland_import	IC_addpos		
9	importduties:id_leadtime	pp38	db1	inland_import	leadtime		
10	importduties:id_othercharges	pp39	db1	inland_import	othercharges		
11	importduties:id_storageperadc	pp40	db1	inland_import	storageperaddday		
12	importduties:id_T1fee	pp41	db1	inland_import	T1fee		
13	importduties:id_ta_max	pp42	db1	inland_import	TA_max		
14	importduties:id_ta_min	devpp42	db1	inland_import	TA_min		
15	importduties:id_ta_perkg	pp43	db1	inland_import	TA_perkg		
16	exportduties:ed_exportcustdo	pp44	db1	inland_export	exportcustomersdoc		
17	exportduties:ed_handlingmax	pp45	db1	inland_export	handlingchargesmax		
18	exportduties:ed_handlingmin	pp46	db1	inland_export	handlingchargesmin		
19	exportduties:ed_handlingperk(	pp47	db1	inland_export	handlingchargesperkg		
20	exportduties:ed_leadtime	pp48	db1	inland_export	leadtime		
21	exportduties:ed_othercharges	pp49	db1	inland_export	othercharges		
22	exportduties:ed_securitymax	pp50	db1	inland_export	securitymax		
23	exportduties:ed_securitymin	pp51	db1	inland_export	securitymin		
24	exportduties:ed_securityperkg	pp52	db1	inland_export	securityperkg		
25	exportduties:ed_Tl_max	pp53	db1	inland_export	Tlmax		
26	exportduties:ed_Tl_min	pp54	db1	inland_export	Tlmin		
27	exportduties:ed_Tl_perkg	pp55	db1	inland_export	Tlperkg		
28	exportduties:ed_tlairport	pp56	db1	inland_export	TLAirport		
29	exportduties:ed_tlcurrency	pp57	db1	inland_export	TLCurrency		
30	fuel:fsc_type	ipp38	db1	fcs_irc	fsc_type		
31	irc:irc_type	ipp39	db1	fcs_irc	irc_type		
32	carriers:ratestructure	m201	db2:db1	logistics:logistics	RateStructure:RateStructure		
33	packages:p_packtype	s300	db2:db2	logistics:packages	PackType:id		
34	packages:p_materialdescriptic	s301	db2	rates	commodity		
35	carriers:servicesummary	s500	db1:db1:db1:db1:db1	logistics:servicedescriptions:s	RateStructure:alds_transit_time:blds_transit_time:uds_transit_time:ALDS_1_44:BLDS_1_44:U		
36	route:routename	s600	db1:db1	lotdescriptions:lanedescriptior	LotNumber:LaneNumber		
37	commodity:name	s700	db1	commodity	name	commodity:id	
38	commodity:id	s701	db1	commodity	id		
39	transittimecodes:codes	s900	db1:db2	transittimecodes:tcodes	code:id		
40	carriers:name	c1	db1:db2	logistics:logistics	Awards:Awards		
41	services:servicename	s1	db1:db1:db1:db2	rates:rates:rates:rates	ALDS_1_44:BLDS_1_44:UDS_1_44:Service		
42	services:packtype	s3	db2	logistics	PackType		
43	services:commodity	s4	db2	rates	commodity	commodity:name	
44	services:transittime	s5	db2:db1:db1:db1:db1	servicedescriptions:servicedes	transit_time:alds_transit_time:blds_transit_time:u	carriers:ratestructure	

Output from R Statistical Package.

```
[Descriptive Statistics]
descriptive.table(DSall [c("TIME","ACCURACY")] ,
func.names =c("Mean","St. Deviation","Valid
N","Minimum","Maximum","Median"))
-- End Command     --
$`strata: all cases `
             Mean St. Deviation Valid N Minimum Maximum   Median
TIME     265.54444     171.34571      90      60     900 234.0000
ACCURACY  61.27976      29.22844      90       0     100  57.7381


[Accuracy Correlations]


corr.mat<-cor.matrix(variables=c(ACCURACY),
        with.variables=c(NODES,LEVELS,OVERLAPS,FUNCTIONS),
         data=DSall,
         test=cor.test,
         method='pearson',
        alternative="two.sided")
print(corr.mat)
qscatter_array(c(ACCURACY),
        c(NODES,LEVELS,OVERLAPS,FUNCTIONS),
        data=RS1g123456correctedremovedcgg61) +
geom_smooth(method="lm")
rm('corr.mat')
-- End Command     --

          Pearson's product-moment correlation

                ACCURACY
    NODES    cor -0.5667
               N 90
              CI* (-0.6925,-0.4074)
           stat** -6.452 (88)
         p-value 0.0000
----------------
   LEVELS    cor -0.6688
               N 90
              CI* (-0.7693,-0.5359)
           stat** -8.438 (88)
         p-value 0.0000
----------------
 OVERLAPS    cor -0.2936
               N 90
              CI* (-0.472,-0.09212)
           stat** -2.881 (88)
         p-value 0.0050
----------------
FUNCTIONS    cor -0.6452
               N 90
              CI* (-0.7518,-0.5057)
           stat** -7.922 (88)
         p-value 0.0000
----------------
        ** t (df)
         * 95% percent interval
```

```
HA: two.sided
```

