Context-Informed Semantic Interoperation

A thesis submitted to the

University of Dublin, Trinity College

in fulfillment of the requirements for the degree of

Doctor of Philosophy

Alexander O'Connor

Knowledge and Data Engineering Group,

School of Computer Science & Statistics,

Trinity College Dublin,

Ireland.

2010

Declaration

Acknowledgements

The author would like to acknowledge the invaluable support of his Supervisor, Prof. Vincent Wade, in undertaking this work. The author wishes to thank Dr. Owen Conlan & Prof. John O'Leary for their help and advice in the compilation of this thesis.

This work was funded by the Embark Initiative of the Irish Research Council for Science, Engineering and Technology, funded by the National Development Plan, and by the Centre for Next Generation Localisation, Science Foundation Ireland (Grant 07/CE/I1142).

Alexander O'Connor

University of Dublin, Trinity College

2010

Abstract

A common trend in modern applications is the move towards more mobile, adaptive, customisable software. The evolution of software from static, invariant tools for narrow portions of a task to adaptive, open interaction frameworks is embodied in the use of a variety of technologies for creating a reconfigurable application. The key challenge to improving application behaviour in response to external knowledge is in making the representation of that external knowledge compatible with the application's representation. This external information, relevant to the user and their task is commonly known as context information. In the past, context information has typically been integrated using an *a-priori* model of context, which constrains the type of information which can be used as context. This thesis presents a model for context integration which does not depend on an a-priori model of context.

This thesis presents a novel approach to integrating contextual information through the use of a context mediator based on ontology mediation. This *context-informed semantic interoperation* approach is based on the exchange of both schema and instance data, in the form of ontologies, between heterogeneous sources of context and a target application. The mediator represents the collective knowledge of a contextual situation by linking ontologies in their native form through a shared semantic view. The approach is innovative in that it combines user-defined and ontological reasoning to provide a more expressive method for bridging differences in representations between different sources and their target without an *a-priori* model. It demonstrates the use of semantic interoperation as an approach to allowing richer knowledge exchange between applications and their surroundings.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

This chapter presents the motivation and objectives for red delivering context to user applications using semantic techniques. This is followed by an analysis of the key challenges associated with completing the stated objectives. A technical approach section describes the process that addressed the challenges associated context-informed semantic interoperation. Finally, a guide to the following chapters of the thesis is provided.

1.1 Motivation

The volume of information which a user has to deal with is increasing all the time, and it is necessary for applications to become better, more intelligent assistants to users[1]. As users become more connected, and more mobile, there is a need for applications to respond to the contextual factors affecting the user and their activity [Economist, 2003]. In order to achieve this, applications must consider a wide variety of information about 'what' the user intends to achieve, as well as the 'how', 'where' and 'why' of their task [Oh and Woo, 2005]. This need for improved personalisation is one of the reasons for the considerable research attention in the

[1]This issue has also been observed in the area of search, where it has been addressed in work such as [Kruschwitz and Al-Bakour, 2005]

area of context-awareness [Chaari et al., 2007, Bolchini et al., 2007].

Contextual information can be acquired from a wide variety of sources, in a wide variety of forms. It was recognised that while location is a useful, even typical, contextual axis, merely presenting the user's co-ordinates to an application is insufficient[Schmidt et al., 1999]. As the number of sources and the types of context information expands, it is progressively more difficult to model context explicitly in each application. The necessary result of this was that applications were restricted to domains that were often location-driven [Ratto et al., 2003][Bardram et al., 2003] and therefore unable to account for the broad spectrum of potential contextual information. While traditional approaches to context-aware computing began by directly interacting with the infrastructure through specific protocols [Lonsdale et al., 2004], this began to give way to a more service oriented approach, for example [Gu et al., 2005, Brønstead et al., 2007] in order to facilitate a broader model of context.

A service-oriented approach to context makes it more compatible with the Semantic Web [Shadbolt et al., 2006], specifically Ontologies. "An ontology is an explicit specification of a conceptualization [*sic.*]" [Gruber, 1993], that is, a formal set of rules and definitions for entities modelled by the ontology. Ontologies offer a method for modelling not only specific data, but also the knowledge associated with that data. Ontologies offer a ready method for permitting applications to describe, in an exchangeable manner, the details of their knowledge and the relations associated with it. An ontology is specifically designed as a means for transferring and sharing information, and when applications share a common world view, they can effectively make use of ontologies to transfer information.

However, in the area of context, the underlying assumption of a common world view is not available. An ontology represents a knowledge perspective on an information model (*i.e.* a conceptualisation) and one of the objectives of this work is to present a method for representing and integrating context which is subject to fewer modelling constraints and application-domain biases than previous work has demonstrated.

In order to define 'context', it is necessary to examine some of the current and earlier work that led to the creation of contextual systems. Context is commonly thought of as coming from the physical realm through sensors, and characterises a user's situation [Dey, 2001]. This definition points to the key feature of context, which is that it is not information that is generated by the application, nor is it information that is directly input by the user. Many similar definitions of context, such as [Chen and Kotz, 2000][Judd and Steenkiste, 2003][Pradhan, 2000], extend the notion to include histories of context, an extensible device model, or relative/semantic values for data. The key feature of this work is that the nature and properties of context exist only relative to the application and the situation. A effective context integration approach must therefore be focused on the agreement between the user's application and its surrounding services.

The use of semantics to represent context, as seen in [Chen et al., 2004a][Mrissa et al., 2008] allowed applications to address the issue of context integration as one of knowledge management. This shifted the objective of designing context systems from a tightly-coupled model-sensor system to an explicitly separate architecture composed of knowledge and data tiers. A knowledge-based approach also facilitated the inclusion of information which was not derived from sensors, but from the information held in other applications.

The realisation that context is by definition situational and that the choice of contextual information is dependent on the applications, users and domains involved is the key to understanding context as presented in this work. Context is that information which, for whatever reason, has not been integrated into an application *a-priori*, but which can be usefully gathered and incorporated when the user's situation becomes apparent.

In this definition, context can be considered as that information which is situationally useful to the user, which has not been integrated by the application designer explicitly. While the advantage to viewing contextual information in this manner is that it represents the potential to gain from a rich variety of sources, it is also associated with the difficulty that much of the attributes of the information are unknown. Furthermore, the objective of integrating data in different representations becomes central to the design of any apparatus for supporting context. In order to address this, context inclusion will be considered as a case of intelligent interoperation [Qian, 1993]: the transfer of new knowledge to an application not explicitly designed for that knowledge.

The motivation of this work is to improve the behaviour of applications by manipulating their knowledge via context. External information, derived from the user's physical and conceptual surroundings, which might be expressed in different forms or formats, is transferred to a user's application, based on their situational need. This is intended as a means to reduce the burden of information management on users, by uniting information held in disparate systems.

This will be achieved through the use of semantic interoperation, semantically-rich linking of knowledge, in context integration. By establishing an interoperation pathway, it is intended to show that information brokered through ontological descriptions can be leveraged in new ways to provide external information to user's applications, thereby improving an application's knowledge.

1.1.1 Motivating Example

The authors in [Mark, 2002] describe a process for facilitating teams of engineers from different disciplines in close collaboration. One of the key features of this 'Extreme Collaboration' is that the groups are supported by a moderator, who helps to select relevant support documentation and data based on each team's assigned task and expertise. In the paper, this moderator is a human, who needs to be an expert in each of team members' skills. It is possible to imagine an application which attempts to provide a similar service.

The application might operate by looking at the interest profile of each member of the group and comparing it with the available list of documents which could be displayed. In its basic operation, this system is a relatively simple document retrieval service, which can match subject keywords from the group profile to tags in document collections. However, there are some key design questions that arise:

- How does the application recognise a group? What if some of the group members are attending virtually?
- How does the application correlate a particular user's interests with the tags in the document collections?
- What happens if some of the users' profiles are in a different format?
- How does the application integrate new document collections, or changes to collections?

All of these considerations are difficult to model completely within a particular application. Instead, it is desirable to allow this broad, external knowledge to be delegated to a mediator which can take a higher-level view of all of the participating parts of the collaboration. In this thesis, this contextual integration is achieved through the use of ontologies to structure the knowledge in each of the services, and to find links between those services. The challenges then arise in finding agreement between those representations in a way that allows the application to provide better functionality. A particular feature of the approach in this thesis is that it achieves this without attempting to dictate in advance what information should be considered contextual – the nature of context depends on the situation.

1.2 Challenges to Informing Context

The issue of context can be divided into a number of challenges [Mitra et al., 2005]. The first challenge is to **describe** and **execute** the conceptual relationships between different ontologies and **identify** important information that needs to be transferred. The second challenge is to make**use** of this information: that is, to **transfer** the information in a form that can be understood by the user's application. This transfer involves **accessing** the information, **structuring** it properly and, perhaps most importantly, **ensuring** it is in a form that agrees with the semantics of the destination application. These challenges will be addressed by the structure and design presented in this thesis.

One issue that is not addressed in this work is that of service discovery itself. There are a number of initiatives in the area of discovering services, beginning with UDDI[2] [OASIS, 2004], and on to the OWL-S [Martin, 2004] annotations for services. The issue of discovery in the specific domain of context has also been addressed, for example in [Thomson et al., 2003]. These technologies represent an approach appropriate to tightly-coupled, orchestrated services, suitable for close service integration. The approach being undertaken in this thesis is focused on loosely-coupled information, and so would require a different approach to finding services with information. The existence of services available at a specific situation is assumed to have been arranged externally, while the relevance of specific knowledge within the service is identified by the system.

The key challenge in this work is to be able to take account of a wider variety of context information, and provide enrichment which allows target applications to alter their behaviour in response to new types of knowledge as well as new information from context. Traditional approaches to context have focused on the development of a broad, pre-existing model of context, often centred on location. This reduces the flexibility of the system by pre-determining the nature and behaviour of context. The large fixed model can also be difficult to manage, particularly where systems are required to interoperate with it. In this work, the representation of context is not pre-determined, but emerges from the combination of concepts available in context with the application's knowledge. This allows the mediator to focus on the concepts which are directly relevant to a particular context.

1.3 Research Question

Can a Context Mediator[3] which uses semantic interoperation instead of an *a-priori* general model of context be designed to effectively influence the behaviour of a target application?

[2]Universal Description, Discovery and Integration

[3]A Context Mediator is an application which is able to analyse ontological knowledge and broker information from sources of context to enrich a target application.

1.3.1 Objectives

The aim of this thesis is to establish a theoretical and practical framework for integrating contextual information into applications. The framework approach is based on a semantic interoperation, which takes advantage of the extensive work in ontologies, ontology matching and other metadata formalisations to identify and translate relevant contextual information. These formalisations are used to establish access for applications to knowledge held in separate, remote services. This access is achieved through the use of rich descriptions represented in an overlay semantic network.

In order to address the research question of the thesis, and to establish an evaluation for the approach, the following objectives were defined:

1. To undertake state of the art studies both on context, and semantic interoperation
2. Develop a context mediator infrastructure to allow integration of service registration and mapping
3. Evaluation of this infrastructure so as to derive an understanding of the advantages and disadvantages of informing context

1.3.2 Contribution

The major contribution to the state of the art is to apply semantic interoperation to context using ontology mediation. The framework developed uses a novel approach to importing, arranging and using alignment descriptions from ontology matching tools. This approach is novel in that it represents an attempt to approach context as a problem of information exchange without creating an integrated model of context in advance. The enrichment of the target application to include both schematic knowledge enrichment as well as data enrichment is a key advantage of this approach, and is derived from the direct use of the collective knowledge of the sources of context and the target application. Systems such as the one presented in this work have also been highlighted in the literature as an important contribution to the state of the art in semantic interoperation [Euzenat and Schvaiko, 2007]:

> *In the long term, we also expect substantial progress on the frameworks for integrating different matching systems. In fact, infrastructures, which are able to store and provide alignments to those who need it, are still missing. Such an infrastructure should also match ontologies and process the alignments on specified data.*

The representation of semantic interoperation requires a rich, flexible representation for the relationship between separate concepts in different ontologies. The minor contribution of this work is an assessment of the suitability and performance of Topic Map technology as a medium for representing rich semantic relationships between different ontological entities. The creation of Topic Maps to express particular mappings for different contexts is a novel approach to the problem of context, and to the problem of consuming and using the mappings output by semantic interoperation utilities.

1.3.3 Technical Approach

Initially, a study of published descriptions of context-aware systems was created to help establish a set of requirements specific to contextual integration. This helped to identify particular properties of context-aware and context-informed systems, as well as the different information modelling and management approaches. A broad base of systems was analysed, ranging from traditional context-aware systems to semantic, service-based context systems. The comparison of these systems revealed the evolutionary steps in creating them, as well as the advantages and disadvantages of different approaches to modelling and integrating context.

The second study undertaken was designed to give grounding in the area of semantic interoperation, specifically with regard to ontology mapping and data transfer. While this project does not specifically address the area of (semi-)automatic ontology matching *per se*, a number of these tools were examined so as to be able to decide what their capabilities were from the perspective of integrating their mappings. The second part of this study focused on mediation tools and mapping repositories, this study undertook to examine the different approaches to representing mappings, along with their advantages and disadvantages. In addition, the systems were specifically analysed from the perspective of understanding how they might interact with the requirements generated from the previous study on context. This analysis generated a general set of properties required for a Context Mediator. A Context Mediator is envisaged as being at the heart of an informing environment, and is responsible for maintaining the links between the sources of contextual information and the target of the interoperation, the user's application.

Having established detailed requirements and challenges for a Context Mediator as part of a context-informed environment, the next step was to develop detailed design parameters for the system which incorporated both the requirements from the state of the art, and the requirements of the novel approach to context interoperation and enrichment described in the approach. A technology evaluation was performed, followed by a use case analysis and the establishment of an outline architecture.

An initial implementation was created to integrate the supporting frameworks and libraries, where available, to form the basis of a functional mediator. The Java programming language was selected as the implementation language, as well as a number of XML[4] technologies including OWL[5] and XTM Topic Maps. The implementation of an initial framework was extended during the case studies devised for evaluation.

The research evaluation methodology was based on case studies, drawn from scenarios found in both the semantic integration and context domains. Specific focus for the analysis of the evaluation has been put on the kinds of information gains made possible by different kinds of application and mediator capabilities. Additionally, different approaches to constructing and importing mappings between ontologies were assessed for performance and expressiveness. The case studies were supplemented

[4]eXtensible Markup Language
[5]Web Ontology Language

with a comparative analysis of systems identified in the state of the art studies, so as to define the novelty of the system presented.

This evaluation led to a set of conclusions which include an assessment of the advantages and disadvantages of semantic interoperation in support of context, as well as some areas of possible future work.

1.4 Thesis Outline

- **Chapter Two:** The design properties of context-aware systems are discussed, along with some definitions for concepts such as *context* and *informing context*. This chapter analyses and compares context systems and surveys of context to draw a set of key properties for the context integration approach in this thesis.
- **Chapter Three** includes a study of semantic interoperation tools and techniques. The initial part of the study concentrates on the techniques directly related to establishing links between ontologies, and the representation of the resulting mappings. The chapter then makes a detailed assessment of other ontology mediation tools. The analysis of these systems is focused on creating a list of key properties necessary for an effective ontology mediator.
- **Chapter Four** describes the design that is the result of the analysis from the second and third chapters. This chapter presents an abstract framework for context-informed semantic interoperation, as well a a detailed discussion of the design goals, requirements and architecture of a context-informed ontology mediator.
- **Chapter Five** presents an implementation of the design in the previous chapter. In particular, this chapter presents the scope of this implementation and describes key implementation features. The structure of the main interfaces in the system are discussed along with a functionality walk-though using a motivating example.

- **Chapter Six** describes the evaluation work undertaken to validate the contributions in the thesis. This includes two case studies used to evaluate the system in terms of key properties of a context-informed mediator and a comparative assessment of the context-informed mediator with respect to systems surveyed in chapters two and three. These are supplemented by analysis which describes the key findings across each part of the study.

- **Chapter Seven** presents the conclusions of the thesis. This includes key findings on the major and minor contributions of the work, as well as a description of the successful completion of the objectives defined in chapter one. The research publications associated with this work are listed, and the thesis concludes with a discussion of possible future work.

Chapter 2

State of the Art in Context Integration

This chapter begins with a characterisation of the nature of context information, as defined in the literature. This discussion leads to a definition of context integration. Three key, semantic context systems are then surveyed, with a particular interest on the nature of their information model, participants and architecture. The systems are analysed to identify the best features of each approach, and the important missing features from current work. The findings are summarised at the end of the chapter.

2.1 Introduction

This chapter presents the issues associated with contextual information, addressing both the principles and application of context. The chapter includes a discussion of the definition of context, drawn in particular from a number of selected surveys. The definition and properties of context which will be used in this thesis are measured in the light of a context-informed, knowledge-based system. This distinction between *aware* and *informed* context is defined in the course of the chapter.

In order to establish a suitable basis, several surveys of context systems were chosen to begin this study. The analyses and conclusions of those surveys formed the basis for the analysis which is extended here.

Once an analytical basis was established, the chapter presented several context systems which include some of the important aspects for distinguishing best practice for supporting applications and context. There is considerable heterogeneity between the systems, and the objective of the survey is to identify the requirements for a context-informed system. These requirements are developed at the end of the chapter.

2.2 Characterisation of Context in Previous surveys

The nature of context has evolved over time, and there are many differing definitions of context in the literature. In order to establish a common basis for analysing context systems, this section characterises context integration based on the comparative analysis of three previous context surveys[Baldauf et al., 2007, Ye et al., 2007, Bolchini et al., 2007]. In addition to the descriptions of systems in these surveys, each one attempts to establish some common, important features of context. It is these features which this section addresses.

A common theme in context research is to draw from an early definition [Abowd et al., 1999],which is extended in [Dey, 2001]. This definition of context can arguably be considered the seminal one, and offers the following concrete definition:

> Context is any information that can be used to characterise the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves.

This definition points to the notion that context is a 'command source' [Bolchini et al., 2007], an external influence which can affect the behaviour of applications. Context data is by its very nature difficult to quantify, and highly variable. There is an inherent need to be able to accommodate a variety of different sources and formats [Ye et al., 2007].

In an effort to make the notion of external, highly dynamic contextual data more concrete, each of the surveys presented has attempted to take on the notion of contextual axes which itemise the ‘key’ [Ye et al., 2007] properties of context. While the specifics differ, there is a common theme which arises from pervasive computing, and which can be summarised by the analysis provided by [Baldauf et al., 2007], where they take the three *entities* described in the definition above, and extrapolate attributes for them. These entities are: places, people and things. The entities are described by their identity, location, status and timing. The other two studies mentioned make the addition of an explicit notion of a task, which can be considered as an event in time with a related goal.

From these entities and attributes, it is possible to derive a notion of how context is formed in the analysis found in many context-aware systems. Context in these systems is a set[1] of data integrated into an application with the intention of provoking changes in the behaviour of the system based on the interpretation of that data.

All three surveys decompose the task of making use of context into the representation of that data and the reasoning of the results. Early systems [Schilit et al., 1994] used simple key:value pairs to represent the location of a user for context. Progressive advances have used object-oriented [Hofer et al., 2003], model-driven [Ceri et al., 2007] and semantic-web [Da and Zhang, 2004] technologies to represent gathered context. One important issue is the separation between gathered data, particularly where sensors are used, and the information which might be used to motivate changes in an application’s behaviour. This distinction is explicitly found in examples in the surveys, and are highly visible in model-driven systems, such as FOCALE [Strassner et al., 2008], where different classes are employed to represent data and information.

[1]in this case, a set of data is considered as a broader class consisting of both unordered data points and sequential data points (as in a process, accounting for the history of the data [Coutaz and Rey, 2002]).

This information provides a basis for the translation of the known facts derived from measuring context into changes in the application: contextual reasoning. Once again, a variety of techniques are reported in the surveys. Ontologies represent a method for collecting facts and establishing rules about them [Ye et al., 2007]. They have the added advantage of granting the system the ability to be extensible and customisable, by allowing the model itself to be manipulated as required [Chen et al., 2004a].

Two of the surveys [Ye et al., 2007, Bolchini et al., 2007] go into particular detail on the analysis of the entities and their attributes, listing desirable properties such as relative time (e.g. ‘Tomorrow’) or relative location (’within 20m of the building’) and the ability to extend the context model to specific cases. One of the objectives of the following sections is to show that the argument for extensible contextual modelling can be extended to the point where explicit axes can be discarded.

2.2.1 Definition of Context Integration

Based on the definitions of context found above, and work in [O’Connor, 2005], the process of integrating context can be defined as:

> The transfer of context information, available from one or more sources, which fulfils an information need in a target application, and which can be translated to a level and transformed to a format understandable by that target.

2.3 Survey of Selected Context Systems

In order to provide a basis for examining a variety of systems, the surveys described above includes some common approaches. In particular, they examine the reasoning capabilities, information models and architecture of the context-aware systems. This survey will examine those properties, and will also attempt to quantify some information about the requirements which each system places on the applications which are the sources of context or the target of the integration.

Three Systems were selected for detailed analysis in this chapter. The systems were selected because they represent well-cited examples of ontology based context integration frameworks. The first two, Cobra and SOCAM, were also selected because they are analysed in detail in the surveys which form the basis for this chapter. The third system, Construct is designed by some of the authors of [Ye et al., 2007], providing an insight into how their analysis was realised.

- **Information Model**: The systems will be analysed to examine the structure of the representation they use for the context information. The principal properties include the type of representation (e.g. semantic, model-driven), a characterisation of the information which is modelled by default (if any), and the extensibility of the representation.
- **Participants**: Context systems are by their nature connected both to sources of context and to applications which gain contextual knowledge. The systems presented in this section will be analysed for how they harvest information and deliver context.
- **Architecture**: The structure of the context system itself will be described in this section. An extensive categorisation already exists [Baldauf et al., 2007], and this will be extended to the systems described.

  Three systems are surveyed in this section: CoBrA & SOUPA, Construct and SOCAM

2.3.1 CoBrA & SOUPA

This section will describe the Context Broker Architecture (CoBrA) and its associated Standard Ontology for Ubiquitous and Pervasive Applications (SOUPA).

General Description

SOUPA [Chen et al., 2004c] represents an attempt to gain the benefits of knowledge sharing and ontological reasoning for pervasive and ubiquitous applications. These applications represent, “a natural extension of the existing computing paradigm”. SOUPA follows the engineering principle of reuse, albeit in an unconventional way.

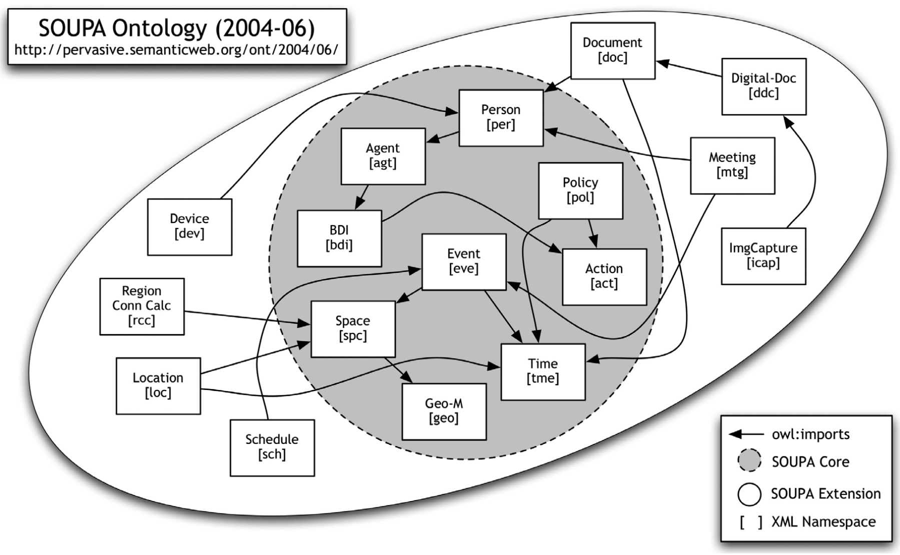


Figure 2.1: The relationship between SOUPA Extension and SOUPA Core, along with the different groupings for entities within the overall model. [Chen et al., 2004c]

One approach to ontology authoring is to import external concepts from other ontologies, where they exist. This allows ontology authors to take advantage of work that has been done previously to standardise and engineer consistent ontologies, while retaining the ability to customise for their own demands.

Instead of taking this approach, the SOUPA authors have created entities in SOUPA which are identical to those found in well-known ontologies, and used the built-in OWL Ontology Mapping operations to express logical equivalence between the SOUPA entities and their ancestors in the original ontologies. The authors indicate that this is in order to improve the performance of the reasoner, by restricting the number of irrelevant entities which would otherwise be loaded if the original ontologies were used [Chen et al., 2004a]. The mapping for the entities for SOUPA has been done during the authoring of the ontology, and is static.

Ontologically, the SOUPA *Core* itself is organised around nine documents, each of which corresponds to entities from external ontologies. These can be loaded as required by the specific application. The *Extensions* are structured to permit both new application domains, and in order to demonstrate a path for expanding SOUPA in general. Each entity in a SOUPA ontology document is prefixed with a short namespace, usually three letters.

Cohesion in the modularity of the SOUPA ontology documents is maintained by employing a chain of OWL imports, where concepts that cross between different documents are referenced from their sources. For example, The Agent Document imports concepts from the Person Document, which itself can import entities from the Document Extension, as required.

Information Model

The SOUPA model is authored in the OWL ontology language. The authors of SOUPA chose OWL[2] in part because it represents the language of choice for the W3C[3] for the Semantic Web [Chen et al., 2004a]. This choice seems to be aligned with the notion that the ability for users to exchange and import an ontology easy is a primary goal, and that OWL in the form of RDF/XML is a medium with a wide audience.

With the addition of appropriate extensions, SOUPA is intended to represent a shared ontology standard for new pervasive and ubiquitous applications. The ontology as presented is divided into two major sections, *Core* and *Extension.* The Core portion of the ontology is intended to support the concepts which are central to pervasive and ubiquitous computing. Concepts within the SOUPA core are grouped into several categories:

1. **Person**: This represents a user's identity, including their name, gender and associations such as user IDs.

2. **Policy & Action**: SOUPA provides for a permit/forbid action enforcement model using description logic.

[2]specifically, OWL-DL, which is compatible with Description Logic reasoners.

[3]World Wide Web Consortium

3. **Agent & BDI**[4]: This category allows the representation of goals, preferences and objectives, particularly with regard to planning. It also includes the facility for expressing conflicts between the entities. In SOUPA, agents can be either natural persons or applications, and this portion of the model can be used to represent the goals and desires of either type of agent.
4. **Time**: Direct representations of points in time are supplemented in SOUPA with expressions for intervals, relative times and ordering relationships for events (e.g. `tme:startsLaterThan`).
5. **Space**: Bi-directional mapping between Coordinates and semantic location. Some of the descriptive notions of location include containment relationships, Geopolitical information and notions of governmental control are also included.
6. **Event**: These represent an entity with spatial and temporal properties, and inherit their characteristics from those sections of SOUPA.

The SOUPA Extension is intended as a framework for adding concepts on to the core ontology as required by specific applications. Some examples provided by the authors include improved notions of scheduling for a meeting application, and a document description schema. Other short descriptions of examples are provided in the literature.

In addition to ontological reasoning, the CoBrA system provides a second tier of reasoning, based on the integration of the Jena [Carroll et al., 2004] reasoner and a rule-based engine, which depending on the version of the system is either a Prolog-like language [Chen et al., 2004a], or a LISP-like one [Chen et al., 2004b]. The need for additional user-defined reasoning is a recurrent theme in semantic context integration.

Participants

The motivating example for CoBrA, which employs a version of SOUPA to broker contextual inference between services, provides a demonstration of the sort of

[4]Belief, Desire, Intention

participants which can be used by SOUPA based frameworks. This example, called EasyMeeting [Chen et al., 2004b], is the result of the integration of the Vigil Pervasive computing environment and the CoBrA system. The Vigil system consists of a set of service management systems, which each manage different categories of services, making them discoverable and available to users depending on certain policies.

The two main participants, systems which exchange contextual knowledge with the broker, are the MajorDemo Agent, which has control over a meeting room's services (heating, lighting, music, etc.) and a Context Sensing agent, which determines the location of users by their personal devices[5]. The capabilities of these personal devices are established through ontological reasoning, for example by the use of inheritance to discover that a particular mobile phone is a member of the class of phones which can use Bluetooth.

The status of users is determined via their personal devices, and the Context Broker reasons the overall state of the meeting room based on the aggregate context information from the sensing agent and other ontological knowledge. The result of this reasoning is manifested in changes to the status of the MajorDemo Agent, which has a variety of states defining different points in a meeting.

In order to exchange information with CoBrA, the Broker requires an ontological description of the knowledge of the application, possibly derived from SOUPA and its extensions. Some of the applications communicate with the Broker using semantic web technologies as a language, while other, heterogeneous participants, which are not semantically aware, can also be integrated.

Architecture

The most important component of the CoBrA architecture is the Broker itself:

> "CoBrA is a broker-centric agent architecture for supporting context-aware systems in smart spaces. Central to the architecture is the presence of a Context Broker, an intelligent agent that runs on a

[5]PDAs, or Mobile Phones

resource-rich stationary computer in the space." [Chen et al., 2004b]

The use of a rules engine points to the need for the Broker to be able to make assertions beyond those afforded by standard ontological reasoning. In the EasyMeeting system, there is admittedly relatively little reasoning possible as the ontologies are restricted to OWL-Lite models. However, even full DL reasoners would be of limited use in some of the reasoning tasks, such as those related to determining the appropriate meeting state.

The architecture of the CoBrA is composed of four principal components, the CoBrA-ONT, which holds ontological knowledge, the Context Inference Engine, which is responsible for the rule base and ontological inference, the Context Acquisition Engine, which creates abstractions for the low-level acquisition of information, and the Module for Privacy Protection, which implements deontic policies to enforce privacy.

2.3.2 Construct

The Construct system was developed as a framework for the design and evaluation of context-aware applications, principally in a pervasive computing environment.

General Description

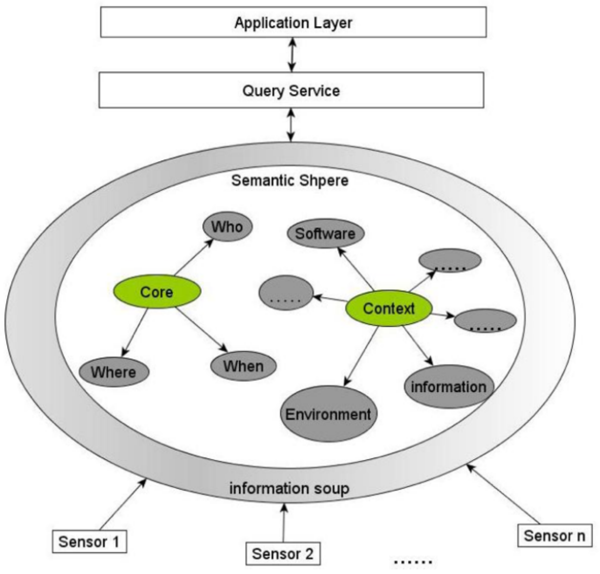
Construct employs semantic web technologies to exchange information, and to perform some of the reasoning in the system. Several services are connected to individual nodes, and interact with them through high-level queries. These high level queries can then be decomposed into lower-level operations, some of which may depend on the ontology.

Different nodes in the Construct network can interact via the Zerconf protocol [Guttman, 2001], which facilitates discovery. The individual nodes maintain independent knowledge, but share overall state by a gossip protocol.

Information Model

Like Cobra-Ont, Construct maintains a 'Core' set of ontologies, extended as required by 'application contexts' [Clear et al., 2006]. All of the ontologies in both categories are authored in OWL. The combined ontologies represent a *semantic sphere*, which is the knowledge space that can be reasoned about by the system. The specific context data is introduced into this sphere through instantiation of classes within the ontologies. This is described by the authors as 'hooking in' context. The Core

Figure 2.2: Information model for Construct, showing the ontologies and application contexts [Clear et al., 2006]



Construct model is composed of three main ontologies:

- **The Where Ontology** provides concepts which represent points in space, regions and notions both of co-ordinate and semantic location. An example hook for this ontology is the co-ordinates reported by a tag worn by a user.
- **The When Ontology** related temporal considerations, such as points of time and intervals as well as notions such s 'Yesterday'. The reasoning of this ontology includes notions of relating different times as before, after or at the same time.
- **The Who Ontology** represents the entity ID of agents ([Ye et al., 2007]),

which can identify uniquely agents and be used to correlate the identities in different ontologies.

The three ontologies presented are grounded in the pervasive area, and represent the 'key' contextual axes. The application contexts are data models which are intended to be written to collaborate with the Core, but are not ontologies themselves, as they are 'too specific' [Clear et al., 2006].

Participants

The participants of the Construct system communicate with the nodes through the ontology languages and their associated query languages. RDQL and SPARQL provide a mechanism for asking questions of the system [Dobson et al., 2007]. Of particular interest is the way that queries are analysed for *known* and *unknown* elements. The known elements are used to construct possible queries to resolve the unknowns, for example by using the id of an agent to find the knowledge held about it in the system. The use of query languages allows for the system to take advantage of transitive and equivalence relationships in the ontologies and data models to resolve differences of conceptual level in the knowledge in the system. This can be used to reason about the correlation between a particular point in space and a building name, for example.

Architecture

The nodes can act together, using the gossip architecture to relay information. For the purposes of this study, it is more interesting to examine the internal features of the nodes. All of the information within the system is held in RDF. Provision is made in the system to transform information which is held within the models into a form which is useful to the application directly. Queries are created in RDQL or SPARQL, and passed to a Query Executer, which has representations of real and virtual sensors. Virtual sensors are logical amalgamations of distinct physical sensors, used in part to resolve abstraction issues, and to help compensate for disagreement between sensors. The Query Service Reasoner is used to resolve, for example, the containment of co-ordinates within rooms, within buildings, and so on. Custom inference is a major component of the reasoning within Construct, in order to provide the proper levels of abstraction and knowledge to fulfil unknowns.

2.3.3 SOCAM

General Description

SOCAM[6] [Gu et al., 2005] describes a set of independent services in which each perform a portion of the functions required to provide contextual information to requesting applications. The authors describe four major functions for the services to perform: acquisition of context, discovery of context, interpretation of contextual knowledge and dissemination of the resulting information.

One of the stated objectives of SOCAM's design is to lower the overhead on a user who wants improved application performance through better integration. This is a common objective for contextual applications.

Information Model

The internal SOCAM model is designed in OWL, and is split into two tiers. The upper tier is composed of five central concepts: Context Entity, Computer Entity, Activity, Location, and Person. The upper concepts all inherit from Context Entity, and themselves form the super-classes for certain mid-level concepts such as, for example, `IndoorSpace` and `Network`. The structure of the information model as an upper, generalised model and application-specific lower ontologies is reminiscent of the upper ontological approach discussed later in this thesis. The authors indicate that one of their primary goals in using this model is to reduce the conceptual overhead associated with the large ontologies needed to describe the wide number of possible contextual sources and concepts.

[6]Service-Oriented Context-Aware Middleware

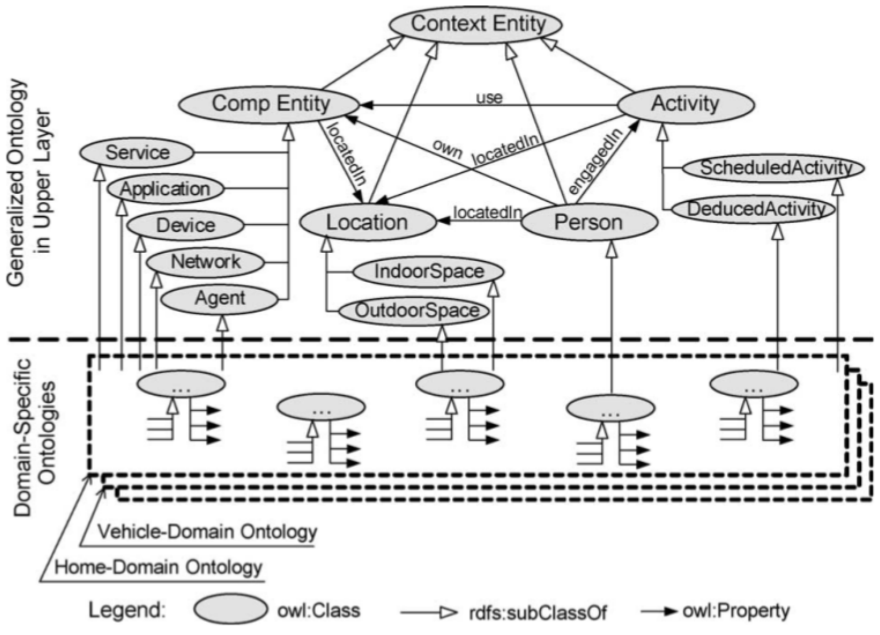


Figure 2.3: Information model for SOCAM, the upper ontological inheritance and links to lower application ontologies. [Gu et al., 2005]

In addition to modelling context conceptually, the SOCAM system represents qualitative classifications for contextual information. The principal delineation is between Direct and Indirect, or inferred, context. Direct context is further specified as coming from sensor information (Sensed) and context which was specified in, for example, a profile (Defined). The nature of these classifications is that they assist in targeting different types of reasoning, and in describing other inherent properties, such as the uncertainty associated with sensor measurements. SOCAM defines a new property relationship for OWL, `owl:classifiedAs` to represent this classification.

Similarly, in order to assist in certain reasoning, SOCAM defined another new element `rdfs:dependsOn` to capture the dependency relationships in the context model so as to be able to generate a Directed Acyclic Graph from the context. This assists in the use of user-defined rules-based reasoning, which supplements the general ontological reasoning in fulfilling application need for context. The need for user-defined reasoning is a recurring one in the design of context systems, and will be discussed below.

Participants

Context Providers are participants in the SOCAM system which gather information from one or more sources and provide appropriate interfaces to allow them to connect their information to the SOCAM ontologies. Providers are registered with a discovery service, which can then instantiate them as required by different queries using the 'Service Locating Service'. Context Providers are said to be internal when they obtain information from a spatially local domain, while they are external if they obtain the information from remote sources.

Context Aware Services represent the participants which request contextual information. Services make their requests to the service discovery registry, and can either listen (push mode) or make explicit requests (pull mode) for contextual information. Triggers for the push mode can be specified using the user-defined rules-based reasoning.

Architecture

Each of the SOCAM services is independent, to the extent that they can be located across separate networks and interact using RMI. The architecture is divided into spatially local domains, which each have a context database to record historical contextual information.

A (possibly remote) context interpreter is used to perform ontological reasoning, principally resolving transitive relationships and model verification, and execute user-defined rule-based reasoning. The reasoning events are created by queries, which are sent to the Service Locating Service, which will contact the appropriate participants and databases based on the service registry which it maintains, and the content of the query.

A key underlying feature of the architecture is the notion of domains, which define the internal or external nature of context sources, and which are part of the nature of the service discovery process.

2.3.4 Analysis of Surveyed Systems

This section will draw together the comparative and individual features of the systems described above in order to identify the best practice amongst context systems. In addition, specific areas where the systems share features and differ will be examined, with the objective of locating the properties which a new system might have and share, and what unfulfilled requirements a new system might benefit from including. The initial analysis will be undertaken under the descriptive headings used to present the system profiles above, followed by further discussion of specific issues.

In discussing the motivation for creating context-aware applications, one recurring theme found in the studies and in the systems presented is the notion that as a user becomes increasingly connected, the burden of managing that information becomes more and more significant. This would appear to be a driving reason which should be considered in the process of designing context systems.

Information models

Initial work in the area of context was in the form of application and source specific solutions, where the objective was to begin to grasp the potential of using external information at all. This gave way to more complex solutions, where several applications might share the same toolkit as a component in their functionality [Dey, 2001], this toolkit format has more recently become an infrastructure, where a variety of applications and sources are connected through an independent software medium [Baldauf et al., 2007].

As the models have become de-coupled from specific applications and sources, designers have increasingly tended towards the meta-modelling approach associated with ontologies to provide flexible representations which can be built up with the foundations of shared ontologies. The use of ontologies appears to be desirable, as it provides general tools for linking information, model consistency checking and some limited reasoning through transitivity and other logical relations.

Experience in the domain of ontology management[7] seems to point to a difficulty arising with the assumption that shared upper ontologies are a viable means for establishing a common basis. There is a tendency, as observed directly in the design of SOUPA, for ontology authors to author their own ontologies, and to adopt concepts rather than referencing them. There are perfectly good reasons for doing this, but the fact remains that the ontologies become difficult to share. Similarly, in SOCAM, the authors went so far as to add several new language features, which are required in order to be able to do some of the inference in the system, once again limiting the reusability of existing ontologies from sources of context, and for applications which could take advantage of the context.

The use of extensions to create application-specific knowledge is present in all three systems. The need to provide these low-level concepts to accommodate the specifics of situations is a recurring difficulty in the systems, and arises both from the difficulty of providing for every possible variation of context, and also due to the computational limits associated with large ontologies. For example, the authors of CoBrA mention the high cost of importing thousands of statements from the full CyC Ontology [Niles and Pease, 2001], and even hint at the use of ontology matching techniques as a possible method for importing foreign concepts.

The result of the analysis of these systems points to the notion that there is a need to be as flexible as possible in the representation of context. In particular, the notion of 'key' axes such as location arises from the pervasive application domain. In generating a future context mediator, it would appear desirable for that system to take advantage of recent developments in ontology alignment and mediation, and furthermore to seek to do so in a manner which takes advantage of pre-existing ontologies from participating systems, rather than trying to create a common ontology that needs to be supplemented with custom extensions to be fully expressive.

[7] *Cf. following chapter*

Participants

The integration of sources of context and applications which can request context is the key feature of a context infrastructure. The objective is to be able to find ways of fulfilling an information need at the abstraction level appropriate to the application, particularly if that information is drawn from several sources at different conceptual levels within the sources of context. In examining the three systems, it is clear that ontologies are a useful part of the solution to finding different levels of abstraction, but that they do not fulfil all of the requirements.

Increasingly, there is a trend towards applications exposing APIs, and to a much smaller extent, ontologies describing their knowledge. This trend would appear to support the notion that it is desirable for participants' own ontologies to be usable in the information exchange of context. This is all the more important as participants become more sophisticated, and where their dependence on context information forms only a portion of their overall functionality. The notion that applications might be optionally enriched with context, rather than fully dependent on it, arises as the application domain moves away form strictly pervasive and into other settings, or when the applications have a rich, existing model (a large number of knowns) which can be improved with additional context. This evolution mirrors the process described above where applications become less dependent on specifics and more able to take advantage of enrichment of their knowledge.

In these systems, the specification of need for the user's application arises from queries formed by that application. For example, in construct this is achieved through RDQL or SPARQL. This allows the systems to pull information from sources, so long as they are aware of those concepts.

The functionality of the agents in CoBrA and the nodes in Construct point to the idea that the context infrastructure might not necessarily have to deal with sensors directly, but might be able to employ a remote or local filtering process to create ontologically, or at least schematically compliant models. This notion of abstracting the low levels of some inputs would appear to be a desirable design decision to make, as it makes the process of designing both ontological and user-defined inference more tractable. This concern would be all the more significant in the event that a model without *a-priori* concepts were to be attempted, as suggested above.

One recurring property of the systems is the notion that context must be queried for, or listened for. This notion allows for applications to make some domain specific queries, but it might be possible in a new system to consider a different mechanism for expressing information need to the context infrastructure, and thereby open the possibility of the reasoning in the mediator being used to effect more subtle changes on the knowledge of the system.