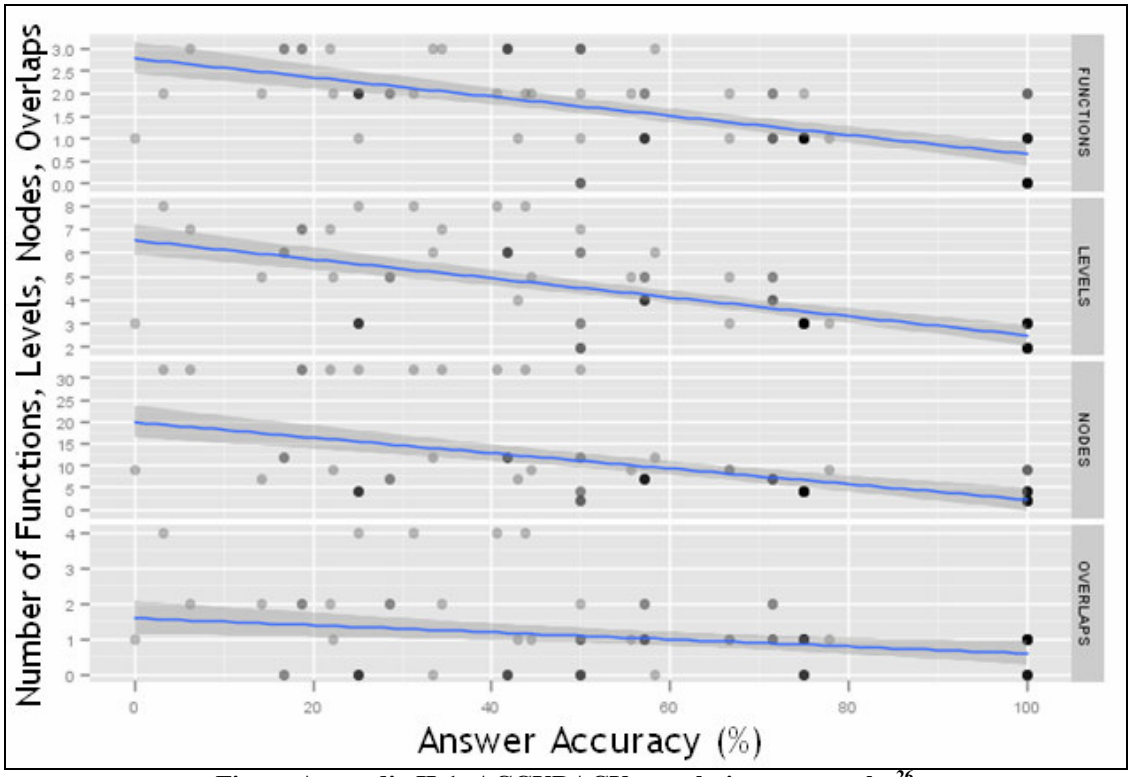

Figure Appendix II-1: ACCURACY correlation scatter plot[26]

[26] Note: The darker points on the plot are where multiple answers overlapped.

[Time Correlation]

```
corr.mat<-cor.matrix(variables=c(TIME),
        with.variables=c(NODES,LEVELS,OVERLAPS,FUNCTIONS),
         data=DSall,
         test=cor.test,
         method='pearson',
        alternative="two.sided")
print(corr.mat)
qscatter_array(c(TIME),
        c(NODES,LEVELS,OVERLAPS,FUNCTIONS),
        data=RS1g123456correctedremovedcgg61) +
geom_smooth(method="lm")
rm('corr.mat')
-- End Command     --

      Pearson's product-moment correlation

                TIME
    NODES    cor 0.3691
               N 90
              CI* (0.1754,0.5353)
           stat** 3.725 (88)
          p-value 3e-04
----------------
   LEVELS    cor 0.3610
               N 90
              CI* (0.1664,0.5286)
           stat** 3.632 (88)
          p-value 5e-04
----------------
 OVERLAPS    cor 0.2941
               N 90
              CI* (0.09269,0.4724)
           stat** 2.887 (88)
          p-value 0.0049
----------------
FUNCTIONS    cor 0.2251
               N 90
              CI* (0.01885,0.4129)
           stat** 2.167 (88)
          p-value 0.0329
----------------
        ** t (df)
         * 95% percent interval

        HA: two.sided
```