Architectures

It appears that the dominant model for designing context systems is through infrastructural approaches [Baldauf et al., 2007]. The use of a set of services, whether loosely coupled as in SOCAM, or bound as an integral system as in Construct, embody the key functionality of a context system, which is to gain a broader knowledge of the events, properties and preferences in a situation than any single participant can have. In effect, the mediator gains a comprehensive view of the context, and can offer that information to the user by means of their agents at an appropriate form for those agents.

There is broad consensus that the use of ontological reasoning alone is not sufficient for the execution of most context tasks. Ontological techniques provide a useful means for standardising the definition and some of the relationships associated with concepts, but reasoners need to be supplemented with additional tools for the design and expression of relationships between concepts. The nature of context is that it can vary between users and between situations, and so it is important to have an idea that there should be a way to express the different relationships between knowledge. This expression of a collective view needs to be supplemented with the ability of the context infrastructure to determine which information is relevant to a particular participant, and to provide it in a way that it can be useful to that participant.

In summary, the architectural task of a context infrastructure is to be able to **identify** the information need of the application, **semantically translate** it from the representations within the knowledge of the sources of context, and **syntactically transform** the new information so that it conforms with the structures usable by the application to be enriched. This process will form the basis for the focus of the work in this thesis.

2.4 Conclusions & Summary

This chapter has presented a portion of the wide base of research and literature into context-aware systems and context research. The key trends which have been found from this analysis are that context is increasingly being provided as an infrastructural offering, from some mediator with a comprehensive view of the situation surrounding the user and their agents. As the design process has matured, and as service-oriented and semantic techniques have become more widespread in their use, this points to the notion that applications which participate in the context enrichment process might themselves have complex information structures, and will need a new method for accommodating them.

The use of ontological techniques has previously been on the basis of establishing common or upper ontologies, but as the domain evolves away from location-centric pervasive scenarios to slightly broader information scenarios, there establishment of a common set of concepts becomes a possible obstacle. This, combined with the advances in ontological mediation as a means for transferring information between different parties, points towards a possible design route for a new system.

2.4.1 Summary of findings

There are several key findings that result from this survey, which indicate what the best practice for Context systems is. These features are summarised below:

- There is a general trend across many of the systems surveyed that ontological representation of context provides an effective means for dealing with the high degree of heterogeneity in context information.

- The use of a context mediator, which is a separate system from the sources of context and the target application is desirable because such a mediator can be designed to support a variety of participants flexibly.
- The effort of integrating context can be shifted from the Target application to the context mediator, which can effect change on the behaviour of the target application by making alterations to models which represent the knowledge of the target application. These model-based exchanges replace the query-oriented approach of other systems.

One issue which is not clearly addressed in these systems is the problems associated with using a pre-existing, integrated model of context in the context mediator. Creating a model for context requires the designers of the model to consider what modelled features to include in context. This has implications both for the size of the model (more general models need larger numbers of concepts), and the range of possible context which can be included, both in terms of the type of information and its behaviour. The main approach to solving this challenge in the work surveyed in this chapter has been to provide for specific extensions to the context model. However, there are several difficulties with this approach: first, it reduces the value of having a common model by requiring systems to interoperate with specific extensions, and secondly it further increases the size and complexity of the resulting shared model.

The system developed in this thesis will include a novel approach to modelling context, which creates a composite model of context derived from the concepts in each of the participants' ontologies, linked through semantic interoperation. This allows for a context model which depends on fewer assumptions about what constitutes context, and which focuses the interoperation effort on the concepts which are directly used in the system. Of particular interest is the notion that a context mediator might be able to push knowledge as well as data to an application. This means that the mediator identifies concepts which could be used by the application, and adds them to the schema for use by the application. This system can therefore be viewed as providing a meta-model for facilitating context integration and knowledge exchange.

Chapter 3

Ontology Mapping & Mediation

This Chapter discusses the state of the art in the different stages of semantic interoperation. The Chapter begins with a description of the terminology which will be used to describe the interoperation process. This is followed by an analysis of ontology mapping tools, which are the first stage in finding links between ontologies. This is followed by a discussion of the representation of those mappings. Finally, some key ontology mediators are discussed. The analyses from each of these surveys is then summarised to create a list of key properties for an effective ontology mediator.

3.1 Introduction

This chapter examines the literature surrounding semantic interoperation with a view towards understanding where alignment and mapping can be used to exchange information. The objective of this chapter is to find the strengths and weaknesses of previous approaches to alignment, as well as to locate the lessons from general ontological mapping which can be applied to the context domain, as defined in the previous chapter.

The first section of the chapter provides a set of terminology for use in the remainder of the thesis when discussing ontology matching and mapping. The nature of ontology mapping is examined, with the objective of addressing some of the possible relationships that can be established between concepts in different ontologies.

There are many examples of ontology-based applications. However, there are relatively few examples of systems which perform Ontology Mediation in the specific form which is envisaged by this work. In order to provide sufficient basis for analysis, this chapter examines the underlying technologies associated with ontology mapping, the expression of that mapping, and some techniques related to mediation which fulfil similar requirements. These are used to draw some conclusions about the important properties of a suitable ontology mediator.

3.1.1 A Note on Terminology

There is a wide variety of terms is employed in the literature to designate different processes and procedures for relating ontologies. One survey [Kalfoglou and Schorlemmer, 2003] mentions six different terms, and notes that the delineation between them is at best uncertain.

In their paper, the authors of [Kalfoglou and Schorlemmer, 2003] define ontology **mapping** as the process of relating symbols and axioms in two ontologies which share the same 'domain of discourse'. This process generates a mapping, which is a functional description that can be used to assign one ontology's symbols to another.

The authors of [Choi et al., 2006] classify ontology mapping into local and global as well as examining mapping for the purposes of merging. Local ontologies represent those which have been authored independently and whose entities do not share a common reference. This is in contrast to ontologies which are authored with reference to a 'global' ontology, also known as an upper ontology, which provides a reference point for mappings, and can be construed as a pre-determined articulation.

The authors of [Kalfoglou and Schorlemmer, 2003] define ontology **alignment** as the process of using relations taken from an external ontology to describe the links between the symbols in the ontologies. The **articulation** of two ontologies, describes and defines the properties of the relations used to express the alignment of the ontologies. The authors describe several uses for an articulation, such as merging and translation, but the survey in this chapter focuses on the direct use of articulations for alignment and mediation.

Similarly, Choi *et. al.* [Choi et al., 2006] define ontological alignment as 'creating links between two original ontologies'. This definition preserves the key feature of alignment, which is that it provides a way to express links between different ontologies which may not have originally been designed to be compatible.

This thesis employs the terminology and definitions from these two surveys in the analysis of ontological mapping and alignment. In particular, this work employs the precise terminology from [Kalfoglou and Schorlemmer, 2003], and borrows the distinctions in mapping types from [Choi et al., 2006]. The focus of this work is on supporting information exchange between ontologies, particularly based on the requirements that arise from the context domain.

Additionally, for the work in this thesis, **Ontology Mediation** is defined as the process of loading and aligning ontologies with the objective of transferring knowledge and information from a set of source ontologies to a target. Unlike ontology merging, the mediated ontologies remain separate.[1]

3.2 What needs to be mapped?

This study focuses on three areas in which ontologies can differ. Different mapping processes handle each to different degrees. The three areas are:

1. **Syntactic**: The first, and most obvious difference in ontologies can arise where they are defined using different representational languages. This can have far-reaching implications, particularly where there is a marked difference in approach to representing the ontology entities and axioms.

   An example of this might be as simple as the difference between what is

[1]This process is distinct from other forms of ontological application, such as ontology-*based* mediation, where the ontology forms a platform for mediation between non-ontological knowledge sources.

valid in different OWL dialects. For example, an assertion of transitivity for a datatype property, which is valid in OWL-Full, will not be valid in OWL-DL[McGuinness and van Harmelen, 2002]. There is particular difficulty in attempting to recognise, let alone preserve, the implications of syntactic differences between different ontology languages and dialects.

2. **Structural**: Even where syntactic differences are accounted for, there can be structural mismatches between ontologies. In many ontology languages, there is considerable variation in the engineering processes by which ontologies are constructed, and this can lead to radical differences between ontologies representing the same things, even when they are written in the same language.

   Structural mismatches occur for example where, in OWL, a concept is represented by a Class in the source ontology and by an instance in a target ontology. Such differences can have wide-ranging implications for how information is transported from one ontology to another. Other examples include differences in the division of concepts, where partially overlapping concepts arise in different ontologies.

3. **Semantic**: Once structural and syntactic complications have been overcome, it is still necessary to realise that, in most cases, ontologies do not specify completely the semantics of their individuals. This can create a critical semantic drift, where seemingly-identical concepts are not able to exchange data because of undocumented representational differences.

   One typical example of a semantic mismatch that can introduce a deceptive error is to imagine a on ontology with a `Car` concept. In one severe case, the same ontology schema is used in both the target and the source, one by a European and one by an American designer. In both cases, the ontology represents a `Fuel Efficiency` value. The Semantic mismatch can occur where the European designer inputs the value (which is an undecorated Integer) as Litres / Kilometre, while the American designer interprets this value as Miles / Gallon. While this specific problem might be addressed by annotating the value with units, but there is little guarantee in the case of ontologies which are authored independently.

It is highly unlikely that automated mapping processes will be able to examine these complications, except in certain rare cases, or where the ontology is designed with an unusual level of explicitness. There are numerous similar examples, many of which are difficult even for humans to interpret. This reflects directly the challenge of finding agreement between different semantic representations: it is dependent on the parameters of the situation which makes the mapping relevant.

This view has been independently arrived at in literature by other authors, for example with concepts such as Semantic Distance, as seen in [Albertoni and De Martino, 2008]

3.3 Survey of Ontology Mapping

From the terminology adopted above, ontology mapping refers specifically to the process of establishing a set of functions for relating the symbols and axioms contained in two ontologies that have some relationship[2].

This section examines a number of mapping techniques, which feed the alignment process. The importance of these techniques is that the form the input to the alignment and mediation process, and so represent the kinds of operations which need to be performed in order to permit mediation to happen. In examining these techniques, the objective is to understand & contrast their properties, rather than a direct comparative evaluation of their merits. With this in mind, the survey is not limited to individual mapping systems, but includes frameworks for addressing the mapping problem as a whole, for example OISIN [O'Sullivan, 2006].

1. **Brief description of the technology**: a short overview of the nature of the

[2]there is little value apparent in attempting to align ontologies which do not share any overlap — the null case

mapping process, whether it requires populated[3] ontologies or not, and how it approaches different alignment cases.

2. **Prerequisites for mapping** specifies what information is required to perform the mappings, such as the format of the ontologies, and any other requirements. This includes a description of mappings, which specifies what metadata can be established by the mapping process. A description characterises information such as the kind of relationships, the confidence interval and any other information described by the process, as applicable.

3. An **Example** is provided in order to illustrate the format that is output from the system, or to describe the nature of the process that the mapping system employs

In this section, a representative technology has been chosen to demonstrate some of the different approaches to the mapping problem. They are chosen as indicative of the approach to ontology mapping which they take. Other systems with similar properties are discussed as required.

3.3.1 COMA++

The COMA++ system [Aumueller et al., 2005] is representative of a class of systems which are designed to enable the mapping of ontologies using one or more matching techniques. COMA++ is an extension of the original COMA system, and is similar in that it presents a method of iteratively supporting schema mapping using a variety of matching techniques.

The architectures of COMA and COMA++ have several elements that allow different matching techniques to be applied automatically, and with manual intervention to generate the mappings. In addition, mappings can be updated and edited by the user, as well as combined from different matching sources to improve the value of mappings, or to generate mappings based on agreement between different techniques.

[3]with instances

A *similarity cube* is employed, consisting of the set of measures of confidence in the match, and references to the matched entities for different matching techniques. This cube is used in the process of assessing the combination and validation of mappings at the review phase.

COMA++ provides for a variety of mapping strategies, which can be employed in different scenarios depending on the nature of the ontologies and their mappings. These strategies include support for partial mappings of large ontologies, called *Fragment-based Matching.* Fragment-based matching is intended to provide a piece-wise approach to matching the overall ontology, by applying the composite matching techniques at a fragment, rather than overall schema level.

The second strategy related specifically to the reuse of mappings. Mappings discovered by the automatic and manual processes can be retrieved later for reuse in new mapping scenarios. One instance where this can be employed is in the process of using a pivot schema - an intermediary schema to which matches for both schemas has already been found, creating an indirect, transitive mapping.

The result of sometimes highly complex mapping processes using COMA++ is an external description of mappings. Mapping descriptions are exportable either in an XML/RDF format or in CSV [Aumueller et al., 2005]. These can then be used in other COMA++ processes, or the mappings can be employed for alignment, merger or mediation processes.

Prerequisites for Mapping

COMA++ supports a variety of traditional schema description languages, such as SQL. For ontologies, COMA++ is compatible with OWL-Lite. Schemas written in different languages are converted to an internal acyclic, directed-graph form and stored in a relational database. This graph form is used for the basis of a variety of schema and instance [Engmann and Massmann, 2007] matching approaches. In each case, mappings are reviewed by users and updated.

The system architecture allows for extensions to be made to add a variety of techniques to establish mappings, in order to provide a framework for combining and evaluating different matching techniques for different models. The nature of the mapping is dependent on the matcher-specific representation. The confidence interval of the match is presented on a scale from 0.0 to 1.0.

The format of schemas and ontologies that can be read by the system depends on the parsers which are attached. This also has an effect on whether the schema- or instance-level matchers are employed [Engmann and Massmann, 2007].

COMA++ is principally a framework for aggregating the result of different ontology matches. Because of this, it does not express mappings in full, nor does it provide an ontology of mapping relations. This means that the result of COMA++ is not a complete articulation, and therefore an incomplete mapping description, which needs to be correlated with an articulation that describes specific relations.

However, the description of this system is important, as it represents a widely-followed approach to generating mappings based on automated and semi-automated matchers. Other examples of systems include Falcon-AO [Jian et al., 2005] and other participants in the Ontology Alignment Evaluation Initiative [Euzenat et al., 2007].

Example

As this system does not produce complete mappings, the description below is a representation of the hybrid matching process as found in the COMA++ system, it is taken from [Aumueller et al., 2005]

3.3.2 OISIN

OISIN[4] [O'Sullivan, 2006] represents a complete mapping framework, including discovery of ontologies, establishment of mappings, and management of the mapping representation. While complete tool support does not exist for OISIN, the design of the process does point to several important considerations for mediation, such as the

[4]Ontology Interoperability in Support of Semantic Interoperability

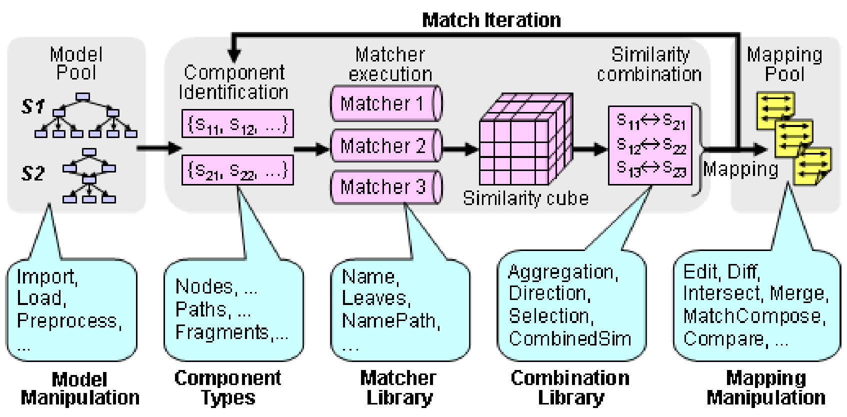


Figure 3.1: COMA++ Match Processing

need to understand that different types of relationship between various ontologies might benefit from the use of different techniques.

The OISIN process begins with the characterisation of the relationship between the ontologies. This includes examining both the modelling and semantic characteristics of the two ontologies, and results in a decision as to whether the ontologies are suitable for match analysis. These decision points allow the OISIN process to discard 'incompatible or low-quality' candidates at an early stage.

Once the decision to match has been made, the next process is to decide upon matching algorithms, and execute the matching process. This results in a measure of *amenability*, which determines the matching processes that are suitable for application to the ontologies in question.

The mapping process supports either canonical expert mapping or usage-oriented mappings by establishing the concept of a *committed mapping*, one that has been reviewed by a relevant human and approved as correct.

The committed mappings represent indications of high-confidence relationships between entities, and can form the basis for further discovery, as well as acting as evidence in reasoning.

Prerequisites for Mapping

Citing its growing prevalence, and noting that the technique offered can be generalised to other formats, the OISIN framework is principally focused on the use of OWL as a representative ontology language. One difficulty encountered is that the output of different matching systems has not been standardised, and there are a number of competing models for expressing matches and mappings (*c.f.* following section).

At different stages of the process, different sets of reports and statistics are generated for the user, such as characterisation and amenability measures. These are combined with user-defined anchor mappings to generate a query-able interface (based in XQuery) that can retrieve mapped entities.

There is also provision in the OISIN framework for exporting the mappings in a variety of formats, including as OWL statements.

3.3.3 SUMO

The Suggested Upper-Model Ontology [Niles and Pease, 2001] is representative of the 'global ontology' approach to ontology mapping. In this approach, ontological concepts are inter-related through a conceptual hierarchy that extends above that defined in a particular ontology. This means that when a concept is instantiated in a low-level ontology that expresses specific semantics, it is defined in terms of the upper level concepts in higher ontologies. By creating a chain of definition up to a shared, abstract upper ontology, it is possible to take advantage of relations defined in the upper ontology, and employ them to establish mappings between different ontologies. This has the effect of allowing for the creation of complete, vertically-integrated mid-&-low-level ontologies by inheriting concepts from a common upper layer.

The uppermost levels of the SUMO ontology arise from a merger between the Sowa [Sowa, 2000] and Russell & Norvig [Russell and Norvig, 1995] upper ontologies. In addition to finding common concepts and linking them, a number of highly abstract upper concepts were removed.

The general concepts found in this uppermost ontology include concepts like `Physical`, representing a physical entity, which has child concepts of `Object` and `Process`. These concepts are highly abstract, but different mid- and low- level ontologies can represent specific domain concepts, such as the *Mortgage* concept from the Finance Ontology.

In order to address this, the concept of the MILO[5] [Niles and Pease, 2001] was introduced, which provides a set of intermediary concepts from the highly-abstract upper ontology. The Mid-Level Ontology can be further supplemented with domain-specific ontologies that describe broad application domains (such as Economy, or Military Personnel). Lower level ontologies can be imported individually, as required, for the definition of specific ontologies.

Prerequisites for Mapping

By aggregating a wide array of general ontologies into a standard merged model, and then supplementing this model with mid-level and domain ontologies, the objective of SUMO was to support the authoring of new ontologies based on those concepts, rather than necessarily attempting to map existing ontologies using SUMO.

This usage model is reinforced by the fact that SUMO and its derivatives are written in a specific first-order description language, called SUO-KIF [Niles and Pease, 2001]. This language allows for rich semantic descriptions of the concept hierarchy and instances of each entity in the ontology.

One effort to combat the complexity of approaching the large and diverse set of concepts has been to create a set of mappings to the WordNet lexicon [Niles and Pease, 2003]. This is intended to present a somewhat more user-friendly method for searching for relevant concepts.

While a mapping to RDF does exist for SUMO [Niles and Pease, 2004], it is not recommended[6]. This does not seem to encourage the use of OWL or RDF in

[5]MILO: Mid-level Ontology.

[6]The information header describes it as 'very lossy'

SUMO-ontology authoring, nor does it encourage the creation of mappings between existing RDF ontologies and SUMO.

Example

```
(subclass Paragraph Text)
(documentation Paragraph EnglishLanguage "A
&Text which consists of one or more sentences,
begins with an indented line, and expresses a single topic.")
(=>
  (instance ?T Paragraph)
  (exists (?S)
    (and
      (instance ?S Sentence)
      (part ?S ?T))))
```

Figure 3.2: Example of SUO-KIF from the Mid-Level Ontology [Niles and Pease, 2001].

The SUO-KIF language allows the ontology author to define instance, subclass, property and domain information for concepts and sub-concepts. Different ontologies can have different dependency sets, which arise from the choice of enveloping concept. The example above, fig. 3.2, declares the Paragraph Subclass of Text, provides an English-language description and then defines the requirement that a Paragraph be part of a Text and contain instances of sentences.

3.3.4 Business Maps

Business Maps represent an approach to establishing an articulation by means of a Topic Map. One of the key elements of this approach is that the maps are established on a bespoke basis: they are mid-level ontologies which are focused on describing the semantic relationships between the concepts. In that sense the maps produced resemble a hybrid between an upper ontology and a mapping expression (see below).

Topic Maps are an ISO standard technology designed to represent a semantic network [Pepper, 2000]. Topic Maps are designed with several uses in mind, but in this case the most important are [JTC1:SC34, 2002]:

- *To qualify the content and/or data contained in information objects as topics to enable navigational tools such as indexes, cross-references, citation systems, or glossaries.*
- *To link topics together in such a way as to enable navigation between them. This capability can be used for virtual document assembly, and for creating thesaurus-like interfaces to corpora, knowledge bases, etc.*

In terms of these objectives, the Topic Map can be said to be a virtual document which qualifies and associated concepts contained in the participants' knowledge bases.

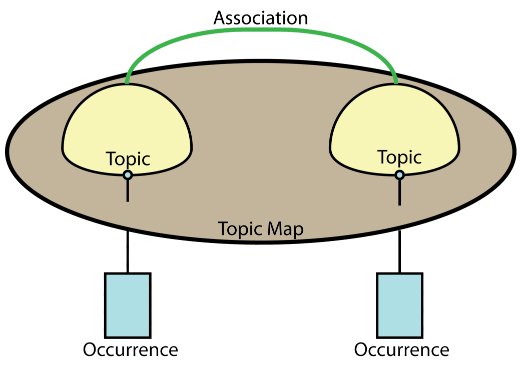


Figure 3.3: Basic Topic Map Elements, the Topics are linked by an Association, and have external Occurrences addressed in other documents.

There are three basic components of a Topic Map:

- **Topics**: Represent conceptual entities in the map. These generally have a property which points to the entity which the Topic Represents. Topics can have labels, which are readable names for the topic; they can also have types, which categorise the Topic based on another Topic in the Map. In this way, it is possible to create closed world representations of concepts and their categorisations in a Topic Map.

- **Associations**: Allow two or more topics to be linked conceptually. Topics can be given role types expressing their role in the association, and the Association can be given one type, representing its nature.

- **Occurrences**: Are references to reifications of the Topic Concepts. The typical example is a URL which points to a resource that is represented by a topic. Occurrences can be typed.

The intention of Business Maps, and similar technologies, is to model the mappings between different ontologies semantically, as a Topic Map [de Graauw, 2002]. This is similar to the creation of an upper ontology, like SUMO, but differs in the sense that it is intended to link pre-existing ontologies. This structure allows Topics to represent entities in different ontologies without being bound to semantic restrictions from within the ontology languages.

The Business Map is very well represented by the notion of an articulation. Topic Maps can be viewed as an ontology technology, with natural facilities for representing the linkages to the mapped ontological concepts. However, in their basic form, Topic Maps are not amenable to ontology-style reasoning, this requires an extension to assert rules about the Map.

One useful property of Topic Maps that can be exploited in this area is that Topic Maps can be merged. That means that where partial mappings have been made between different ontologies, these can be amalgamated to create a broader mapping view. This has the potential to be extremely useful where different matching methods produce different parts of an overall mapping.

Prerequisites for Mapping

In this model, where the Topic Map is established as part of the mapping between the ontologies, the prerequisites for mappings are two-fold. The first requirement is a set of ontologies suitable for mapping. These do not need to be in any particular format or language, so long as they can be processed, and their entities reflected as instances within the Topic Map structure.

The second requirement in this approach is the structure of the map itself. In the Business Maps, this reflects the domain of B2B[7] information exchange [de Graauw, 2002]. In the example, Business Maps are used to represent vocabulary and other differences between different business modelling languages. This imposes the requirement to author an ontology that maps the two schemas, and represents the sorts of mappings required.

Example

The example in fig. 3.3.4 shows the syntax for declaring an association in a Business Map Topic Map. The Association is an instance of a unidirectional mapping. It defines the source and destination roles of the mapping, and constrains the scope of the association to the 'Gigasellers', 'sales' and 'europe' topics.

3.3.5 Analysis of Mapping Frameworks & Implications for Mediation

From the systems presented so far in this chapter, a number of conclusions can be drawn about the complete workflow required to deliver ontological mediation. It is important to understand that there is considerable variation in different technologies, in their approach to mediation and in the delineation between different parties in the process. The systems will be described as presented in the literature, and then attempt to isolate and align the components from each system, so as to obtain a comparable set of features.

The analysis of technologies that exist earlier in the semantic integration pipeline yields some indication of the properties and requirements that a mediation system will

[7]Business to Business

```
<association>
  <instanceOf>
    <topicRef xlink:href="itm.xtm#unidirectional_mapping"/>
  </instanceOf>
  <scope>
    <topicRef xlink:href="context.xtm#gigasellers"/>
    <topicRef xlink:href="context.xtm#sales"/>
    <topicRef xlink:href="context.xtm#europe"/>
  </scope>
  <member>
    <roleSpec>
      <topicRef xlink:href="itm.xtm#source_item"/>
    </roleSpec>
    <topicRef xlink:href="bizwords.xtm#name"/>
  </member>
  <member>
    <roleSpec>
      <topicRef xlink:href="itm.xtm#destination_item"/>
    </roleSpec>
    <topicRef xlink:href="gbl.xtm#name"/>
  </member>
</association>
```

Figure 3.4: An example Business Map Association, taken from [de Graauw, 2002]

have. From the analysis of the mapping tools, there is evidence from the COMA++ and OISIN studies that the nature of the ontologies has considerable impact on the different sorts of techniques that can be used.

3.3.6 Modelling the Relationships between Ontologies

From the systems described in that section, it is also clear that there is considerable variation in the nature of the mappings that will be reported by apparently similar systems. The output of different processes results in different levels of information with different applicability to mediation. In the case where the ontologies are being authored together, then upper ontological techniques such as those demonstrated in SUMO are able to provide tight, definitive information about how different entities interact.

However, this integrated approach comes at a cost. In particular, it is unclear if ontologies authored without reference to the relevant MILO and SUMO upper ontologies can be made to conform to those ontologies without considerable revision. In addition, the use of SUO-KIF as a format for specifying ontologies presents a possible restriction, given the apparent popularity of other technologies, such as RDF.

In effect, the links between SUMO-derived ontologies are implicit, and are modelled within the domain language. This means that the relationships appear as part of the definition of the entities, and is intrinsic to them.

In contrast, the majority of the mapping expression formats attempt to model the properties of the relationships between the ontologies by establishing or, in the case of Align, providing a way to express the relationships which the articulation ontology contains.

3.4 Survey of Mapping Expression

This section will provide a survey of the formats and methods available for defining an articulation. In particular, it will present descriptions of the ontologies from which relations can be drawn, and examine how those expressions are represented. Each of the systems described below contributes a different aspect to the survey:

- The mapping capabilities of the core **OWL** language demonstrate the baseline challenge of representing inter-ontology agreement from within the ontology description.
- **C-OWL** is of key interest because it was applied in the DRAGO mediator

(see below). In addition, it represents an extension to ontology languages to create a formal description of the relationship between ontologies.

- **Align RDF** was chosen because it is a common format for output from many of the mapping tools, and is the standard representation for some evaluation tasks.
- **SWRL** was chosen because it includes features relating specifically to data manipulation and functional mapping.

The significance of these formats is that many of the tools used to generate ontological mappings are either independent frameworks or complete off-line processes [Kalfoglou and Schorlemmer, 2003]. This means that they are not themselves concerned with what use is made the mappings which they discover. Several interchange formats have been devised, which express different amounts of information about the mappings depending on the relationships defined by the articulation.

From the literature, some common attributes arise. The mapping expressions need to be able to:

1. express simple differences such as label differences
2. bridge structural mismatches, such as property - to - class mappings
3. bridge semantic mismatches, such as subsumption and disjointness
4. describe mappings, with mapping confidence metrics and other information

3.4.1 OWL support for Mapping Expression

The Semantic Web Ontology Language, OWL, makes certain affordances for expressing basic Alignments[Hughes and Ashpole, 2004]. This represents a useful baseline for the features which are found in other formats. The principal operation granted by OWL is the `owl:SameAs` relationship, which can be used to represent equivalence relationships between OWL properties and classes. However, this relationship is extremely limited as it only expresses a logical equivalence, and does not allow for any semantic drift[8]. The exact nature of this relationship can be discovered from the OWL reference description [Bechhofer et al., 2004]:

> The `owl:sameAs` statements are often used in defining mappings between ontologies. It is unrealistic to assume everybody will use the same name to refer to individuals.

In effect, the objective is to account for small drifts such as labels, rather than more complex structural mismatches. It is clear that these provisions do not make allowance for bridging complex relationships. In practice relationships are restricted to the same type using `owl:sameClassAs` and `owl:samePropertyAs`. This means that there is considerable limitation as to the sorts of concepts that can be linked. Within this restriction, there is limited scope to represent disjointness, and subsumption is limited to being represented by the canonical subclass relationship.

3.4.2 C-OWL

The authors of this technology present a specific definition for ontological *context*s [Guha, 1991][9], which are local, highly individual views of a domain [Guha, 1991].

The objective of C-OWL is to provide an OWL-like means to represent the links between a C-OWL context, which is not share-able, and an ontology, which can be shared. These relationships are expressed in terms of a set of *bridge rules*.

C-OWL defined five operations which can relate ontological concepts, these are:

1. Equivalence, denoted by $\equiv$
2. Disjointness, denoted by $\perp$

[8]Semantic Drift can be defined as the distance between two concepts that arises between two representations of similar concepts.

[9]in the interests of clarity, the term 'C-OWL contexts' will be used to refer to the local models described in the C-OWL literature[Bouquet et al., 2003]. This is in order to maintain a distinction with the notion of context described previously

3. Overlapping, denoted by $*$
4. more specific (inverse subsumption), denoted by $\sqsupseteq$
5. more general (subsumption), denote by $\sqsubseteq$

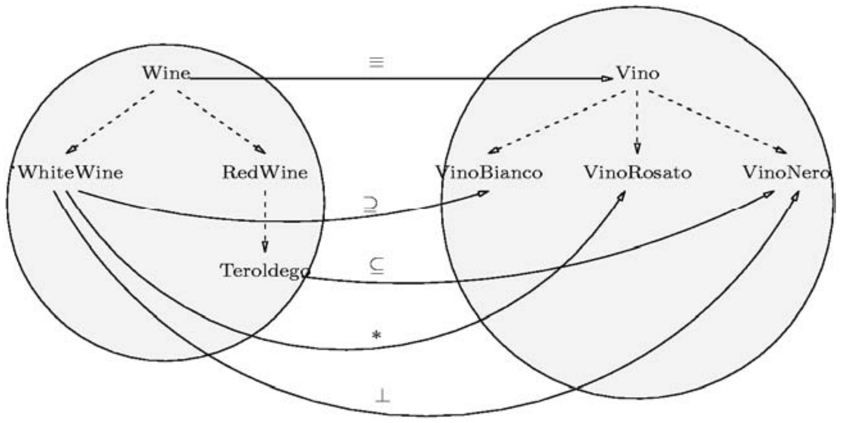
Directionality is an important feature of bridge rules, as many of the rules defined are not symmetric.

The format of C-OWL reflects a modification of core OWL semantics. C-OWL is, to some extent, language independent, in that employs URIs to refer to the source and target of a bridge rule. This means that the actual format representing the source and target ontologies can vary so long as it is addressable within C-OWL. In practice, however, the stringent definitions of the bridge rule operations mean that there is little room for expressing highly disjoint structural mismatches, many of these operations are designed to facilitate ontological reasoning.

While there is no format for describing the properties of the link beyond the bridge rules, this detailed description carries considerable semantics about the nature of the relationship. However, C-OWL resembles conventional OWL links in the sense that they are once more logical associations, and tightly defined. There is also no facility for representing confidence or other properties.

Example Syntax

The syntax shown in *fig.*3.5 corresponds to the equivalence relationship described by the arrow that travels from 'wine' to 'vino'.



```
<cowl:mapping>
 <rdfs:comment>Example of a mapping of wine into vino</rdfs:comment>
 <cowl:sourceOntology rdf:resource="http://onto1/wine.owl"/>
 <cowl:targetOntology rdf:resource="http://onto2/vino.owl"/>
 <cowl:bridgRule cowl:br-type="equiv">
  <cowl:sourceConcept rdf:resource="http://onto1/wine.owl#wine"/>
  <cowl:targedConcept rdf:resource="http://onto2/vino.owl#vino"/>
 </cowl:bridgeRule>
 </cowl:mapping>
```

Figure 3.5: C-OWL Wine Ontology Bridging. This example is a partial reproduction from one found in [Bouquet et al., 2003]

3.4.3 INRIA Align RDF

This format presents different levels which define the the relation expressions and language-independence of the alignment[Euzenat and Schvaiko, 2007].
In particular, levels 0 and 1 are mostly language-independent. The main difference is that the level 1 alignments align lists of entities rather than pairs. Level 2 provides for extensive, language-specific embedding of descriptions, and in this case the relationships require language-specific knowledge to be used.

The Align format is based in the RDF syntax, but is loosely coupled to the standard. The `rdf:resource` entity is used to represent concepts in alignment `Cells`. This permits addressable entities to be mapped without regard to the differences in structure or language, either by bypassing this issue in levels 0 and 1, or by expressing them explicitly in level 2. This bridging is more language-independent than in C-OWL because relations are not defined by the standard.

Provision is made in the ALIGN format for link metadata to be expressed. This includes information such as relationship confidence (for example, as a value from 0.0 to 1.0, representing the confidence of the matches that produced the mapping).

Align represents an extremely light method of expressing mappings, which is unburdened by use (this is in particular contrast to languages such as SWRL[10], which express mappings as a set of rules that can encode the purpose of the alignment, such as ontology merging).

Example Syntax

This example in *fig.*3.6 is a partial reproduction of an alignment from the Gold Standard alignments published with the OAEI[11] 2007 benchmark [Euzenat et al., 2007]. Note that entity references have been shortened to improve presentation. It describes the equivalence relationship with confidence value 1 between the *title* and *hasTitle* concepts.

[10]Semantic Web Rules Language

[11]Ontology Alignment Evaluation Initiative

```
<rdf:RDF>
 <Alignment>
 <xml>yes</xml>
 <level>0</level>
 <type>11</type>
 <onto1>
  http://oaei.ontologymatching.org/2007/benchmarks/101/onto.rdf
 </onto1>
 <onto2>
  http://oaei.ontologymatching.org/2007/benchmarks/301/onto.rdf
 </onto2>
 <uri1>
  http://101/onto.rdf
 </uri1>
 <uri2>
  http://301/onto.rdf
 </uri2>
 <map>
  <Cell>
   <entity1 rdf:resource="http://101/onto.rdf#title"/>
   <entity2 rdf:resource="http://301/onto.rdf#hasTitle"/>
   <measure rdf:datatype="http://www.w3.org/2001/XMLSchema#float">1.0</measure>
   <relation>=</relation>
  </Cell>
 </map>
 </Alignment>
</rdf>
```

Figure 3.6: Align RDF Example Syntax [Euzenat et al., 2007].

3.4.4 Semantic Web Rule Language

SWRL inherits from both OWL and RuleML [Horrocks et al., 2004] in order to create a language for expressing the merger of two OWL ontologies. SWRL is explicitly tied to OWL. There are two components to a SWRL rule: the *antecedent* (also called the *head*) and the *consequent* (also called the *body*). The head and body each contain one or more *atoms*, which are related by a *rule*.

Informally, a rule implies that [Horrocks et al., 2004]:

> [...] if the antecedent holds, then the consequent must also hold.

Different interpretations exist where the atoms involved are OWL Classes, Data- or Object-type Properties or Instances. This method of expressing mappings as rules has interesting implications for the format of the alignment description. In particular, it permits the expression of changes to the target ontology which arise from indirect conditions, in other words, it allows for indirect relationships to be established.

This grants SWRL alignment descriptions the ability to address data concerns, and structural differences. Structural boundaries are crossed by effecting changes on the resultant entity, usually a Property or Individual. This makes the SWRL alignment description the richest of those described here. The nature of these rules is that they range from the broad to the highly specific. For example, it is possible to assert a rule that will compare XML schema date values and determine if one is within a certain range of another. One such rule is `swrlb:addYearMonthDurationToDateTime`.

However, there are some limitations. The first is that SWRL is, in its default bindings, extremely closely related to OWL [Horrocks et al., 2004]. This can be altered with the creation of new bindings, but the process is required for each representation. The second limitation is that the rules themselves are intended for ontology mergers [Horrocks et al., 2004]. While this is a valid and useful application of mappings, it is not the only use and arguably limits the utility of SWRL where other mapping applications are intended.

Example Syntax

This example is taken from the SWRL W3C user submission and represents the use of functions to compute a discount for a 'gold-level customer'. It defines the conditions, such as the gold status of the user and the value of the purchase being over 500, and applies the 10% discount if the rules are met.

3.4.5 Analysis of Mapping Expressions

The first observation to make about all of the exchange formats, other than OWL itself, there is a need to go outside of the standard languages in order to represent the relations between entities in different ontologies. It is evident from the proliferation of mapping formats that there is a need to express these mappings outside of the standard OWL format. Perhaps the most serious complications that arise from using core OWL as the medium for expressing semantic mappings is that it confuses the definitions of concepts with their relationship to others. There is no clear separation for external relationships (whose validity might depend on external factors) and the essential description of the entity in the ontology definition. This presents an interesting question about the nature of mapping that is particularly relevant in context as to the degree of independence or separability of mappings from their ontologies, and how easy it is to share mappings between different contexts.

The second observation to make is that the nature of the relationships defined in the format are perhaps the defining characteristic of the capabilities of the language. In the case of Align, despite the fact that no specific relationships are defined, it is clear from the need for different alignment 'levels' that a relationship schema must be defined externally, and that the degree to which it is tied to the ontology language(s) defines how mappings can be expressed.

Of the three only SWRL directly approaches the issue of data manipulation. There is an interesting design decision here, in that the interpretation of the relationship between concepts is tightly coupled in SWRL. In this sense, the opposite is that of C-OWL, which provides tight definitions for the logical relationship between

```
<ruleml:imp>
 <ruleml:_rlab ruleml:href="#goldDiscount"/>
 <owlx:Annotation>
  <owlx:Documentation>Gold customers get a 10 percent discount on purchases \
   of \$500 or more</owlx:Documentation>
 </owlx:Annotation>
 <ruleml:_body>
  <swrlx:individualPropertyAtom  swrlx:property="&ex;#hasStatus">
   <ruleml:var>customer</ruleml:var>
   <owlx:Individual owlx:name="&ex;#gold"/>
  </swrlx:individualPropertyAtom>
  <swrlx:datavaluedPropertyAtom  swrlx:property="&ex;#hasTotalPurchase">
   <ruleml:var>customer</ruleml:var>
   <ruleml:var>total</ruleml:var>
  </swrlx:datavaluedPropertyAtom>
  <swrlx:builtinAtom  swrlx:builtin="&swrlb;#greaterThanOrEqual">
   <ruleml:var>total</ruleml:var>
   <owlx:DataValue owlx:datatype="&xsd;#int">500</owlx:DataValue>
  </swrlx:builtinAtom>
 </ruleml:_body>
 <ruleml:_head>
  <swrlx:datavaluedPropertyAtom  swrlx:property="&ex;#hasDiscount">
   <ruleml:var>customer</ruleml:var>
   <owlx:DataValue owlx:datatype="&xsd;#int">10</owlx:DataValue>
  </swrlx:datavaluedPropertyAtom>
 </ruleml:_head>
</ruleml:imp>
```

Figure 3.7: Example SWRL syntax [Horrocks et al., 2004]

entities, but is not so descriptive on the nature of the data relationship between the participating ontologies.

The decision as to whether or not specific data transformation operations should be included in mapping descriptions depends on how the mapping descriptions will be put to use. Leading on from the analysis of mapping tools, and from the definition of C-OWL ‘contexts, it is clear that mappings arise from a local interpretation of both the target and source ontologies, which can depend on external factors not immediately visible to the mapping process.

Some of the technologies described endorse language-independence [Bouquet et al., 2003], or at least present mechanisms for extending their format to apply to other bindings [Horrocks et al., 2004]. This demonstrates that, while language independence is a powerful requirement, it is difficult to accommodate in the process of describing mappings.

Leading on from this, it is also apparent that the mappings that are presented here are not complete descriptions. Instead, they follow the properties of mappings given previously in this chapter. This means that in order to be able to interpret the mapping descriptions, it is necessary to be able to interpret the participating ontologies: the target and source as well as the articulation ontology. Note that this articulation ontology is most explicitly expressed in Align, where it is separate and not defined by the default standard.

For example, the C-OWL bridge rules define with considerable rigour the nature of the relationship being established between the entities, but does not model the domain at all, and the relationships are defined abstractly from the domain.

In effect, this comes down to a question as to what the domain model of the articulation ontology is, as in the case of SUMO, itself within the domain of the target ontologies, or whether it is an abstract description of relations. This means that, for example, if two ontologies at the same level had the *wine* and *vino* concepts, these would be linked in SUMO as instances or subclasses of the same wine class, integrating them conceptually, while the C-OWL approach would be to describe the relationship between the concepts as equivalence, without integrating them directly into the overall ontology.

3.5 Properties of Mediation

This section describes the key properties which constitute a description of an ontology mediator. The purpose of a mediator is to be able to express and execute the information pathway between different ontologies, with the objective of transferring information. These properties will form the basis for the analysis of the example systems in the following section.

3.5.1 Nature of the Articulation

The first characterisation to make of an ontology mediator is to examine the qualitative aspects of the representation it employs to bridge the ontologies. This can be of the form of an upper ontology established *a-priori*, a bespoke external mapping (such as a Topic Map), or some other form.

Property of Mediation 1: Nature of the Articulation Ontology.
Is the description of the alignments in the mediator generalised or based on the domain definitions?

3.5.2 Linguistic independence

While there is evidence[12] to support the idea that RDF/OWL continues to grow as an ontology language, several other candidate languages exist, and the process of providing mediation means that there is a need to pass between different dialects. This points to it being desirable for a Mediator to have some degree of independence from the languages which the mediated ontologies are expressed in.

While SWRL and C-OWL are technologies deeply embedded with the OWL semantics, the Align format is less constrained, providing independent and specific representations dependent on level.

[12]The author of [O'Sullivan, 2006] cites a June, 2005 search on Google as reporting ten thousand entries. In July, 2008 the same query "`http://www.google.com/search?q=filetype:owl+owl`" reported in excess of sixty-seven thousand results.

Similarly, the OISIN process was implemented with OWL in mind, though the procedure itself is applicable to a wide variety of ontological language, and indeed general schema mapping. The Matches found by the COMA++ system have been shown in a variety of formats, including SQL and XML XSD (Schema).

It is important to know if a contextual mediator is tied to the semantics of the languages in which the ontology it accepts are represented, and whether this is extensible.

Property of Mediation 2: Linguistic independence.
Are participating ontologies required to be written in a specific language?

3.5.3 Mapping representation

In importing the mappings and the ontologies which are to be mediated, an ontology mediator can represent the participating information models in different ways. This property is reflected in several ways by the expression and mapping technologies. In order to find agreement between a wide variety of different schema formats, and in order to be able to apply a variety of techniques to the model, COMA++ represents all ontologies in an internal graph format. Pure OWL ontologies with external references mix the definition of the entities and the references to external properties at the same level.