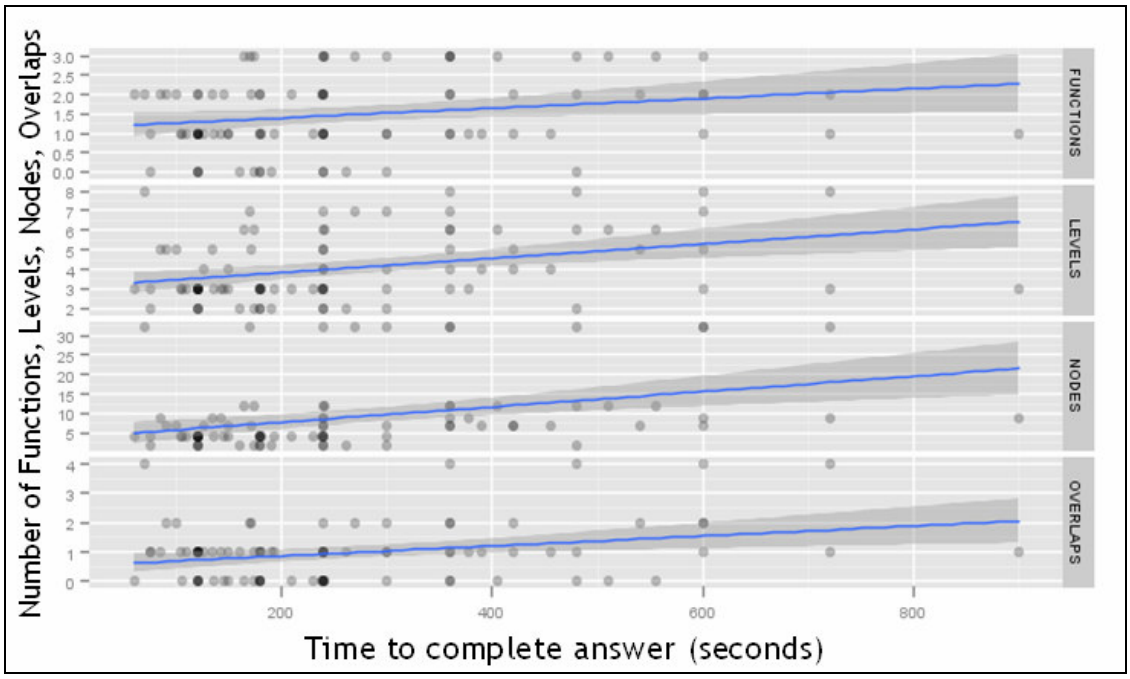

Figure Appendix II-2:: TIME - Correlation scatter plot

Experimental Data for Experiment Four

The mappings and ontologies files from experiment two were reused in experiment two

Experimental Data for Experiment Five

Ontology-Based Dependency Model (domestic electrical domain)

```
<?xml version="1.0"?>

<!DOCTYPE rdf:RDF [
    <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
    <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
    <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
    <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
]>

<rdf:RDF xmlns="http://www.owl-ontologies.com/Ontology1270901584.owl#"
     xml:base="http://www.owl-ontologies.com/Ontology1270901584.owl"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:owl="http://www.w3.org/2002/07/owl#">
    <owl:Ontology rdf:about=""/>
    <owl:Class rdf:ID="AE"/>
    <owl:ObjectProperty rdf:ID="app2controlunit">
        <rdfs:domain rdf:resource="#APPLIANCE"/>
        <rdfs:range rdf:resource="#CONTROLUNIT"/>
        <owl:inverseOf
rdf:resource="#inverse_of_transitive_symmetric_dependency_relation_9"/
>
        <rdfs:subPropertyOf
rdf:resource="#transitive_symmetric_dependency_relation"/>
    </owl:ObjectProperty>
    <owl:ObjectProperty rdf:ID="app2socket">
        <rdfs:domain rdf:resource="#APPLIANCE"/>
        <rdfs:range rdf:resource="#SOCKET"/>
        <owl:inverseOf
rdf:resource="#inverse_of_transitive_symmetric_dependency_relation_15"
/>
        <rdfs:subPropertyOf
rdf:resource="#transitive_symmetric_dependency_relation"/>
    </owl:ObjectProperty>
    <owl:Class rdf:ID="APPLIANCE">
        <rdfs:subClassOf rdf:resource="#AE"/>
    </owl:Class>
    <owl:Class rdf:ID="Cause">
        <rdfs:subClassOf rdf:resource="#DependencyAtrributes"/>
    </owl:Class>
    <owl:DatatypeProperty rdf:ID="cause_dst">
        <rdf:type rdf:resource="&owl;FunctionalProperty"/>
        <rdfs:domain rdf:resource="#Cause"/>
        <rdfs:range rdf:resource="&xsd;string"/>
    </owl:DatatypeProperty>
    <owl:DatatypeProperty rdf:ID="cause_src">
        <rdf:type rdf:resource="&owl;FunctionalProperty"/>
        <rdfs:domain rdf:resource="#Cause"/>
        <rdfs:range rdf:resource="&xsd;string"/>
    </owl:DatatypeProperty>
    <owl:DatatypeProperty rdf:ID="cause_value">
        <rdf:type rdf:resource="&owl;FunctionalProperty"/>
        <rdfs:domain rdf:resource="#Cause"/>
        <rdfs:range rdf:resource="&xsd;string"/>
```

```
    </owl:DatatypeProperty>
    <owl:Class rdf:ID="CONTROLUNIT">
        <rdfs:subClassOf rdf:resource="#AE"/>
    </owl:Class>
    <owl:ObjectProperty rdf:ID="controlunit2swfuse">
        <rdfs:domain rdf:resource="#CONTROLUNIT"/>
        <rdfs:range rdf:resource="#SWFUSE"/>
        <owl:inverseOf
rdf:resource="#inverse_of_transitive_symmetric_dependency_relation_10"
/>
        <rdfs:subPropertyOf
rdf:resource="#transitive_symmetric_dependency_relation"/>
    </owl:ObjectProperty>
    <owl:Class rdf:ID="COOKER">
        <rdfs:subClassOf rdf:resource="#APPLIANCE"/>
    </owl:Class>
    <APPLIANCE rdf:ID="COOKER1">
        <app2controlunit rdf:resource="#CU1"/>
    </APPLIANCE>
    <CONTROLUNIT rdf:ID="CU1">
        <controlunit2swfuse rdf:resource="#SWFUSE1_CT1"/>
        <inverse_of_transitive_symmetric_dependency_relation_9
rdf:resource="#COOKER1"/>
    </CONTROLUNIT>
    <owl:Class rdf:ID="DependencyAtrributes"/>
    <owl:ObjectProperty rdf:ID="DependencyRelation"/>
    <owl:Class rdf:ID="DI_APP_COOKER">
        <owl:equivalentClass>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#DependencyRelation"/>
                <owl:hasValue rdf:resource="#COOKER1"/>
            </owl:Restriction>
        </owl:equivalentClass>
    </owl:Class>
    <owl:Class rdf:ID="DI_APP_LIGHT1">
        <owl:equivalentClass>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#DependencyRelation"/>
                <owl:hasValue rdf:resource="#LIGHT1"/>
            </owl:Restriction>
        </owl:equivalentClass>
    </owl:Class>
    <owl:Class rdf:ID="DI_APP_LIGHT2">
        <owl:equivalentClass>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#DependencyRelation"/>
                <owl:hasValue rdf:resource="#LIGHT2"/>
            </owl:Restriction>
        </owl:equivalentClass>
    </owl:Class>
    <owl:Class rdf:ID="DI_APP_TV1">
        <owl:equivalentClass>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#DependencyRelation"/>
                <owl:hasValue rdf:resource="#TV1"/>
            </owl:Restriction>
        </owl:equivalentClass>
    </owl:Class>
    <owl:Class rdf:ID="DI_FUSE1">
        <owl:equivalentClass>
            <owl:Restriction>
```

```
            <owl:onProperty rdf:resource="#DependencyRelation"/>
            <owl:hasValue rdf:resource="#SWFUSE1_CT1"/>
        </owl:Restriction>
    </owl:equivalentClass>
</owl:Class>
<owl:Class rdf:ID="DI_FUSE2">
    <owl:equivalentClass>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#DependencyRelation"/>
            <owl:hasValue rdf:resource="#SWFUSE2_CT2"/>
        </owl:Restriction>
    </owl:equivalentClass>
</owl:Class>
<owl:Class rdf:ID="DI_FUSE3">
    <owl:equivalentClass>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#DependencyRelation"/>
            <owl:hasValue rdf:resource="#SWFUSE3_CT3"/>
        </owl:Restriction>
    </owl:equivalentClass>
</owl:Class>
<owl:ObjectProperty rdf:ID="funtional_dependency_relation">