Where the representation models are extrinsic, the question remains as to how the representation of the information within the entities is exchanged. Of the systems described, SWRL makes particularly descriptive provision for data transformation. Arguably, it is not appropriate for this information to be found in a general exchange format, but for the purposes of developing an internal representation for a mediator, there is a clear need to be able to describe the data relationship, as well as the knowledge relationship between entities, if information is to be transferred. It is also convenient to store this information in the exchange format, as it would otherwise need to be represented and correlated to the mappings in any case.

This property can also be seen in business maps. The Topic Map is able to represent associations between entities and the nature of those associations.

Another difference between the intrinsic and extrinsic descriptions of mappings is that extrinsic mappings often include a notion of the confidence or strength of a link. OISIN defines committed ontological links, which can be used as a basis for mapping subclasses. The Align format provides for a measure from 0.0 to 1.0 of confidence, and COMA++ uses a similar measure.

Property of Mediation 3: Nature of Mapping representation.
Is the representation tied to the definition of the entities or not?
Is there information about the strength of the link?
Is there information about the data relationship of entities?

3.6 Survey of Ontology Mediators

The sections above have described the affordances which the overall semantic interoperation process can offer to an ontology mediator. These key properties form the basis for the design of such mediators. This section of the survey analyses two key examples of ontology mediators from the literature, with the objective of understanding what practical mediation requires. The first system, WSMO is an example of a mediator that makes extensive use of functional data transformation and which can accommodate a variety of different ontological representations. The second system, DRAGO, was chosen because it represents the formal, ontological approach to inferring across distributed knowledge bases. These systems demonstrate the spectrum of design choices which are part of ontology mediation.

Ontology Mediation can take advantage of the results of mappings found using the techniques described above (*c.f.*3.3) , rendered in one of the exchange formats described (*c.f.*3.4), to transfer knowledge from one or more ontologies to a target. It is a specific process which keeps the participating ontologies separate.

3.6.1 WSMO

The Web Service Modelling Ontology is an upper ontology that is a component of a complete mediation framework [Feier et al., 2005]. The objective of the framework is to provide a means for choreographing and orchestrating semantic web services. This is consistent with the model of web services in general.

In WSMO terminology, a Web Service is an atomic unit of functionality (a service), which is invoked using web protocols. *Semantic* Web Services include ontologies that describe the 'means to buy and search services' [Feier et al., 2005]. The WSMO itself is an ontology for describing semantic web services.

WSMX, the execution environment interprets WSML (Web Service Modelling Language) annotations to services in the process of discovering, selecting and mediating semantic web services.

WSMO employs ontologies as sources for domain terminology in the annotation of web services and goals for the purposes of organising service chains. Ontologies can be imported directly, or by means of *mediators*. Two levels of mediator are described, with the *data level* mediators being responsible for ontological integration [Feier et al., 2005].

Mediators are described are categorised in one of four categories. *ooMediators* are responsible for aligning two ontologies, so that terminological mismatches and other alignment tasks are resolved. Other Mediators exist to refine the goal composition (ggMediators), link Web Services and Goals (wgMediators), and manage integration at a protocol level between web services (wwMediators [Feier et al., 2005]).

Within the structure of the system, mediators are treated as third-party entities, and modelled as Web Services with particular properties[Roman et al., 2005]. This means that the WSMO framework itself does not intervene in the details of how mappings and alignments are achieved, but is aware of the pre- and post-conditions, as well as the functionality of different mediators and can choreograph them.

While the focus of the WSMO design is not on ontological interoperation, but on service choreographing and orchestration, it represents a detailed example of a semantic interoperation framework. WSMO is a semantic broker[13], and by relating the goals of the user, it discovers and sequences web services using semantic descriptions of their knowledge.

Nature of Articulation

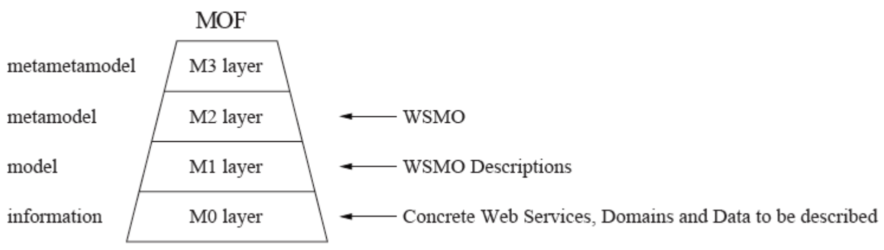


Figure 3.8: WSMO Modelling layers, with the top item being represented in the MOF format [Roman et al., 2005].

One particularly interesting feature of WSMO is that the designers have separated the conceptual level at which the knowledge of the services is expressed from the level at which service interactions are described. Furthermore, the data level is expressed in a separate modelling layer (*fig.*3.8). This is an important feature in realising the orchestration of different services. The expression of data relationships arises from the fact that mediators must maintain relationships with separate representations, and so similarity of data values is not guaranteed.

The top-level WSMO concepts are mapped to the web service and goal terminologies by first importing those ontologies into a broader model that expresses the requisite entities from each web service. As described above, individual mediators can be specified at the 'data level' to represent ontological alignment, and these can be composed of multiple individual reasoners.

The top-level entities include general notions such as `Relations` and `Concepts`, as well as more concrete examples such as `Web Services`. These are the building blocks

[13]As the term 'mediator has specific meaning in WSMO', for the purposes of clarity, 'broker' can be taken to mean that it mediates between different services.

for the descriptions of the services that will interact.

The specific mappings between the top-level ontology and the participating web service ontologies is expressed using WSML, which is a set of languages [Roman et al., 2005] which inherit from first-order and frame logic.

Linguistic Independence

The WSMO itself is derived from MOF[14], a meta-language that provides a way to formally describe the top-level concepts.

Internally, WSML is used to describe the ontologies. The authors note that other languages are not sufficient for the purposes of choreography and orchestration, and have therefore defined WSML themselves [Roman et al., 2005].

The use of mediators permits WSMO systems to import ontologies specified in OWL and other languages. However, given that the semantics of these other languages are poor, it is possible that there would need to be considerable intelligence on the part of the mediators used to import concepts to represent them sufficiently.

In treating mediators as services, the WSMO architecture means that mediators have considerable flexibility. There are few restrictions on their functionality, so long as they have well-described properties. Provision is made for different levels of mediator (as discussed above), and mediators can themselves be choreographed for complex operations.

Mapping Representation

The separation of levels between the entities expressed at the model level in WSML and the data relationships expressed in process-level mediators (wwMediators) mean that there are effectively three separate representations of mappings within the WSMO architecture.

The first level of representation is that which occurs between entities described in

[14]Meta Object Facility [Roman et al., 2005]

ontologies, and is expressed with the full richness of the WSML language. Relations and Functions can be used to create numerical and non-numerical rules that relate the concepts, and can be used to check that information is in the correct format. As this is an ontological language, there is no provision for fuzziness in relationship specification, though it could perhaps be included in the Non-functional properties that accompany entities within WSML.

The second level of representation allows process-level mediators to manage protocol and syntactic differences between the web services themselves at a low level. The nature of the mediators is not specified by WSMO, as they are independent third parties, treated as services.

Similarly, the process by which ontologies are mediated using ooMediators is unspecified. This permits WSMO to import ontologies and employ mappings that have been created and reviewed externally, and execute them externally while taking advantage of the results.

In summary, the split approach between mediated and unmediated representation means that WSMO systems can employ a domain-specific relationship language to describe the interoperation of web services, and where this is not appropriate, it can make use of 'black box' mediators to provide suitably altered knowledge.

3.6.2 DRAGO Distributed Reasoner

One advantage of the use of ontologies is the option to perform inference [Bechhofer, 2003]. There are two categories of inference, T-Box (where the terminology of the ontology is reasoned over) and A-Box (where the assertions, or instances are reasoned over). It is the combination of A- and T-Box information that makes an ontology a knowledge base.

DRAGO[15] is a distributed reasoner system based on the DDL[16] language [Serafini and Tamilin, 2005], that expressed ontologies as description logic knowledge

[15]Distributed Reasoning Architecture for a Galaxy of Ontologies [Serafini and Tamilin, 2005]
[16]Distributed Description Logic

bases and represents the mappings between them.

DRAGO is designed to be able to reason over a set of mapped ontologies, located at different points on a network, via a distributed reasoning algorithm. The topology of the distributed knowledge base is a peer-to-peer system [Serafini and Tamilin, 2007], where information can be passed between nodes in the system as required,

Distributed reasoning is performed on the basis of both terminology and assertion information [Serafini and Tamilin, 2007]. This permits the communication of both schema and instance information between nodes.

The functionality of the system is achieved by converting conventional DL-compatible Ontologies into DDL knowledgebases, which are then able to be mapped by conventional techniques [Serafini and Tamilin, 2005]. The nature of the bridge rules that are used is such that A-Box and T-Box migrations and rules are orthogonal.

A distributed tableau algorithm is central to the DRAGO architecture. This algorithm is designed to allow an large, unwieldy ontology to be reasoned in a piece-wise fashion at a local level through mapping. There are clear advantages of feasibility and performance in not attempting to reason an entire general upper ontology.

In implementation, C-OWL(*c.f.*3.4.2) is used as the language for expressing mappings. Specifically, the following bridge rules exist[Serafini and Tamilin, 2007]:

- $i : C \underset{\longrightarrow}{\sqsupseteq} j : D$, the into-bridge rule
- $i : C \underset{\longrightarrow}{\sqsubseteq} j : D$, the onto-bridge rule
- $i : a \mapsto j : b$, for corresponding individuals

In this notation, C and D are concept names, and a and b are individuals.

The three relationships defined are used to relate the concepts and individuals in distributed ontologies. Because of the inference-based approach, DRAGO can provide satisfiabillity, subsumption, instantiation and retrieval on distributed ontological queries.

Implementation of the DRAGO system, pictured in fig. 3.9 is achieved by extending

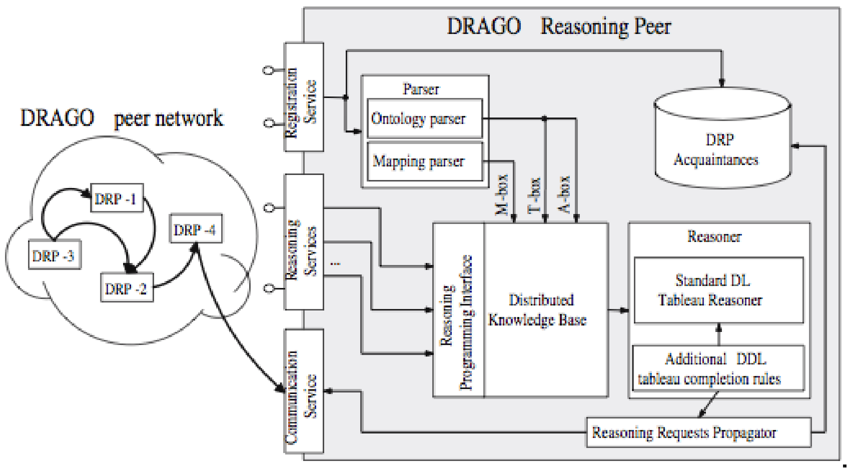


Figure 3.9: DRAGO Architecture [Serafini and Tamilin, 2007]

the Pellet reasoner system with an M-Box, to represent mapping reasoning. Pellet is an OWL reasoner, and its reasoning is combined with the C-OWL semantics to express mapped ontologies.

Nature of Articulation

The DRAGO approach is based on adding an M-Box to the reasoning system of a standard OWL reasoner. This means that the ontology for describing the semantic links between participating ontologies is restricted to the definitions of the bridge rules. These are formally defined concepts, and as they are represented in C-OWL, derive from the OWL semantics and the extensions made to those semantics.

In effect, the objective in DRAGO is to treat the individual ontologies not as conceptually lower portions of an upper ontology, but rather to consider each separate ontology as a peer in a distributed, holistic view. This approach has many advantages, most notable of which is the fact that individual reasoning tasks can be resolved locally, which has performance and architectural advantages.

Linguistic Independence

Many of the linguistic properties of DRAGO derive from the fact that it employs distributed reasoning to resolve semantic matches. This means that any ontology subjected to such reasoning must be compatible with the tableau reasoning algorithm, or convertible to that format. An ontology parser is used to convert the ontology description files into an internal DL knowledgebase format, suitable for reasoning. For the most part, this would appear to restrict DRAGO ontologies to ontologies written in OWL. This is due to the specific properties of OWL that make it amenable to DL-reasoning.

As DRAGO is a reasoner-oriented technology it appears that there is a particular benefit from using the OWL-DL dialect. The authors do not indicate whether this is a requirement, but it would seem likely that OWL-Full would not be fully usable, due to the use of DL reasoning algorithms.

Mapping Representation

The DRAGO framework does not specify how mappings are themselves achieved. Mappings are established by external methods, which means that the best techniques can be selected for specific ontology properties, for example with a framework such as OISIN.

The mapping representation in DRAGO bears many of the properties of those discussed for C-OWL. DRAGO mapping comes in one of two forms for the Schema and also expresses the correspondence between instances. This presents an extremely limited amount of information about the relationship between the ontological entities. However, the accounts for experiments in [Serafini and Tamilin, 2007] indicate that even a low level of information can be useful for operations such as instance migration.

As this is a language that extends from Description Logic, there is no provision for uncertainty or confidence intervals in the descriptions of the links. There is also no method to reason about the data relationship of different entities, they information within must be the same.

3.7 Additional Properties of Ontology Mediators

Based on an analysis of example Semantic Mediation systems, it is clear that there are a number of properties common to these systems that derive uniquely from their position in the interoperation workflow, and from the particular properties of mediation, as opposed to alignment or merging.

3.7.1 Nature of the Internal Representation

The first observation is to note that mediators' features depend heavily on how they represent internally the ontologies which they are mediating. In the case of WSMO, this is achieved through the use of a domain specific ontology which is written in a custom language. This permits WSMO to be highly expressive and detailed as to the relationships between terms and instances. In the case of DRAGO, there is a relatively small list of relationships that can be used to express the mappings, but they permit the mediation to be executed within the semantics of the ontology language. This relates to the question of the nature of the articulation that is associated with the mediation. It is clear from both approaches, and from the analysis arising from the earlier technologies, that there is a need for a semantic representation of the mappings that exist between ontologies. A semantic representation means that the mappings are translated into well-defined relationships, which can either be reasoned or otherwise retrieved as required, based on their properties.

Property of Mediation 4: Nature of the internal representation

Does the mediator convert external ontologies?

Are the converted ontologies merged into a global ontology?

Is the reasoning of the participating ontologies retained?

3.7.2 Data Transformation

WSMO recognises levels of entities and data, and separates them. While both DRAGO and WSMO import the schema and instance information of the participating ontologies into their internal formats, only WSMO recognises the need to be able to allow for there to be representational differences in the content of the instance information. This would seem to be an important feature to support services, that arises from the application domain of the design. This property can be observed in SWRL, where extensive functionality is provided, for example in date manipulation.

Property of Mediation 5: Data Transformation.
Is the mediator able to execute transformations on the data that the ontologies hold?

3.7.3 Knowledge Translation

In approaching the issue of data (as opposed to knowledge, which is represented in the ontology), the WSMO approach is to hand off the detail of how information is transformed appropriately. The different classes of mediator are called with a pre-condition of the source and target ontology, and the result is a transformed ontology. This approach has merit in that it allows the issue to be handled with considerable flexibility, however it would be useful to be able to reflect more complex operations in the mediation. For example, this might include accounting for complex structural gaps at a schema level between ontologies. There are features in WSMO WSML that represent an example of this capability.

Property of Mediation 6: Knowledge Translation.
Is the mediator able to bridge semantic gaps[17] *in the ontologies?*

3.7.4 Mapping Importation

There is, from a study of the literature, a wide variety of techniques open to perform ontology mapping. Many of these depend on the aspects of the process that developed the ontology, and also on the domain in which the ontologies are written. It is a desirable feature for mediators to be able to import mappings from external formats,

[17]See definition above.

such as those described above. This is necessary because many of the mapping processes are either manual, or require such time as to be off-line processes.

Property of Mediation 7: Mapping importation.
Is the mediator able to import mappings from external sources, how tied are these to their representation?

3.8 Conclusion

This chapter has described and analysed the technologies that are directly associated with Semantic Mediation, as well as the influences and approaches of the technologies that support mediation. In the process of examining these systems a clear set of criteria by which mediation can be judged has appeared. What is apparent from the review of the systems which exist is that there are a number of approaches, and that each has advantages and disadvantages.

One approach which appears to have some merit is the concept of creating an ontology mediator which takes a component-based approach. One lesson from the design of DRAGO is that mapping and local reasoning can be carried out with relatively strong separation. It might be possible to create a reasoner that takes the separation of reasoning environment to a logical conclusion. In this case, the mediation is between *interpreted knowledgebases* rather than ontologies, and instead of attempting to break up the ontologies into an internal representation (as WSMO does), or limiting oneself to a narrow range of possible ontology formats (as in DRAGO).

There appears to promise in the notion, partially presented in business maps, of overlaying a separate ontology, designed for information linking rather than canonical description. In taking such an approach, it would be possible to shed the restrictions of expressing mappings between entities that are greatly dissimilar[18]. In this case, it would not be possible to retain a contiguous reasoning approach, but rather it would

[18]For example, where the representation of an entity in one ontology is fulfilled by a Class instance, and in another it is fulfilled by a data property.

likely be necessary to take an approach similar to that of the mediators found in WSMO systems.

Neither of the ontology mediators described takes into account the confidence rating which some mapping tools can produce. These measures of confidence can, in some cases, contribute to the selection of the correct mappings from a list of possible candidates. An example of use might be that a mediator might only choose to transfer using mappings of a particular rating, or with a particular metadata attribute. This mechanism has particular importance in situational mapping scenarios as highlighted in the context domain.

In conclusion, there appears to promise in an ontology mediation system that is designed with the complexities of the alignment workflow in mind. This is particularly reflected in the notion that there is a need for a system that can accommodate the richest possible linking, and the widest variety of formats, both for ontologies and for mappings themselves. However, one area not yet addressed in the systems described in this chapter is the issue of supporting different levels of knowledge transfer in mediation. The mediation systems described above address the issue of transferring instance information between different systems, but the notion of transferring schematic information is not addressed.

3.8.1 Summary Table for the Mediators

The table below summarises the properties of each of the two mediators discussed in this chapter. The properties are taken from the analysis provided above.

#	**Property**	**WSMO**	**DRAGO**
1	Nature of Articulation	WSMO Ontology + WSML mapping	C-OWL Bridging relationships
2	Linguistic Independence	Dependent on OWL and C-OWL	Mediator converts to internal formats
3	Mapping Representation	WSML Rules	Bridge Rules + Instance Correspondence
4	Nature of internal Representation	WSMO Ontology	DDL Knowledgebase
5	Data Transformation	Mediation + WSML Rules	None
6	Knowledge Transformation	Mediation + WSML Rules	T-Box Reasoning
7	Mapping Importation	Mediation	OWL-C or converting parser

Table 3.1: Ontology Mapping & Mediation Feature Matrix

Chapter 4

Design & Architecture of a Context Mediator

This chapter is concerned with the design principles and architectural structure of a Context Mediator suitable to support context-informed semantic interoperation. The chapter opens with an abstract description of the contextual environment surrounding a user, along with a high-level description of the process of context integration which drives the design of the system. Building on the results of the state of the art chapters in Ontology Mapping and Context, this chapter then describes the key design goals of the system, and relates the system described here to the literature. This includes the key design decisions made in creating the new system. A list of requirements are then specified for the new context system, which describe the parameters of the design of a context mediator using semantic interoperation techniques. The major novelty of this design is that it supports the context-informed approach through a model-based exchange with semantic interoperation. The architecture of this mediator and its principal components are then discussed, before a summary conclusion.

4.1 Introduction

The objective of this chapter is to describe the basis upon which Context-Informed Semantic Interoperation is designed. There are key requirements which are drawn from the State of the Art in both Context and Semantic Interoperation. The framework is intended to address the challenge of effective context integration without the need for an *a-priori* model of context. A semantic network is used to represent the articulation of the mappings between the sources of context and the user's target application.

This chapter will particularly address the process of integration from the perspective of the changes in knowledge (broadly represented by the ontology schema), information (broadly represented by instance information in the participating ontologies) and data (broadly represented by data values within the instances). The system presented here will address the process of importing and representing the knowledge of the sources and target, as well as the mappings between them.

4.2 Abstract Framework

In order to provide a theoretical reference point for discussing the approach in this chapter, this section describes the key components and their relationship to each other in the contextual frame of reference.

A **Participant** is an application or service which is a member of the context space. Participants are either producers or consumers of context. The majority of participants are **Sources of Context**. Sources produce contextual information which can be used to enhance the knowledge and behaviour of the context consumer, which is called the **Target Application**. This target application's knowledge is enhanced by the **Context Integration** process.

The context integration process is undertaken by the **Context Mediator**, which uses different kinds of inference and reasoning to correlate the information from the sources of context in a form usable by the target application. The information will be transferred and converted based on links and attributes that describe the differences between the source and target representations, and which will be collectively known as the **Integration Pathways**.

This process can be viewed as being one which is focused on the context mediator acting as a forum for agreement between different parties to an information exchange. In particular, the sources of context are aware of relevant information about the user or their situation. This information would be useful to the target application, if it were in a form which is understandable by the target (i.e. if it were added to the target's ontology correctly).