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <rdfs:subPropertyOf rdf:resource="#DependencyRelation"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hascauseattribute">
    <rdfs:domain rdf:resource="#AE"/>
    <rdfs:range rdf:resource="#Cause"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasimpactattribute">
    <rdfs:domain rdf:resource="#AE"/>
    <rdfs:range rdf:resource="#Impact"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasstrenghtattribute">
    <rdfs:domain rdf:resource="#AE"/>
    <rdfs:range rdf:resource="#Strength"/>
</owl:ObjectProperty>
<owl:Class rdf:ID="Impact">
    <rdfs:subClassOf rdf:resource="#DependencyAtrributes"/>
</owl:Class>
<owl:DatatypeProperty rdf:ID="impact_dst">
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <rdfs:domain rdf:resource="#Impact"/>
    <rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="impact_src">
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <rdfs:domain rdf:resource="#Impact"/>
    <rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="impact_value">
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <rdfs:domain rdf:resource="#Impact"/>
    <rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>
<owl:ObjectProperty rdf:ID="inverse_functional_relations">
    <rdf:type rdf:resource="&owl;InverseFunctionalProperty"/>
    <rdfs:subPropertyOf rdf:resource="#DependencyRelation"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="inverse_of_junction2junction">
    <rdfs:domain rdf:resource="#JUNCTION"/>
```

```
        <rdfs:range rdf:resource="#JUNCTION"/>
        <owl:inverseOf rdf:resource="#junction2junction"/>
        <rdfs:subPropertyOf
rdf:resource="#symmetic_dependency_relation"/>
    </owl:ObjectProperty>
    <owl:ObjectProperty rdf:ID="inverse_of_junction2swfuse">
        <rdfs:domain rdf:resource="#SWFUSE"/>
        <rdfs:range rdf:resource="#JUNCTION"/>
        <owl:inverseOf rdf:resource="#junction2swfuse"/>
        <rdfs:subPropertyOf
rdf:resource="#transitive_symmetric_dependency_relation"/>
    </owl:ObjectProperty>
    <owl:ObjectProperty
rdf:ID="inverse_of_transitive_symmetric_dependency_relation_10">
        <rdfs:domain rdf:resource="#SWFUSE"/>
        <rdfs:range rdf:resource="#CONTROLUNIT"/>
        <owl:inverseOf rdf:resource="#controlunit2swfuse"/>
        <rdfs:subPropertyOf
rdf:resource="#transitive_symmetric_dependency_relation"/>
    </owl:ObjectProperty>
    <owl:ObjectProperty
rdf:ID="inverse_of_transitive_symmetric_dependency_relation_15">
        <rdfs:domain rdf:resource="#SOCKET"/>
        <rdfs:range rdf:resource="#APPLIANCE"/>
        <owl:inverseOf rdf:resource="#app2socket"/>
        <rdfs:subPropertyOf
rdf:resource="#transitive_symmetric_dependency_relation"/>
    </owl:ObjectProperty>
    <owl:ObjectProperty
rdf:ID="inverse_of_transitive_symmetric_dependency_relation_16">
        <rdfs:domain rdf:resource="#SWFUSE"/>
        <rdfs:range rdf:resource="#SOCKET"/>
        <owl:inverseOf rdf:resource="#socket2swfuse"/>
        <rdfs:subPropertyOf
rdf:resource="#transitive_symmetric_dependency_relation"/>
    </owl:ObjectProperty>
    <owl:ObjectProperty
rdf:ID="inverse_of_transitive_symmetric_dependency_relation_17">
        <rdfs:subPropertyOf
rdf:resource="#transitive_symmetric_dependency_relation"/>
    </owl:ObjectProperty>
    <owl:ObjectProperty
rdf:ID="inverse_of_transitive_symmetric_dependency_relation_2">
        <rdfs:domain rdf:resource="#SWITCH"/>
        <rdfs:range rdf:resource="#LIGHT"/>
        <owl:inverseOf rdf:resource="#light2switch"/>
        <rdfs:subPropertyOf
rdf:resource="#transitive_symmetric_dependency_relation"/>
    </owl:ObjectProperty>
    <owl:ObjectProperty
rdf:ID="inverse_of_transitive_symmetric_dependency_relation_3">
        <rdfs:domain rdf:resource="#JUNCTION"/>
        <rdfs:range rdf:resource="#SWITCH"/>
        <owl:inverseOf rdf:resource="#switch2junction"/>
        <rdfs:subPropertyOf
rdf:resource="#transitive_symmetric_dependency_relation"/>
    </owl:ObjectProperty>
    <owl:ObjectProperty
rdf:ID="inverse_of_transitive_symmetric_dependency_relation_9">
        <rdfs:domain rdf:resource="#CONTROLUNIT"/>
        <rdfs:range rdf:resource="#APPLIANCE"/>
```

```
        <owl:inverseOf rdf:resource="#app2controlunit"/>
        <rdfs:subPropertyOf
rdf:resource="#transitive_symmetric_dependency_relation"/>
    </owl:ObjectProperty>
    <owl:Class rdf:ID="JUNCTION">
        <rdfs:subClassOf rdf:resource="#AE"/>
    </owl:Class>
    <JUNCTION rdf:ID="JUNCTION1">
        <inverse_of_junction2junction rdf:resource="#JUNCTION2"/>
        <inverse_of_transitive_symmetric_dependency_relation_3
rdf:resource="#SWITCH1"/>
        <junction2swfuse rdf:resource="#SWFUSE2_CT2"/>
    </JUNCTION>
    <JUNCTION rdf:ID="JUNCTION2">
        <junction2junction rdf:resource="#JUNCTION1"/>
        <inverse_of_transitive_symmetric_dependency_relation_3
rdf:resource="#SWITCH2"/>
    </JUNCTION>
    <owl:ObjectProperty rdf:ID="junction2junction">
        <rdfs:domain rdf:resource="#JUNCTION"/>
        <rdfs:range rdf:resource="#JUNCTION"/>
        <owl:inverseOf rdf:resource="#inverse_of_junction2junction"/>
        <rdfs:subPropertyOf
rdf:resource="#symmetic_dependency_relation"/>
    </owl:ObjectProperty>
    <owl:ObjectProperty rdf:ID="junction2swfuse">
        <rdfs:domain rdf:resource="#JUNCTION"/>
        <rdfs:range rdf:resource="#SWFUSE"/>
        <owl:inverseOf rdf:resource="#inverse_of_junction2swfuse"/>
        <rdfs:subPropertyOf
rdf:resource="#transitive_symmetric_dependency_relation"/>
    </owl:ObjectProperty>
    <APPLIANCE rdf:ID="LAMP1">
        <app2socket rdf:resource="#SOCKET_2"/>
    </APPLIANCE>
    <owl:Class rdf:ID="LIGHT">
        <rdfs:subClassOf rdf:resource="#APPLIANCE"/>
    </owl:Class>
    <LIGHT rdf:ID="LIGHT1">
        <light2switch rdf:resource="#SWITCH1"/>
    </LIGHT>
    <LIGHT rdf:ID="LIGHT2">
        <light2switch rdf:resource="#SWITCH2"/>
    </LIGHT>
    <owl:ObjectProperty rdf:ID="light2switch">
        <rdfs:domain rdf:resource="#LIGHT"/>
        <rdfs:range rdf:resource="#SWITCH"/>
        <owl:inverseOf
rdf:resource="#inverse_of_transitive_symmetric_dependency_relation_2"/
>
        <rdfs:subPropertyOf
rdf:resource="#transitive_symmetric_dependency_relation"/>
    </owl:ObjectProperty>
    <owl:DatatypeProperty rdf:ID="lvl_dst"/>
    <owl:DatatypeProperty rdf:ID="lvl_level"/>
    <owl:DatatypeProperty rdf:ID="lvl_src"/>
    <owl:Class rdf:ID="SOCKET">
        <rdfs:subClassOf rdf:resource="#AE"/>
    </owl:Class>
    <owl:ObjectProperty rdf:ID="socket2swfuse">
        <rdfs:domain rdf:resource="#SOCKET"/>
```

```
        <rdfs:range rdf:resource="#SWFUSE"/>
        <owl:inverseOf
rdf:resource="#inverse_of_transitive_symmetric_dependency_relation_16"
/>
        <rdfs:subPropertyOf
rdf:resource="#transitive_symmetric_dependency_relation"/>
    </owl:ObjectProperty>
    <SOCKET rdf:ID="SOCKET_1">
        <inverse_of_transitive_symmetric_dependency_relation_15
rdf:resource="#TV1"/>
        <socket2swfuse rdf:resource="#SWFUSE3_CT3"/>
    </SOCKET>
    <SOCKET rdf:ID="SOCKET_2">
        <inverse_of_transitive_symmetric_dependency_relation_15
rdf:resource="#LAMP1"/>
        <socket2swfuse rdf:resource="#SWFUSE3_CT3"/>
    </SOCKET>
    <owl:Class rdf:ID="Strength">
        <rdfs:subClassOf rdf:resource="#DependencyAtrributes"/>
    </owl:Class>
    <owl:DatatypeProperty rdf:ID="strength_dst">
        <rdf:type rdf:resource="&owl;FunctionalProperty"/>
        <rdfs:domain rdf:resource="#Strength"/>
        <rdfs:range rdf:resource="&xsd;string"/>
    </owl:DatatypeProperty>
    <owl:DatatypeProperty rdf:ID="strength_src">
        <rdf:type rdf:resource="&owl;FunctionalProperty"/>
        <rdfs:domain rdf:resource="#Strength"/>
        <rdfs:range rdf:resource="&xsd;string"/>
    </owl:DatatypeProperty>
    <owl:DatatypeProperty rdf:ID="strength_value">
        <rdf:type rdf:resource="&owl;FunctionalProperty"/>
        <rdfs:domain rdf:resource="#Strength"/>
        <rdfs:range rdf:resource="&xsd;int"/>
    </owl:DatatypeProperty>
    <owl:Class rdf:ID="SWFUSE">
        <rdfs:subClassOf rdf:resource="#AE"/>
    </owl:Class>
    <SWFUSE rdf:ID="SWFUSE1_CT1">
        <inverse_of_transitive_symmetric_dependency_relation_10
rdf:resource="#CU1"/>
    </SWFUSE>
    <SWFUSE rdf:ID="SWFUSE2_CT2">
        <inverse_of_junction2swfuse rdf:resource="#JUNCTION1"/>
    </SWFUSE>
    <SWFUSE rdf:ID="SWFUSE3_CT3">
        <inverse_of_transitive_symmetric_dependency_relation_16
rdf:resource="#SOCKET_1"/>
        <inverse_of_transitive_symmetric_dependency_relation_16
rdf:resource="#SOCKET_2"/>
    </SWFUSE>
    <owl:Class rdf:ID="SWITCH">
        <rdfs:subClassOf rdf:resource="#AE"/>
    </owl:Class>
    <SWITCH rdf:ID="SWITCH1">
        <switch2junction rdf:resource="#JUNCTION1"/>
        <inverse_of_transitive_symmetric_dependency_relation_2
rdf:resource="#LIGHT1"/>
    </SWITCH>
    <SWITCH rdf:ID="SWITCH2">
        <switch2junction rdf:resource="#JUNCTION2"/>
```

```
        <inverse_of_transitive_symmetric_dependency_relation_2
rdf:resource="#LIGHT2"/>
    </SWITCH>
    <owl:ObjectProperty rdf:ID="switch2junction">
        <rdfs:domain rdf:resource="#SWITCH"/>
        <rdfs:range rdf:resource="#JUNCTION"/>
        <owl:inverseOf
rdf:resource="#inverse_of_transitive_symmetric_dependency_relation_3"/
>
        <rdfs:subPropertyOf
rdf:resource="#transitive_symmetric_dependency_relation"/>
    </owl:ObjectProperty>
    <owl:ObjectProperty rdf:ID="symmetic_dependency_relation">
        <rdf:type rdf:resource="&owl;SymmetricProperty"/>
        <owl:inverseOf rdf:resource="#symmetic_dependency_relation"/>
        <rdfs:subPropertyOf rdf:resource="#DependencyRelation"/>
    </owl:ObjectProperty>
    <owl:ObjectProperty rdf:ID="transitive_dependency_relation">
        <rdf:type rdf:resource="&owl;TransitiveProperty"/>
        <rdfs:subPropertyOf rdf:resource="#DependencyRelation"/>
    </owl:ObjectProperty>
    <owl:ObjectProperty
rdf:ID="transitive_symmetric_dependency_relation">
        <rdf:type rdf:resource="&owl;SymmetricProperty"/>
        <rdf:type rdf:resource="&owl;TransitiveProperty"/>
        <owl:inverseOf
rdf:resource="#transitive_symmetric_dependency_relation"/>
        <rdfs:subPropertyOf rdf:resource="#DependencyRelation"/>
    </owl:ObjectProperty>
    <owl:Class rdf:ID="TV">
        <rdfs:subClassOf rdf:resource="#APPLIANCE"/>
    </owl:Class>
    <APPLIANCE rdf:ID="TV1">
        <app2socket rdf:resource="#SOCKET_1"/>
    </APPLIANCE>
</rdf:RDF>
```