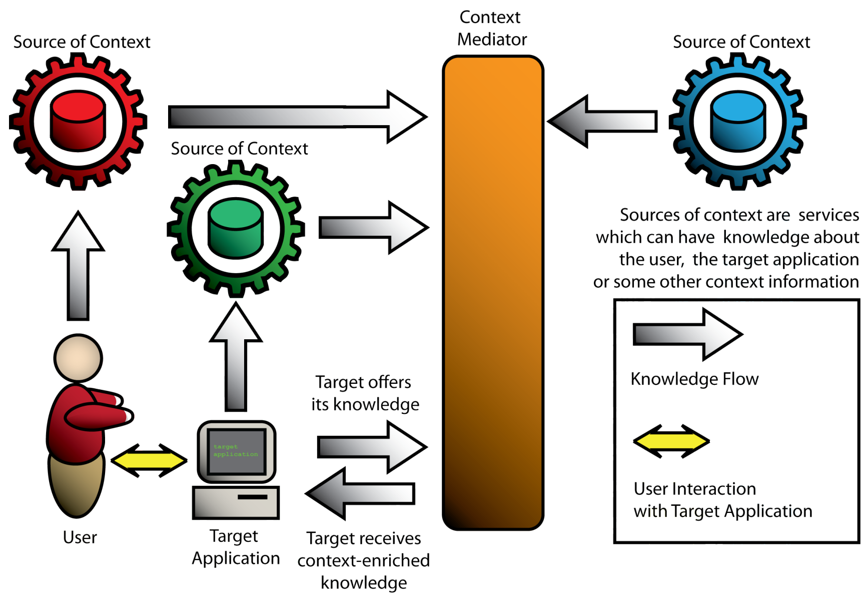

Source of Context
Context Mediator
Source of Context
Source of Context
Sources of context are services which can have knowledge about the user, the target application or some other context information
Knowledge Flow
Target offers its knowledge
User Interaction with Target Application
User
Target Application
Target receives context-enriched knowledge

Figure 4.1: Abstract Context Mediation Framework.

In **Fig.4.1** information is gathered by Sources of Context about the User while they interact with the Target Application. Meanwhile, the Context Mediator transfers the knowledge from the Sources to the Target Application, based on the mappings represented in the system and the information in the target's ontology.

4.2.1 Context Identification

The identification of relevant context information is a key challenge in context mediation. Previous systems have focused on the use of a method where the target application forms a query about specific information, and integrates the response. The system presented in this thesis takes a different approach. In order to facilitate the exchange of knowledge, as well as information and data, the process is one of model enrichment. At a suitable point in the logic of the Target application[1], the application author can update the ontology which is registered with the context mediator. A new ontological model is returned to the Target Application, with changes based on the mappings and the knowledge in each participant's ontology.

This model is focused on the mediator 'pushing' new ontological information to the Target Application, based on the collective state of the sources and the target. This is to facilitate the application designer: they do not have to know what information a context mediator might know, instead the target application should be designed in an open way that can accommodate new knowledge.

From the push vs. pull, perspective, control rests with the Target Application. It is the Target application which initiates the exchange, and there is no absolute requirement for the Target Application to incorporate everything which is returned by the context mediator.

The design presented here means that the target application need not know about specific context that might exist. Instead, it needs to be designed with particular functionality to support a third party (the mediator) adding new knowledge and information to the application, based on external considerations. This is a highly flexible solution which facilitates significant changes in behaviour by the Target.

For a mode extensive discussion of the specific properties of context and the architecture of target applications, please see [O'Connor, 2005].

[1]For example, where an adaptive application is about to recompose its presentation.

4.3 Design Goals

The objective of context mediators in general is to provide a 'global' view of information shared in a particular context, and to communicate the relevant parts of that information to the user's application [Gu et al., 2005].

The goal of this design are to construct a context mediator which can create an informing environment for applications. This means that the mediator maintains a collective view of the knowledge of both the producers and consumers of context, and can suggest enriched versions of a Target Application's knowledge when required by that application.

Specifically, this goal breaks down into the creation of a system that can

1. Represent and characterise the knowledge held by different participants. This means that the system represents the contextual knowledge which is represented in the Sources. The mediator must also be aware of the knowledge which the Target Application has, in order to be able to reconcile the sources and target's knowledge.

2. Represent and characterise the mappings between concepts in the knowledge held by different participants. This defines the parameters of the agreement between the sources and the target. The mapping is principally between the target and several sources.

3. Transfer knowledge from the Sources of Context, at the correct level of abstraction, to the Target Application, based on the mappings

4. Perform other operations on the representation of the mappings, and the information held by the participants, in order to make the transfer happen correctly. One example of this sort of operation is to take advantage of mapping information from several different tools. This allows the system to choose, for example, only links which have been rated highly by two of three mapping tools.

These goals are achieved through the use of semantic mediation techniques to express and execute mappings between ontologies from different participants. The use of semantic mediation permits this mediator to avoid the need to have a pre-conceived notion of what elements constitute context. Instead these entities are derived from the ontologies which take part in the context integration. This has the advantage of permitting a wide variety of information to become contextual knowledge. Virtually any information which can be mapped and rendered in the correct form for the target application can be used for context.

4.3.1 Design Assumptions

The design presented in this thesis depends on a number of assumptions about the nature of the participants and the representation of the information in the contextual scenario.

One key assumption is that the nature of the context mediation is a many-to-one relationship, where one or more sources of context are aligned with one target application. This assumption allows the system to make use of a variety of conventional semantic interoperation tools, which are commonly focused on one-to-one relationships, which may not be symmetrical.

The inherently heterogeneous and varied nature of contextual information means that it is necessary to address both the representation and the content of context information to integrate it. It is therefore assumed that all of the information and knowledge that is exchanged between the sources of context and the target application will be represented in an ontology.

Similarly, ontologies can be represented using a wide variety of languages. In this design, the ontologies are manipulated by the native reasoners for that language. It is therefore assumed that any ontology used in a version of this design will be in a representation which can be integrated with a suitable native reasoner interface integrated into the design.

Finally, the requirement that the information be transferred through ontologies imposes the constraint that sources of context must be able to emit information in an ontological form which includes both schema and instance data. While this requirement may seem onerous, the nature of semantic interoperation means that simpler sources of context can emit relatively simple ontological information. In addition, since the mediator is responsible for loading native inference, this also relieves the burden of inference from sources of context.

On the Target Application side, it is a necessity for the Target Application to be able to both emit and receive ontology information. Different types of contextual enrichment impose different requirements on the Target Application, from being capable of handling straightforward data updates, up to the inclusion of substantial schema alteration which facilitates the inclusion of new knowledge in the target. The default assumption in the design is that the target application can alter its behaviour in response to some level of ontology enrichment.

4.3.2 Context Integration Process

The context integration workflow can be outlined as beginning with the registration of participants with the Context Mediator. This registration involves a process of offering an ontological description of that participant's knowledge.

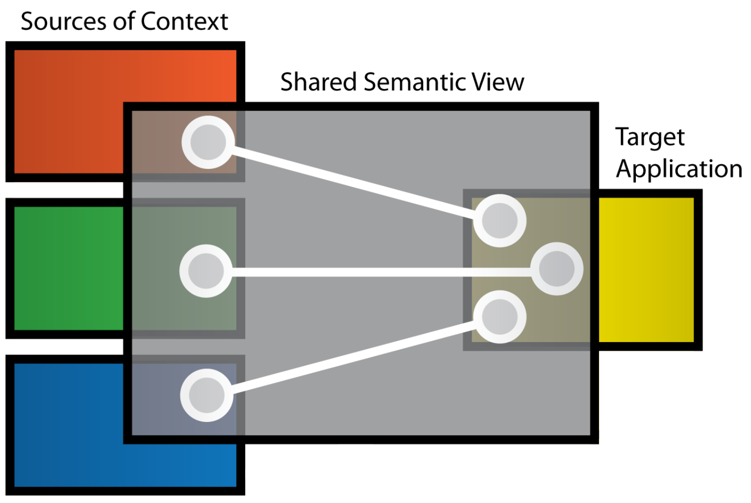


Figure 4.2: Shared Semantic View Representation

At a high level, the Shared Semantic View can be regarded as being overlaid on the ontological representations of the knowledge within the sources of context and the target application. The integration pathways represent links between concepts in the different participants and information can flow along those pathways. It is important to note that the pathways can characterise an information or data relationship, for example to provide for a data transformation operation.

Once participants are registered, a **Shared Semantic View** can be established. This is a representation of the integration pathways of the mediator. Initially, this Shared Semantic View will represent the concepts found in different participants' ontologies. These can be found, for example, through a knowledge discovery process whereby the ontologies are analysed for their entities. The second phase in establishing the Shared Semantic View is the process of establishing the pathways between the concepts themselves. This can be done by the use of online mapping techniques, or through off-line processes, and imported through a description file. Once this and any additional reasoning is complete, the Shared Semantic View is established.

The transfer process itself is activated when a request arrives from the Target application for enrichment. One key feature of the informed, rather than the aware model of context integration, is that the Target does not need to form explicit requests, but instead offers its knowledge, which is analysed by the Mediator for enrichment.

Once the Target offers its knowledge in ontology form to the Mediator, reasoners in the Mediator use the Shared Semantic View in three stages:

1. **Identification**: where relevant information needs in the Target's knowledge are found. One form of this is, for example, where an instance of a class is known in a source but not in the target.
2. **Semantic Transformation**: is the process whereby the integration pathways are used to arrange information to be transferred to the Target into a compatible semantic form, for example by ensuring that the proper level of abstraction is expressed. For example, by converting from a percentage measure to an

interval, star measure.

3. **Syntactic Translation**: in this phase, the information which is in the correct semantic form is added to the Target ontology, complying with the structures and syntax of the Target's ontology. For example, by taking information from a literal in one ontology and representing it as an instance in the target's ontology.

Each of the processes outlined above will need to be supported by *user-defined* reasoners. These reasoners are required because the expression of the integration pathways lies outside the characterisations which are possible within the ontological languages. For example, the OWL structures do not make provision for supporting certain abstraction-spanning relationships, or the characterisation of data transformation requirements.

4.4 Requirements and Influence from the State of the Art

This section describes the decisions that result from the conclusions from the state of the art. Broadly, the influence of the state of the art in semantic mapping techniques has helped address some of the key requirements in the method by which ontological concepts and their mappings are represented and characterised. This system is dissimilar to the two semantic mediators described in the state of the art in its approach to representing and executing mappings, and in its approach to identifying context information for transfer. It facilitates new levels of exchange by making knowledge transfer possible.

This section is divided into key categories for the description of the requirements intended to fulfil the design goals of the system. The first area discusses the architectural approach to the mediator. This is a fundamental decision which affects the entire design. Requirements are then defined for each component of the framework.

4.4.1 Architectural Requirements

There is some agreement on the advantages of the use of a centralised mediator architecture for supporting context exchange [Baldauf et al., 2007]. This model is a natural one for a mediator, as it creates the natural meeting point for the different knowledge held by each participating service.

Many systems are built on the assumption of the existence of a *Smart Space* [Wang et al., 2002], a sensor-enriched physical space which the user inhabits. Typically, this space would include location technologies, such as Ubisense [Steggles and Gschwind, 2005]. These devices are usually locally managed sensor systems, and impose requirements on context systems to be able to interpret highly-unpredictable physical measurements from sensors.

4.4.2 Participants - Produces & Consumers of Context

One consistent challenge arising from the design of Context Mediators has been that of attempting to reconcile differing levels of abstraction and dynamicity. The nature of the producers and consumers of context which the mediator supports has great impact on the design requirements for the system. This section first examines the sorts of context-producing services which exist, and then discusses the nature of the target applications which consume the context.

Sources of Context

The SOCAM [Gu et al., 2005] model of 'internal' and 'external' context services was founded on the notion that certain participants would be locally administered and therefore have different interface requirements to remote, external services. One example of this is the notion that location would be measured by local sensors and managed by a local interpretation service.

However, since that time, the advent of GPS[2]-enabled mobile phones and the

[2]Global Positioning System

Software-as-a-Service (SaaS) approach to service development has altered the validity of this model. It is now possible to conceive that the majority of the participants in a context integration will, from the perspective of the mediator, be external and those few remaining internal services will be able to communicate in such a fashion as to be indistinguishable from external services[3].

The design of CoBrA [Chen et al., 2004b] seems to point to the viability of allowing for agent or other software methods to be given responsibility for handling the highly dynamic and uncertain signal processing of sensor management. This further reinforces the conclusion that the issue of dealing with low-level data such as that gathered from sensors is not a concern for the context mediator, but can be assumed to be interpreted by a SaaS offering, or other independent agent.

The requirement which arises from this is that the design of this system will assume that web-service based communication is possible with the sources of context, and that they will communicate their knowledge with the mediator through ontologies.

Target Application Specific Requirements

On the side of the Target Application, there is an observable trend towards customisation and personalisation of applications for users. This can take the form of templates and plugins[4], and permits the application to alter the presentation and management of its content based on customisations devised by site authors.

A more fundamental and powerful move towards creating personalised user experiences is evident in the move towards adaptive systems, especially Adaptive Hypermedia [Brusilovsky, 2001]. Adaptive systems employ a variety of distinct models to represent key aspects about a user their content and more, and can choose different strategies to give users access to individually-created information presentations.

[3]for an example of the sort of context technology that is becoming available in a SaaS offering, see Yahoo! FireEagle (`http://fireeagle.yahoo.net`), where local sensors and remote services collaborate to provide knowledge.

[4]such as, for example in popular blog software like Wordpress, `http://www.wordpress.org`

Supporting adaptive systems imposes certain requirements on the mediator. Adaptive systems already contain considerable knowledge about the user and their task, and are equipped with detailed models. The motivation for adding context in this case is that it must be information which was previously unknown or unavailable to the designers of the adaptive system. This means that it is difficult to query for such information specifically, as in traditional context-aware systems. Instead, the adaptive system designers can designate contextual 'hooks', which are points when certain relevant models can be enriched by the context mediator. This requires the context mediator to accept ontological descriptions of adaptive models as the input from the Target Application, and also requires the mediator to return the enriched models in the same form.

4.4.3 Mediator Information Model Requirements

The use of semantic web techniques, specifically ontologies for reasoning and information transfer is widely supported by the systems reviewed, as well as others [Strang and Linnhoff-popien, 2004]. Ontologies provide a structured medium for transferring not only data, but also a representation of the knowledge of a participant. In the case of this system, context lies within the knowledge of the participants - the sources and target. The key requirement for this system is that it should provide a medium for transferring information from these sources to the target. In order to take full advantage of the heterogeneity of potential context sources, the mediator is required to generate these pathways dynamically, and to be able to express them from mappings created using established semantic techniques.

The use of this sort of dynamic model of integration imposes additional requirements on the structure of the mediator. The first implication is in the management of the participant ontologies themselves. The use of semantic mapping also makes requirements on the mediator design.

Ontology Management

Ontologies related to participants need to be maintained by the Mediator as Knowledge Bases, that is as active, query-able entities which include both terminological (T-Box in DL[5]) and assertion (A-box in DL) information. It is necessary to be able to access both the schema and instance information from the ontologies, in order to be able to import and map the ontologies, as well as transfer information between them.

The Mediator will also need to be able to update parts of the ontology with information from the participants, or with the results of an integration. This imposes a requirement for the use of an abstraction layer, which will provide management functions for a number of ontologies in different languages.

Requirements Arising from the Use of Semantic Mapping

In describing the system as an ontological as well as context mediator, it is necessary to draw from the results of the survey of semantic interoperation to gain an understanding of the properties of a semantic mediator which would be suitable for serving context in the form outlined above.

One observation which has motivated the interest in ontology mapping has been that there is a need to support multiple ontologies, each with different focus, in order to be able to support tasks in a distributed environment [Kalfoglou and Schorlemmer, 2003]. This points to the notion that there are different ontologies being generated which suit specifically the properties of the systems and domains that they describe. The lack of a consensus on global ontologies is a key reason to pursue ontology mapping [O'Sullivan, 2006], which can be used to bridge the gap between different ontologies at various levels.

The notion of a mapping approach is also appealing from the nature of context itself: the objective of merging heterogeneous knowledge is present in both problems, and some of the problems can be naturally addressed by the union of the two approaches.

[5]Description Logic

For example, a key aspect of ontology merging is to be able to deal with differences in levels of subsumption within different representations. Similarly, one of the key uses of ontology reasoning in both Construct and SOCAM is to be able to use ontological reasoning to infer transitive relationships, and to check consistency.

Another advantage to the mapping approach is that mappings can be established on a 'contextual' basis. This means that they can represent relationships that are not necessarily intrinsic to the definition of an entity (as they would be if they were expressed within an ontology), but are rather the result of external factors. The design of this system will include an aspect of mapping management, with the tool used to represent the mediation pathways needing to be able to express those pathways and alter them. This points to synergy with the prevalent use of user-defined reasoning methods, which appear to be important to previous context systems. These user-defined reasoning issues can be compared closely with the functions available in, for example in SWRL [Horrocks et al., 2004] and WSMO [Feier et al., 2005].

Ontology mapping is a complex process which can only be partially automated, at best [O'Sullivan, 2006, Kalfoglou and Schorlemmer, 2003]. There are many techniques [Engmann and Massmann, 2007] , and several methods for expressing the results of those techniques [Bouquet et al., 2003, Euzenat and Schvaiko, 2007]. These technologies can be employed in two means, the first is by providing a direct method for establishing and communicating mappings, and the second is in helping to describe the representation which the integration pathways might exist in.

Ideally, the representation of the mediation pathways in this system will be in a format that can be exchanged, and the system itself will be able to handle a variety of mapping descriptions.

4.4.4 Reasoning Requirements

Reasoning refers to a number of different functions that can have an impact on the information in the participants or the mediator itself. Previous systems, such as

WSMO, made use of a custom set of reasoners (called mediators) which could be used to supplement the ontological reasoning of the WSMO upper ontology. In the model for this system, one of the key features is that there is no upper ontology *a-priori*. This means that there needs to be a model for permitting the open-ended extension of custom inference.

Ontological Reasoning

The DRAGO system is an example of one approach to the use of Ontological reasoning to fulfil the requirements of a mediator which does not have an *a-priori* semantic model. In DRAGO, the C-OWL bindings are used for the basis of a distributed inference process that permits different knowledge bases to exchange instance information. This model is effective, but somewhat limited for the purposes of contextual mediation. The first challenge is to overcome the fact that C-OWL and DRAGO depend on DL-based OWL reasoning, and so are of somewhat limited flexibility as far as language goes. The second limitation is that the C-OWL relationships do not facilitate the alteration of the data before it is passed to the new instance. Experience from the Context mediators described previously points to the usefulness of ontology technologies for providing a format for exchanging knowledge in a structured fashion. This, combined with the DRAGO-like idea of collaborating knowledge bases forms the approach for the system described here. The objective is to use an ontology and its inference engine as the participant, and to link those active knowledge entities within the Shared Semantic View. This design permits the mediator to take advantage of heterogeneous models by dealing with them natively as far as is possible.

User-Defined Reasoning

There are several properties of context which make it difficult to model *a-priori*. The general issue is that at different points, contextual information, and the inferences about linking that information, only exist when certain other parameters are also true. Context is a personal model of information, which depends on the state of all of the participants and the user, as well as even more abstract factors. For this reason, it is desirable for the context mediator to expose a powerful user-defined reasoning interface. The different categories of behaviour that pervasive contextual systems support are not all compatible with one design [Bolchini et al., 2007], but the use of runtime reasoning, as well as some assumptions about the nature of the participants, will provide the basis for designing a reasonably flexible mediator. User-defined reasoning should be able to take advantage of the use of external knowledge itself, and should be, where possible, reusable. One difficulty associated with not providing a fixed model is that it is difficult to provide suitable tools for the creation of useful shared semantic views. However, if the interface for user-defined reasoning is not only expressive, but composable, some advantages of reuse become feasible.

4.5 Operational Requirements

This section will define some of the operations which might be used to perform context integration to enrich the knowledge of a target application. While the assumption is made that an ontology exists to represent the knowledge which a participant has, the reality is that the majority of sources especially will likely not be fully adaptive applications.

4.5.1 Types of Operations

One category of integration operation is in the form of altering existing knowledge. The operations in this category include:

- **Addition**: where a value which is not known by the Target Application is known by the sources. This can be considered as the basic operation of enrichment.
- **Deletion**: is a stronger assertion, where the Mediator removes one or more values from a Target's ontology. An example operation in this case would be

to remove some items from a list based on contextual criteria.

- **Update**: is a combination of the two previous operations, where the knowledge in the sources overrides the knowledge in the Target.

These operations are compatible, for the most part, with conventional applications. It is possible for the data values in the Target's ontology to represent the state of functional elements of the application. In this case, the addition, or more likely update of a particular field might result in a behavioural side-effect in the system. This category of behaviour can be seen in the literature in the management of the state of the meeting agent in the EasyMeeting application [Chen et al., 2004a].

A second category of integration operations involves the creation of new knowledge within the Target Application's models. This differs from the previous category, in that it is specifically the creation of new concepts within the Target Ontology, in order to be able to include new information.

The operations involved are similar, but in this case the requirements for the participants change. Many applications will not be able to inculcate new knowledge in this form, and it is necessary to consider more semantically aware participants for this to be useful. At a minimum, however, the knowledge does not have to be totally new. In fact, totally new information is likely to be extremely difficult to integrate automatically. On the other hand, one mechanism that is likely to be viable is to add concepts to the ontology which are not gathered or used, but where provision has been made for their inclusion by the developers of the Target Application.

4.5.2 Discovery Requirements

Service discovery is the process of finding, configuring and communicating with other services within an environment [Zhu et al., 2005]. Apart from the process of discovery itself, there are complex considerations with regard to state management, invocation and orchestration. Some of these techniques can be automated, and some are manual.

For the purposes of the system presented in this thesis, discovery will be the process of interrogating participant ontologies for the concepts which they hold, and correlating them with mapping representations to form integration pathways. The overall process for the context mediator described in this thesis includes the use of potentially complex matching processes, which do not operate automatically. A manual service selection mechanism will be chosen as the initial basis for service management in the context mediator, with the context integration process being automated where possible.

4.5.3 Privacy Requirements

An ontology mediation approach such as the one defined here does not lend itself easily to the application of a deontic privacy policy model such as that provided by CoBrA. This arises principally because the use of ontology matching without a pre-defined set of concepts means that it is considerably more difficult to decide what concepts are or are not sensitive.

This design assumes that privacy is managed at the service level: any information rendered to the context mediator is done so on the basis of it being shared information, and should be controlled at the source of the information. For an example of this in practice the FireEagle location service previously mentioned (*c.f.* Section 4.4.2) includes extensive methods for defining the way in which location information is shared with different collaborating services.