APPENDIX III

This appendix describes a simple worked example to illustrate the outputs of the TomE tool. It assumes the existence of a mapping file based on a simplification of the mappings using in the generalised ontology-based integration system designed for experiment one.

Simplifying Assumptions Made.

Assumption 1: Assume that the domain under study has three architectural elements called UE, MP and GE that represents the output of the decomposition step in the process described in Figure 3-4. Figure III-1 below, shows the architectural elements for "GE", "UE" and "MP" as subclasses for the ontology-based metamodel concept "ArchitecturalEntities".

These architectural elements are made dependent by adding specialised concepts for the dependent relations (ue2mp and mp2ge) as shown in Figure III-1.

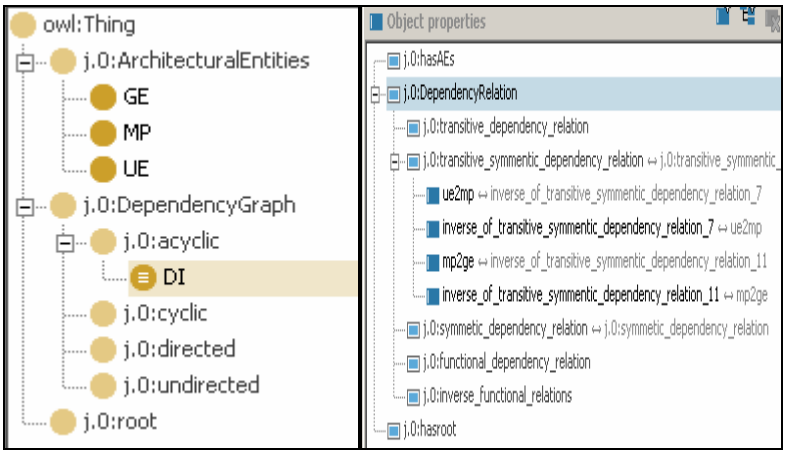

Figure III-1: Dependency Model Classes and Dependency Relations

Assumption 2: Assume a mapping file from an ontology integration system that has the following dependencies in its mappings:

Mapping 1: UE1->MP1->GE1
->GE2

Mapping 2: UE2->MP2->GE2
->GE3

Mapping 3: UE3->MP3->GE3

Mapping 4: UE4->MP4->GE4 [F{UE4}]

Mapping 5: UE5->MP5->GE5

Assumption 3: Assume that these mappings can be interpreted as follows:

An ontological concept called UE1 has a mapping called MP1. The mapping MP1 collects information from data source resources identified as GE1 and GE2.

Mappings MP1 and MP2 share a common database resource (GE2).