4.6 Architecture

This section will define the principal components of the architecture for a context-informed mediator. The objective of the architecture is to fulfil the design goals listed above. In summary, these goals include the use of ontologies to describe participant knowledge, the use of a Shared Semantic View structure to describe the integration pathways, and supplementing ontological reasoning with user-defined reasoning.

In summary, the design approach of this system is to take the key benefits of previous approaches: a mediation model, ontological reasoning and knowledge transfer supplemented with user-defined reasoning. The system described in this chapter will extend from these systems by employing an ontology mediation approach, with a novel representation for the mediation pathways. This approach will not depend on an upper ontology described beforehand, and will employ a representation that can be altered and exchanged. The system will be targeted on applications which can include a degree of personalisation or adaptivity.

The design is innovative in the way it is applied to context as a mediator based on semantic interoperation; it does not require an *a-priori* model of context. The minor innovation in the design is that it integrates native ontological reasoning with user-defined reasoning and that it represents a shared semantic view of the concepts within the participating ontologies using a separate semantic structure.

4.6.1 Overall Architecture

There overall architecture for this system can be described as having three principal components:

- **The Shared Semantic View Manager** which is responsible for the management of the representation of the articulation of the ontologies. This component provides access to the mappings which represent the integration pathways, as well as references to the concepts themselves. The links within the Shared Semantic View need to be able to represent the different types of relationship: semantic relationships, such as equivalence and subsumption, as well as mechanical relationships such as data transformation.
- **The Schema Manager** is the place where the ontologies from producers and consumers of context are held. The Schema Manager provides access to the entities which are represented in the ontologies as well as the information about particular values within the ontology. This dual role means that the schema manager is responsible for the ontological reasoning necessary to execute

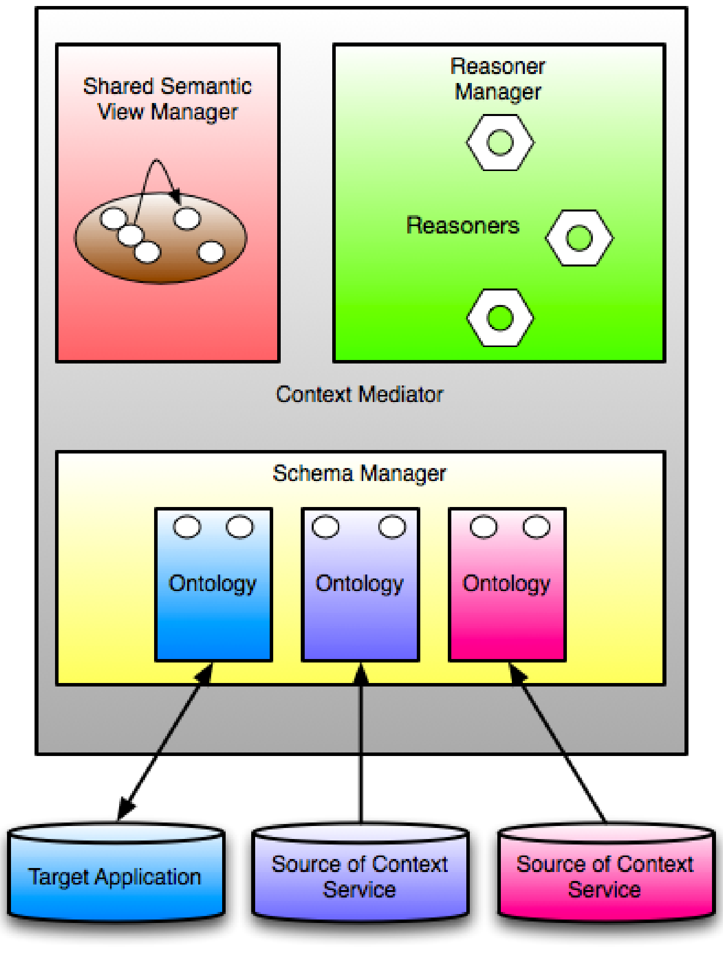


Figure 4.3: Overall Architecture Diagram showing the main components of the system.

queries against the ontological knowledge base , as well as providing descriptive information of the knowledge schema. The Schema Manager needs to be able to represent at least one ontology for each participant, independently and each with possible different formats.

- **The Reasoner Manager** is a repository for individual user-defined reasoner components. These are executable pieces of code which perform different tasks. Reasoners resemble mediators in the WSMO architecture. There are several reasoner functions, such as uplifting ontological concepts, transferring information from one type of concept to another, constructing an enriched target application ontology and more. The Reasoners are designed to be composable, to allow one reasoner to call another, and can be called at runtime.

4.6.2 Architecture of the Shared Semantic View Manager

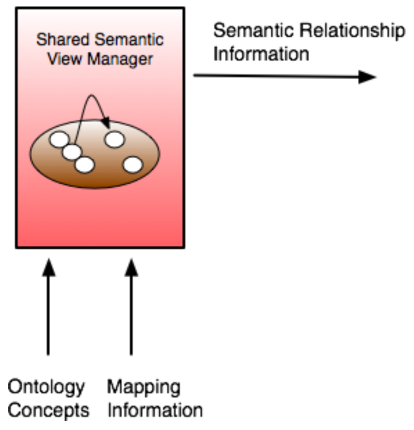

Shared Semantic View Manager
Semantic Relationship Information
Ontology Concepts
Mapping Information

Figure 4.4: Shared Semantic View Manager Component Diagram, with Topic Map pictures. White dots represent topics, and the arrow represents a mapping.

The Shared Semantic View Component of the system is concerned with the creation, modification and management of instances of Shared Semantic View representations. In choosing a representation, it is necessary to examine the requirements for that component. The Shared Semantic View's function is to express the ontological links derived from an external mapping process for use in mediation, and to further characterise those links as required in order to be able to perform identification, semantic translation and syntactic transformation. These requirements point to the need for an expressive structure, one which is capable of classifying and linking ontological concepts.

The use of Topic Maps [JTC1:SC34, 2002], seen in one form in the Business Maps [de Graauw, 2002] technology, would appear to be a suitable method for attempting to express a shared semantic view. However, instead of attempting to engineer a specific map *a-priori*, the objective in this design will be to establish a map programatically, based on the alignment, import and other modification tasks associated with integration.

The intent of using a Topic Map-based approach to representing the ontology is based on a desire to create an articulation between the Target Ontology and several Source of Context Ontologies. Topic Maps permit the representation of these in a lightweight form, that is not constrained by the strict requirements of OWL or other reasoned ontologies. This has the advantage of permitting the Shared Semantic View, represented as a Topic Map, to express a variety of levels of integration pathway, with different levels of abstraction, in a unified structure.

Topics, Associations and Occurrences can be used to describe the linkages within a heterogeneous collection of views — the various participant ontologies. It is further possible to use the type model in Topic Maps as a medium for recording and establishing the characteristics of different concepts and their relations, and thus provide a basis for user-defined reasoning. There are three main components to a Topic Map [Pepper, 2000]: Topics, Associations and Occurrences.

- **Topics** represent atomic concepts. The majority of Topics in this system will be uplifted from the concepts known to different ontologies in the Ontology Manager. This might include representations of entities, as well as their attributes (classes and properties in OWL).

- **Associations** express integration pathways. These represent the relationship, direct or indirect, between concepts in different ontologies. Different types can represent different relationships, such as those relationships described in C-OWL.

- **Occurrences** are links to external representations of a Topic. These can be direct, in the form of URNs for OWL Classes, or could in principle represent more abstract references[6].

Each of these can be given a type, which are categorisations formed on the basis of other topics. This creates the useful property that it is possible to create a closed-world representation model, where themes and types are also expressed with the map. By expressing the Shared Semantic View as a Topic Map, or collection

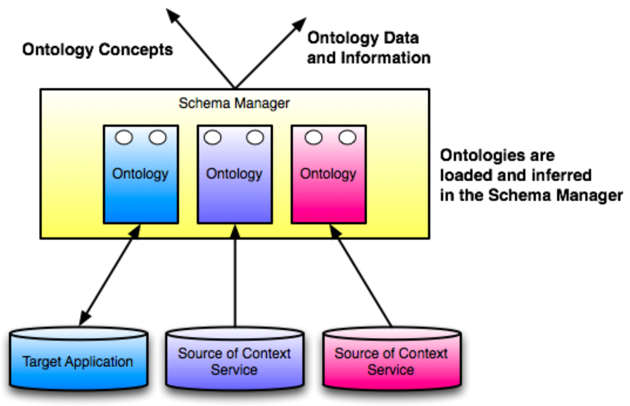
[6]In general Topic Maps, off-line references have been used as the contents for occurrences.

of Topic Maps, the system described here gains some of the benefits of a classical upper-ontology approach, such as an executable, query-able knowledge structure to represent the collective knowledge of the system, without the drawback of needing to establish a specific conceptual framework *a-priori*.

The structure of the Shared Semantic View component surrounds the Topic Map Manger, which is a generic tool for creating and using Topic Maps. The specific Topic Map operations are abstracted somewhat for use by the Reasoners, with the intention of allowing the reasoners to retrieve and alter the different maps' contents.

4.6.3 Architecture of the Schema Manager

The Schema Manager forms the layer which communicates with the participants. The primary responsibility of the Schema Manager is to collect and load the ontologies offered by the participant services. These ontologies need to be loaded using appropriate ontological reasoners for their format and type. The Schema Manager can be used by user-defined reasoners to access ontological schema information and instance data. The Schema Manager accepts Ontologies from all participants, and returns enriched ontologies to the Target Application. One of the challenges



Ontology Concepts
Ontology Data and Information
Schema Manager
Ontology
Ontology
Ontology
Ontologies are loaded and inferred in the Schema Manager
Target Application
Source of Context Service
Source of Context Service

Figure 4.5: Schema Manager Component Diagram. White Dots represent ontological concepts, and the coloured rectangles represent the loaded ontologies.

associated with implementing the schema manager is the degree to which different ontologies can be abstractly accessed by the user defined reasoners. In practice, it is likely that different reasoners will be needed to deal with the differing capabilities of each ontology type supported.

4.6.4 Architecture of the Reasoner Manager

The Reasoner Manager maintains the list of available user-defined reasoners, as well as providing an interface for setting parameters and preferences in different reasoners. The objective of this architecture is to permit the user to define high level reasoners, which can be decomposed into lower level operations as required. This is made possible by the facility for run-time instantiation and execution of reasoners. User-defined reasoners execute all of the tasks within the system not handled either by the Topic Map Engine, or the Ontological Reasoners. Reasoners interface with the Shared Semantic View Manager, and the Schema Manager to execute different tasks, as required. The reusable nature of individual reasoners makes it possible to choose specific operations in response to the run-time state of the Shared Semantic View.

4.6.5 Information Flow

Information flows from the participants to the ontology manager through ontology files. Reasoners then uplift certain concepts from the ontologies to be represented in the shared semantic view. The mappings are taken from external files, and another reasoner represents them in the shared semantic view. Other reasoners process the Target Ontology to identify its knowledge requirement, and use the links in the Shared Semantic View to gather the information from the source ontologies, processing it via other reasoners as necessary. The result is then returned to the Target Ontology, which is returned to the Target Application.

Fig.4.6 shows the information flow of the system. The ontologies are loaded by the Schema Manager, with associated inference as required. The sequence is as follows:

1. **Reasoner A** Imports concepts from the participants' ontologies, and represents

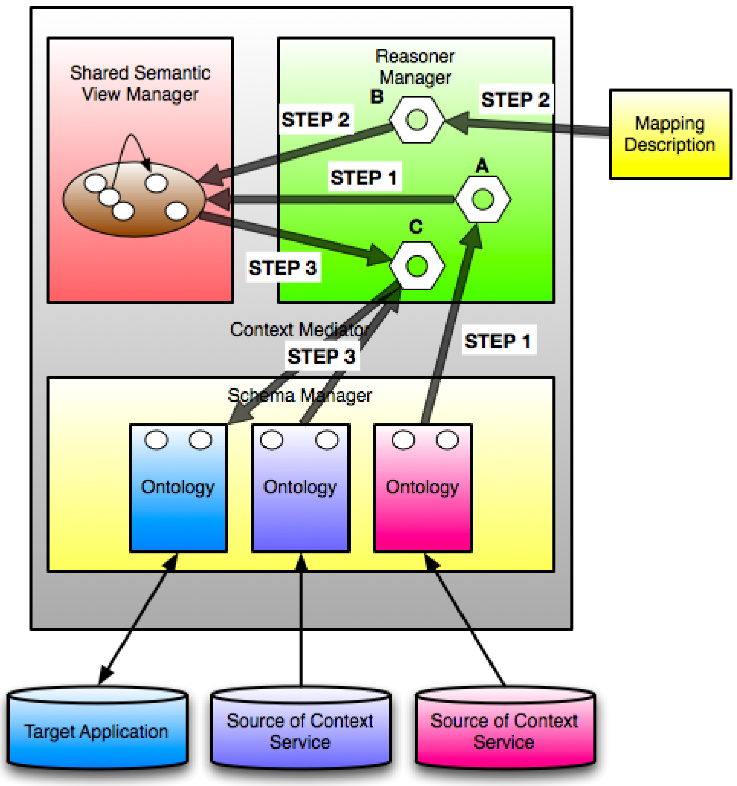


Figure 4.6: Information Flow Diagram.

them in the Shared Semantic View. It also creates types for describing the concepts.

2. **Reasoner B** Creates associations between the concept topics, as well as the type topics needed to describe the relationships, based on the mapping information which was output from an external mapping tool.

3. **Reasoner C** Identifies context information to be transferred, based on the information in the Target's Ontology, and based on the mappings and their descriptions. Reasoner C pulls information from the participant ontologies as required, transforms it, and adds it to the Target Ontology.

This ontology can then be returned to the Target Application.

This process is described diagrammatically in **Fig. 4.7** and **Fig. 4.8**.

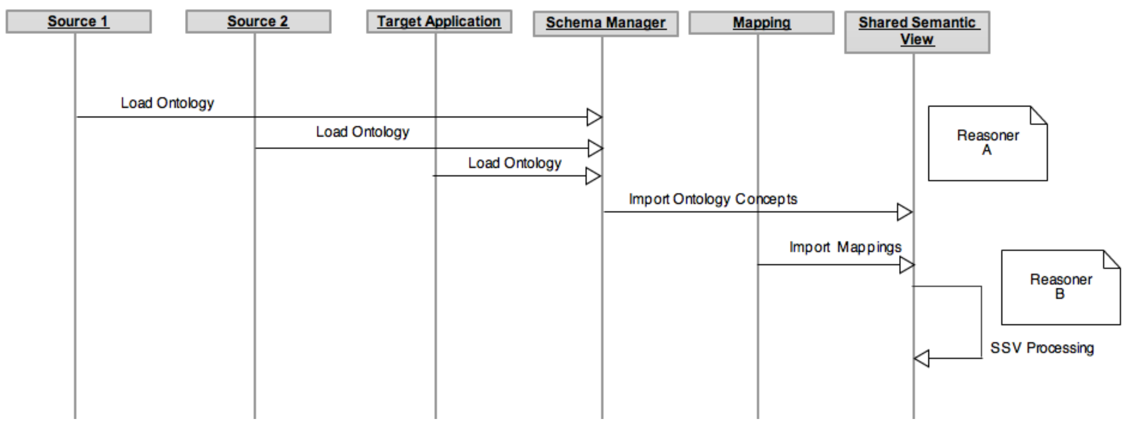


Figure 4.7: Sequence diagram representing the information flow for the first two steps in the process.

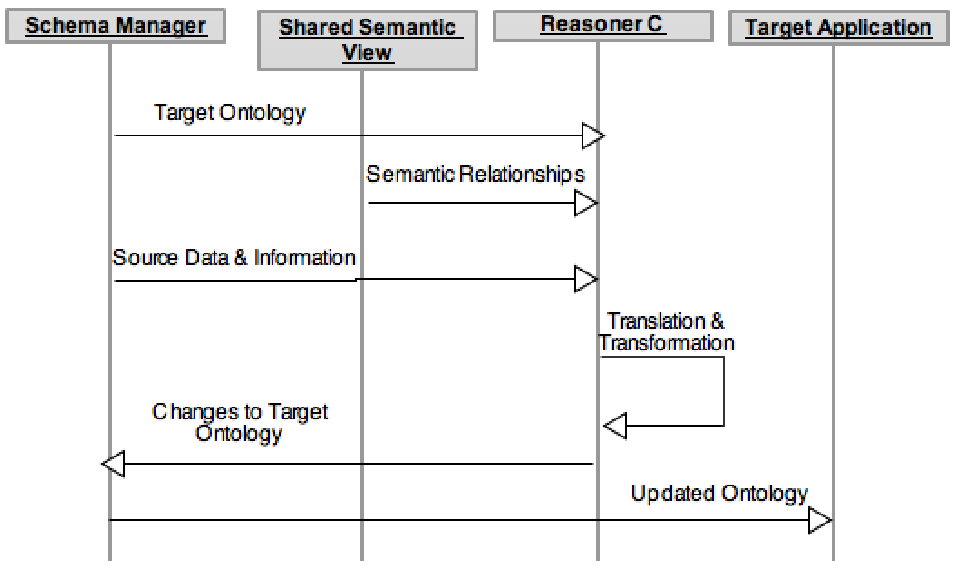


Figure 4.8: Sequence diagram representing the information flow for the third step of the process.

To give concrete examples of user-defined reasoners of different functions, some example reasoners might include:

- An ontology importing reasoner, which creates topics which refer to each Class and Property in an OWL ontology.
- A mapping reasoner, which uses an external vocabulary (such as an ontology) to generate and characterise links between concepts in different services based on an RDF-based description file which was generated externally.
- An enrichment reasoner, which has a complete description of the process of identifying, semantic and syntactic translation and transformation of participating ontologies. This complete reasoner might be able to reuse other more specialised reasoners such as those two above.
- A reasoner which alters the description of the alignments and concepts in the Topic Map, for example by tagging properties in a document ontology with the languages found in the instances of that service. This is a good example of the way the ACP can access schema and instance-related data in order to better describe the shared view represented in the Topic Map.

This thesis does not present a particular taxonomy of functionality for user-defined reasoners. Different reasoners can be created, some of which are general, and some of which are specific. Reasoners in the system can be called in sequence, which facilitates reuse.

4.7 Conclusions

This chapter has presented the process by which the key influences of the states of the art chapters have helped to defined the design goals of a context-informed mediator. The mediator acts as a central point for the transfer of knowledge to a target application from the sources of context, and does so without the use of an a-priori model for context. The system inherits from and extends current work in semantic mediation, by attempting to deal with the key challenges that context created, especially those of heterogeneous data and representational differences.

A key feature of the design of the system is that it represents the conceptual mapping between the sources and the target application separately from the internal ontological representations and the functional exchange process. The Shared Semantic View provides a rich type system, which is the basis for the exchange process. Functional mappings are therefore supported in the system by decomposing them into the reasoner function, the type and the mapping itself. This allows the system to support extremely rich interactions, and to draw from several sources (such as mapping tools) to execute these mappings.

Below is a table which summarises the main requirements drawn from this chapter, which should be found in the implementation of a suitable mediator system.

Table 4.1: Summary of Requirements Drawn from Design

#	**Description**	**Category**
1.	Mediator Model of Interaction	Architecture
2.	Ontology Reasoning	Ontology Manager
3.	Creation of Knowledge Bases from Ontologies	Ontology Manager
4.	Representation of Mappings and Concepts	Shared Semantic View
5.	Expressing syntactic differences between concepts	Shared Semantic View
6.	Topic Map Manager	Shared Semantic View
7.	Expressing semantic differences between concepts	Shared Semantic View
8.	User-defined reasoning	Reasoner Manager
9.	Run-time calling of reasoners	Reasoner Manager
10.	Identification of contextual need	Reasoner Manager
11.	Resolving semantic differences between concepts	Reasoner
12.	Resolving syntactic differences between concepts	Reasoner
13.	Import Mapping Descriptions	Reasoner
14.	Creation of suitable enriched ontology	Reasoner

Chapter 5

Implementation of the ACP: Adaptive Context Portal

This chapter discusses the implementation of a context mediator, based on the design principles laid out in the previous chapter. The chapter begins by discussing the scope of the implementation, followed by a description of the platform technologies. This is followed by an overview and discussion of the key features of the implementation. Finally, an example walkthrough of the operation of the implementation is provided.

5.1 Introduction

The design principles laid down in the previous chapter describe the nature of a context-informed mediator. The key feature of such a system is that it adopts a centralised mediation model, where the system can express the Shared Semantic View of a particular contextual situation. This means that the mediator has the capacity to combine the knowledge from a set of participants to the context integration, with the objective of enriching a target application's knowledge with information from the sources of context. Contextual information is treated as that information which is held in external sources, but which could be useful to the Target Application which is being enriched by the context integration process.

The mechanism for transferring knowledge from the sources of context to the target application is the use of a Topic Map to express integration pathways. Integration pathways describe the data and knowledge relationship between concepts in different ontologies. The pathways are expressed as *categorised* and *reified* associations between topics in the Topic Map. By employing a Topic Map structure, the mediator is able to cross semantic and syntactic boundaries, as well as representing changes in levels of abstraction between different models. This structure also permits the system to incorporate several ontologies and link them using descriptions from a variety of sources.

In order to execute context integration, the system has three key phases:

- **Identification** of information need of the target. This represents an understanding of the difference between the available context and what the target application already knows.
- **Semantic Translation** of the content from the sources. It is necessary to amend the data being transferred so that it agrees with the conceptual representation in the target application, for example by accounting for different levels of abstraction, or different models.
- **Syntactic Transformation** of the expression of the concept. In order for the target application to be able to use the new knowledge, it must be represented in a structure that is understandable by that target application.

The result of this is process is an enriched ontological description of this is the last candidate. next esc will revert to uncompleted text. he knowledge which is available to the Target Application.

A key element of this process is that the use of the Target Application's ontology as a means for expressing and fulfilling information needs. This is the most important part of an informed system: there is no need for the target application to form specific queries, or to be aware of the nature of the contextual environment in which it exists: the context-*aware* environment, through the mediator, *informs* the application.