The mapping for UE4 is called MP4. Mapping MP4 has a function specified, called "F" that requires access to ontological concept UE4.

In the case of this simple mappings file, we can see that MP1 and MP2 have a dependency relation due to GE2. Mappings MP4 and MP5 also have a dependency relation due to the function F that accesses UE4.

TomE tool output

The output of the TomE for this mapping file produces the graphical views shown in the figures below.

The full graph of dependencies (Figure III-2) shows that there is a dependency relationship between MP1 and MP2 and another dependency relationship between MP4 and MP5.

This is the view of all dependencies provided by the TomE system.

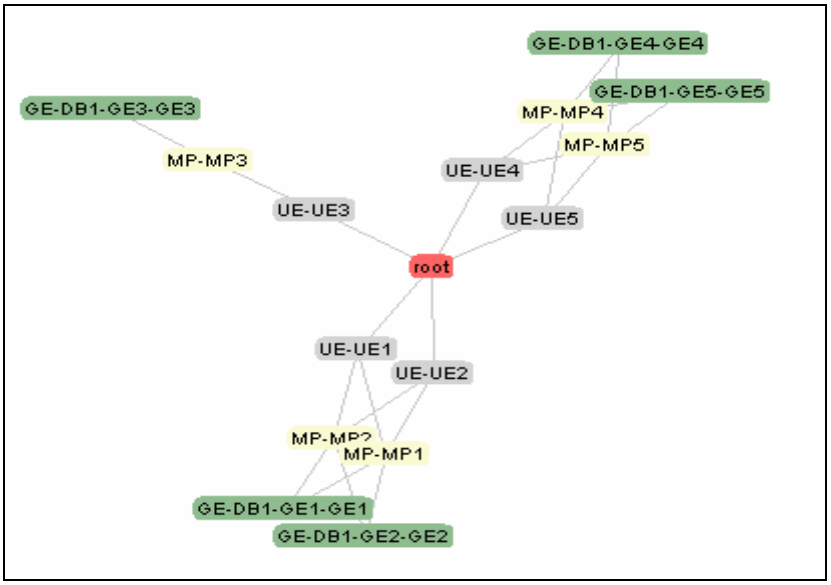

Figure III-2: Full Dependency Graph

Figure III-2 shows the all the dependencies chain that have been computed.

The "root" node represents the metadata for the graph itself and contains the type and version information. Nodes of different types have different colours to aid viewing. The "root" node can be populated using the information (e.g. name, version) from the "Dependency Graph" concept from the OBDM.

The name of each node is a concatenation of the node type and node instance name (e.g., UE-UE1 indicated a node of type UE with instance UE1).

The dependency view for UE1 is shown in Figure III-3 below.

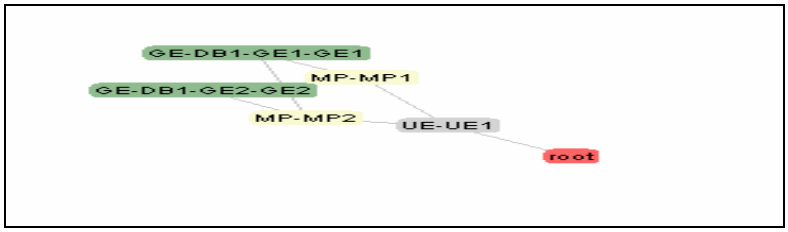

Figure III-3: Single Dependency Graph

This view can be selected either by clicking on the required node on the main graph view or by loading it directly using the "file->load" option in the menu bar in the TomE tool.

The dependency view with levels and types for UE1 is shown in Figure III-4 below.

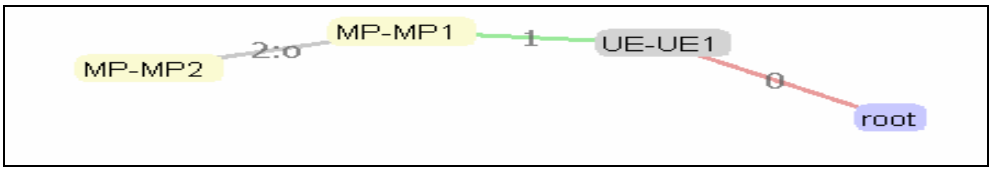

Figure III-4: Dependency View with Levels

APPENDIX IV

This appendix describes the directory structure of the DVD which contains the Java code for generalised ontology-based integration test bed (HotFusion) and TomE tools.

The following directory structure and contents are provided on DVD

root/HotFusion/src	Java source classes files for the generalised ontology based integration test bed
root/HotFusion/config	Eclipse project classes files for the generalised ontology based integration test bed
root/HotFusion/docs	Readme file
root/TomE /src	Java source classes files for TomE tool
root/TomE/config	Eclipse project files for TomE tool
root/TomE/docs	Readme file
root/ontologies	OWL files for the dependency metamodel and OBDM.
root/mappings	Mapping files used in this thesis.