This chapter will define an implementation of such a context mediator. The chapter will begin by defining the scope of the implementation, and describe some functional and non-functional elements of the system. This domain analysis will be followed by a discussion of the implementation platform. including a description of the key libraries and supporting technologies upon which the implementation is founded. Some key aspects of the design will then be described, with particular attention paid to aspects of the implementation which depart from any standard use of the libraries. These key implementation aspects are then made concrete through the use of an example, and a walkthrough of the implemented system.

5.2 Implementation Scope

There are certain aspects of context integration which this implementation will concentrate on, and others which it will not represent. The objective of this implementation is to form a basis for the evaluation of context-informed mediation, and in this regard the central component of the implementation is that mediator. The principal mediator components are implemented from the design of the system, including the Reasoner interface, Schema manager and Shared Semantic View system.

5.2.1 Ontologies

The conceptual relationship of the mediator to the ontologies it bridges is a key aspect of the system. Ontology Engineering is a rich area of research, with considerable variation in the resulting conceptual representation; there exist a great many models for engineering ontologies, each with different approaches [Cristani and Cuel, 2005]. It is possible to imagine a spectrum of different ontologies, with different properties, but the important realisation is that ontological engineering and data modelling are different[Spyns et al., 2002]. The result of this variation is that some ontologies, such as ones derived from Cyc [Reed and Lenat, 2002] constitute a knowledge base of computational background knowledge, which can be used to compile rules that ultimately simulate intelligence. On the other hand, Gruber's notion of a crystallised conceptualisation [Gruber, 1993] can be regarded as more focused on the transfer of knowledge in a more descriptive form than non-semantic representations. In this case, the value of the ontology is in the fact that the information is accompanied by a structured description of its conceptual framework, which can be reasoned about to generate knowledge. This implementation is focused on knowledge transfer, and so will generally support knowledge-transfer types of ontologies, with more structured relationships and content, than looser, more AI-oriented structures.

The size of ontology schemas is a second aspect of the nature of ontologies. There are some common examples of ontologies with thousands of classes[1]. Very large ontologies incur substantial cost in processing and storage, and so the general assumption of this work will be to focus on ontologies with smaller conceptual bases. There are several ways in which the ACP can approach large ontologies. The first is that the native ontology reasoner profiles can be varied in response to the requirements of particular ontologies. This means that, where suitable, a lightweight reasoner profile can be used in a large ontology to lower the cost of loading. The second support is that the ACP uses a semantic interoperation approach that builds based on the concepts required for a particular situation. This means that portions of a large knowledge base can be targeted for integration into the context scenario.

The development of this implementation will focus on OWL/RDF-based ontologies, with their associated DL reasoners, where applicable. This technology was chosen because it appears to be reasonably widely used, and due to the availability of associated tools.

5.2.2 Service Registration

As discussed in the design chapter, there is no automatic service discovery model for this system. In the implementation of the ACP, a service registration model is chosen. All services are assumed to be web-accessible, as are their ontologies. The

[1]For example, the NCI Human anatomy ontology [Noy et al., 2008] runs to 3304 classes.

system allows for services to be registered and identified by their URL. One of the areas of evaluation of the system will relate to the process of discovering the entities within an ontology, and properly uplifting them to the Shared Semantic View Topic Map. The use of topics to represent ontology concepts and the relationship between them means that there is a wide variety of possible methods to represent different levels of detail in the Topic Map.

One interesting aspect of context is the notion that two services might share the same ontology, for example because they are different instances of the same platform, but not share the same information base. This situation is familiar in particular in the area of user modelling, and presents one reason why the system is designed not to make assumptions about the equivalence of entities which have the same ontological reference. While two instances of a class `userid` could be the same because they refer to the same person, it is also possible, indeed likely, that they could differ in value while still fulfilling all of the consistency of the semantic modelling. The Mediator design can solve this problem by allowing for separate instances of concepts which have the same external (ontological) definition. The Topic Map accounts for topics with the same name by merging the label, but allowing for separate underlying topics to remain.

This issue is particularly prevalent in the mediation of context, because the context mediation process involves sharing information across a wide field of participants. The mediation model addresses the need to be able to express data, as well as knowledge differences, by virtue of the fact that the participant knowledge is only merged into a virtual document, rather than a concrete merger of the models.

5.2.3 Privacy

Also following from the design chapter, this implementation will not control access to data for privacy or security purposes. Part of the objective of this design and implementation is to create system that provides a flexible toolkit for composable user-defined reasoning, and in attempting to reach this objective, the semantics of enforcing privacy policies have not been included. It would likely be possible to create a deontic policy enforcement service as a component reasoner, but that is out of scope for this research. There is an assumption, therefore, that the contextual information which has been published has been exposed with the intention of sharing it.

5.2.4 Abstraction

One of the key difficulties in implementing a system such as this is the abstract modelling of ontological concepts, particularly the interface between the reasoners and the different ontologies. There is a distinct challenge in attempting to encapsulate all of the features and properties of different ontology languages in such a fashion as to make them equivalent. It is likely that there will be some breach of this abstraction. This may be an unavoidable consequence of the differences in various ontological representations.

5.3 Implementation Platform

This section describes the technologies that are used in the implementation of this system, that were developed externally. The objective of this section is to discuss some of the advantages of the chosen technologies, as well as some of the complications that arise from their use.

The primary implementation language for the ACP is the Java programming language. Java was chosen because of the wide availability of libraries, particularly libraries arising from academic research. Java 5 [Sun Microsystems, Inc., 2004] is the target implementation version. This allows for the use of Java Generics, and the expanded Collections framework, as a means for structuring the encapsulation and instances of classes and objects in the system. The use of typed collections is a particular advantage of this implementation, although the many of the supporting platform libraries discussed below target Java 1.4, which means that they do not support the generics model. Other features of Java 5 which proved useful included the improved for-each loop construct, and the auto-boxing of primitives.

There are two libraries which are used to provide access to the semantic structures used in the ACP. There is considerable complexity in the functionality of these libraries, and their use allowed for the creation of a system which addressed these semantic structures from a conventional starting point. Ontology management in the ACP, handled through the Schema Manager, makes use of the Jena [Carroll et al., 2004] ontology library. Jena provides features for loading, parsing, querying and modifying ontologies. Different ontological reasoners can be used, including different levels of rule inference, as necessary. The library provides some tools for managing a number of ontologies loaded simultaneously, and this functionality is enhanced by the Schema Manager.

Shared Semantic Views are represented in the design of this system by Topic Maps. In order to be able to make use of standard Topic Map features, the TM4J [Ahmed, 2004] library was used. While there is an initiative to standardise programmatic access to Topic Maps [Heuer and Schmidt, 2008], in this implementation, the native methods for the library were used, in order to gain full access to the features of the Topic Map Engine. The TM4J engine provides methods for creating topics, occurrences and associations, and for copying topics between maps. Of particular note is the fact that TM4J supports persistent storage of Topic Maps through the use of the Hibernate [Redhat Middleware, 2004] persistence provider system. This is the mechanism by which SSV persistence is achieved in the ACP, allowing maps to be preserved between executions of the reasoners.

5.4 Implementation Overview

The architecture for the ACP is based on the design architecture previously described. This architecture is based on three main functional components:

- a **Schema Manager**, which is responsible for handling the ontologies

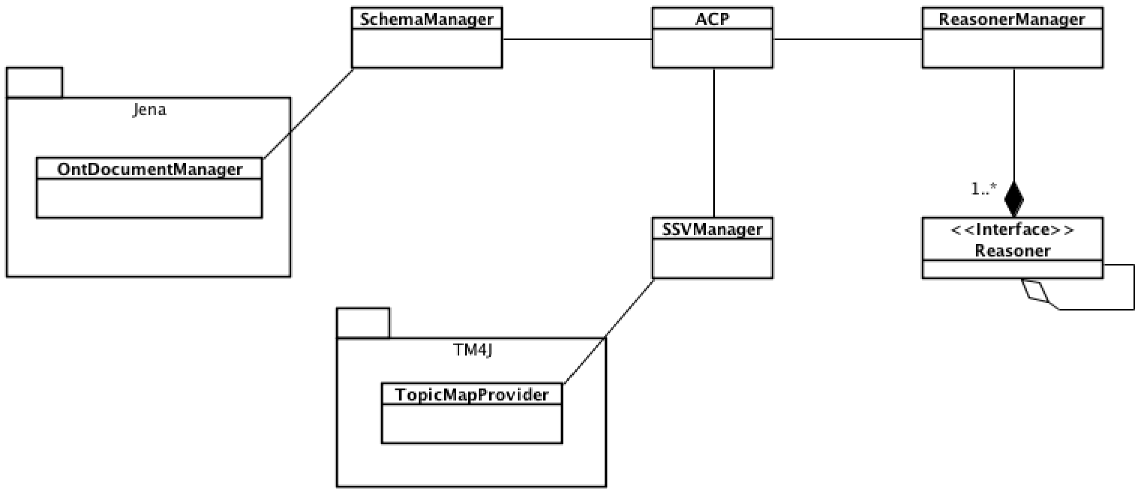


Figure 5.1: Architecture Overview, showing Façade model

describing the participants.

- a **Shared Semantic View Manager** which maintains the Topic Maps that represent the Shared Semantic View.
- and, a **Reasoner Manager** which manages the user-defined reasoners.

The individual components of the system are represented as classes within the implementation. The implementation is intended to be able to be called in different ways, for example from a J2EE server, or as a server back-end. With that in mind, a Façade design pattern was chosen, where the ACP Class represented the externally accessible functionality of the system, while shielding the internal features of the components[2]. The use of this Façade is to provide a simple interface for calling and managing the deployments of the ACP implementation, it is not a key architectural requirement, rather it is to help deploy different experimental scenarios easily.

The ACP class includes methods for initialising each of the components, and providing them with configuration information necessary to launch each component. The ACP also provides the appropriate methods for querying the state of the Topic Map, for example there is a method which returns which topics are loaded in a particular map. Similarly, there are methods associated with loading different ontologies and

[2]For more information on the Façade pattern, see [Portland Pattern Repository, 2009]

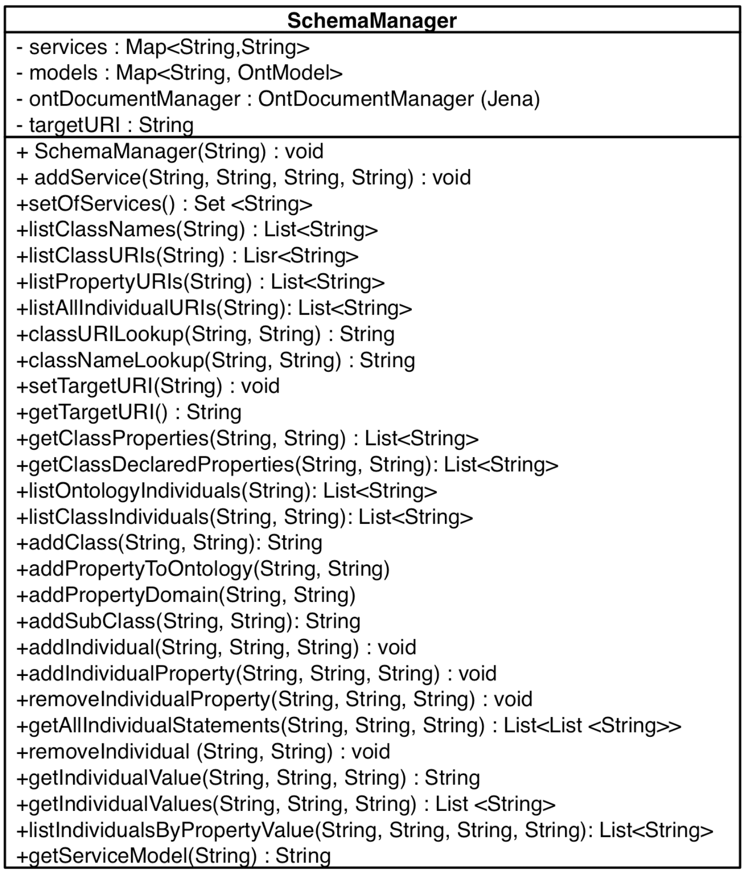
services, which are associated with the Schema Manager, and methods for calling and configuring user-defined Reasoners in the Reasoner Manager. For the most part, the Façade methods are pass-through methods to the relevant component.

The information flow between the components is largely governed by the user-defined Reasoners. The Reasoners have access to the SSVs and ontologies, and can query them and make alterations as required. Reasoners have access to the methods of the Schema and SSV Managers – this provides a degree of abstraction from the specific implementation library to the reasoner.

5.5 Key Aspects of Implementation

This section discusses the specific implementation of each of the main components of the ACP: the Schema Manager, the Shared Semantic View Manager, and the Reasoner Manager. Each component is discussed from the perspective of the manager component, as well as the supporting libraries, if any, which are used.

5.5.1 Schema Manager



```
SchemaManager
- services : Map<String,String>
- models : Map<String, OntModel>
- ontDocumentManager : OntDocumentManager (Jena)
- targetURI : String
+ SchemaManager(String) : void
+ addService(String, String, String, String) : void
+setOfServices() : Set <String>
+listClassNames(String) : List<String>
+listClassURIs(String) : Lisr<String>
+listPropertyURIs(String) : List<String>
+listAllIndividualURIs(String): List<String>
+classURILookup(String, String) : String
+classNameLookup(String, String) : String
+setTargetURI(String) : void
+getTargetURI() : String
+getClassProperties(String, String) : List<String>
+getClassDeclaredProperties(String, String): List<String>
+listOntologyIndividuals(String): List<String>
+listClassIndividuals(String, String): List<String>
+addClass(String, String): String
+addPropertyToOntology(String, String)
+addPropertyDomain(String, String)
+addSubClass(String, String): String
+addIndividual(String, String, String) : void
+addIndividualProperty(String, String, String) : void
+removeIndividualProperty(String, String, String) : void
+getAllIndividualStatements(String, String, String) : List<List <String>>
+removeIndividual (String, String) : void
+getIndividualValue(String, String, String) : String
+getIndividualValues(String, String, String) : List <String>
+listIndividualsByPropertyValue(String, String, String, String): List<String>
+getServiceModel(String) : String
```

Figure 5.2: Class Diagram Description of the Schema Manager

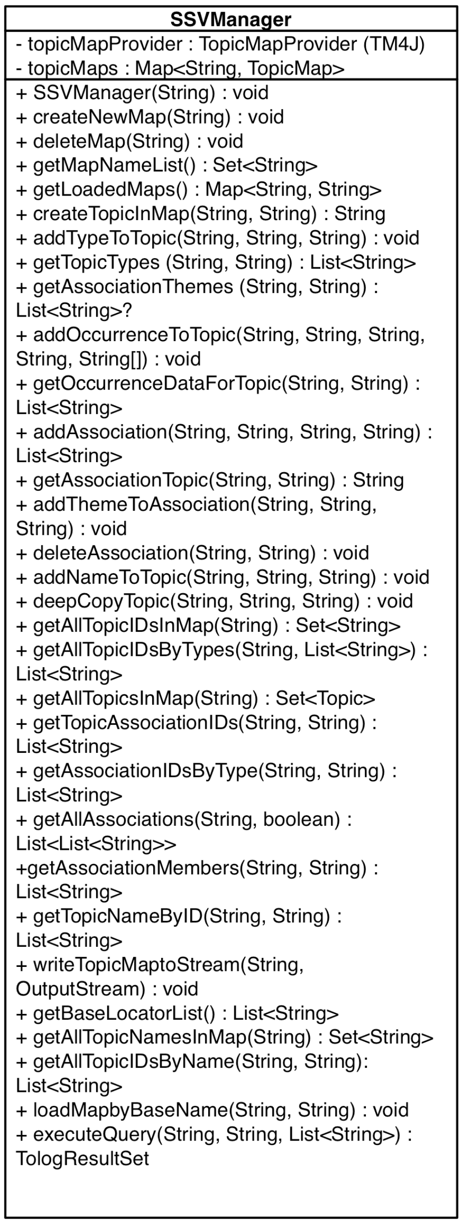
The Schema Manager is responsible for providing access to the ontologies related to participant services. The methods provided for access to schema components broadly follow the OWL semantics, in that they recognise Classes, Properties and Individual instances[3]. The Schema Manager identifies services by URL, and associates ontologies with their URL. This is intended to permit the Schema Manager to represent different versions of the same ontology in different services. The Schema Manager can import ontologies from remote URLs, or can make use of a local file specified by the path.

For this implementation, the Schema Manager defaults to the use of in-memory storage and OWL language. This can be altered in the configuration of the Schema Manager, on a per-service basis. As with all three of these components, there is an

[3]Note the semantics of Classes with properties, instantiated by individuals are common to other ontology languages.

initialisation method, separate from the constructor, used to provide setup variables. This allows the Façade class to load the different components in an order and at a time when it is best suited.

5.5.2 Shared Semantic View Manager



```
SSVManager
- topicMapProvider : TopicMapProvider (TM4J)
- topicMaps : Map<String, TopicMap>
+ SSVManager(String) : void
+ createNewMap(String) : void
+ deleteMap(String) : void
+ getMapNameList() : Set<String>
+ getLoadedMaps() : Map<String, String>
+ createTopicInMap(String, String) : String
+ addTypeToTopic(String, String, String) : void
+ getTopicTypes (String, String) : List<String>
+ getAssociationThemes (String, String) :
List<String>?
+ addOccurrenceToTopic(String, String, String,
String, String[]) : void
+ getOccurrenceDataForTopic(String, String) :
List<String>
+ addAssociation(String, String, String, String) :
List<String>
+ getAssociationTopic(String, String) : String
+ addThemeToAssociation(String, String,
String) : void
+ deleteAssociation(String, String) : void
+ addNameToTopic(String, String, String) : void
+ deepCopyTopic(String, String, String) : void
+ getAllTopicIDsInMap(String) : Set<String>
+ getAllTopicIDsByTypes(String, List<String>) :
List<String>
+ getAllTopicsInMap(String) : Set<Topic>
+ getTopicAssociationIDs(String, String) :
List<String>
+ getAssociationIDsByType(String, String) :
List<String>
+ getAllAssociations(String, boolean) :
List<List<String>>
+getAssociationMembers(String, String) :
List<String>
+ getTopicNameByID(String, String) :
List<String>
+ writeTopicMaptoStream(String,
OutputStream) : void
+ getBaseLocatorList() : List<String>
+ getAllTopicNamesInMap(String) : Set<String>
+ getAllTopicIDsByName(String, String):
List<String>
+ loadMapbyBaseName(String, String) : void
+ executeQuery(String, String, List<String>) :
TologResultSet
```

Figure 5.3: Class Diagram Description of the Shared Semantic View Manager

The Shared Semantic View Manager employs TM4J with a Hibernate persistence back-end. The declaration for Hibernate storage is as follows:

```
SSVManager
TopicMapProviderFactory tmpf =
new org.tm4j.topicmap.hibernate.TopicMapProviderFactoryImpl();
...
topicMapProvider = tmpf.newTopicMapProvider(properties);
```

the properties file argument specifies a set of database connection properties which are necessary to start the storage. The use of this storage means that each time the SSV is initialised, maps must be loaded from the hibernate storage.

One important aspect of the system is that topics & associations in this implementation are referred to using the ID attribute. This is not strictly in compliance with the Topic Map standard. The reason for this implementation decision is that in the programmatic generation of Topic Maps, the selection of appropriate subject indicators is not always available. By using topicIDs, uniqueness of reference is guaranteed. In order to make this decision explicit, the system makes specific reference to IDs in many of the methods which employ them, with the intention of flagging the ID semantics to the user. This mechanism is also important in the case where topics have been automatically merged, in order to retain certainty of reference.

Another factor in the creation of maps is that there are certain functional topics which are used in order to express the Shared Semantic View. In an attempt to keep the system's assumptions as general as possible, Reasoners are required to create the basic topics for the system. One topic which is common in many of the Reasoner implementations is the "Type" topic, which is used to indicate topics that are descriptive types, rather than ontological concepts, in the map. It is then the responsibility of each Reasoner which alters the map to ensure that its types exist on the map, and that it does not create duplicates. For example two types for reasoners could include the equivalence and subsumption relationships found in a mapping description, and the OWLClass type, used to designate OWL Class concepts. During the ontology uplift phase, the Topic Map would likely not include these types, and so the OWL import reasoner would be responsible for creating the OWLClass type, which would then be used on topics which refer to OWL Classes. The alignment import reasoner is used to import an alignment description, the alignment reasoner needs to create the subsumption and equivalence type topics. These reasoners both have roles in creating the Topic Map state which can then be used by a transfer reasoner. If the transfer reasoner does not properly look up the type topic associated with OWL Classes and the relationships, then the transfer reasoner will not be able to find the appropriate concepts and associations to perform the transfer. In each map, the specific ID of each topic type is different, but it can be found easily by looking up the name and/or types of a concept.

Reification of Associations

Early versions of the ACP system design used the association type to designate the relationship (such as subsumption) between the topics. Topic Maps only permit the use of a single type with an association. As the system was further developed, it became desirable to be able to add more information about the properties of a link to the Topic Map.

The traditional representation of several properties of an association is through the use of several separate Associations, each with the appropriate type. This was not satisfactory, given the importance of the metadata associated with semantic mappings. In consultation with the TM4J Developer, Kal Ahmed, the author of this thesis were able to decide on the use of reification as a medium for creating topics which describe the association. This was achieved by the author of this thesis by using the association ID as a subject indicator in the reifying topic. This topic can then be used to describe the association's attributes. Note that in the code below, which describes the retrieval of the association topic, some extraneous code has been removed.

```
SSVManager
public String getAssociationTopic(String mapName, String associationID){
        Locator topicSubjectLocator =
        tm.getLocatorFactory().createLocator("URI", associationID);
        Topic associationTopic =
        tm.getTopicBySubject(topicSubjectLocator);
        topicID = associationTopic.getID();
return topicID;
```

5.5.3 Reasoner Manager

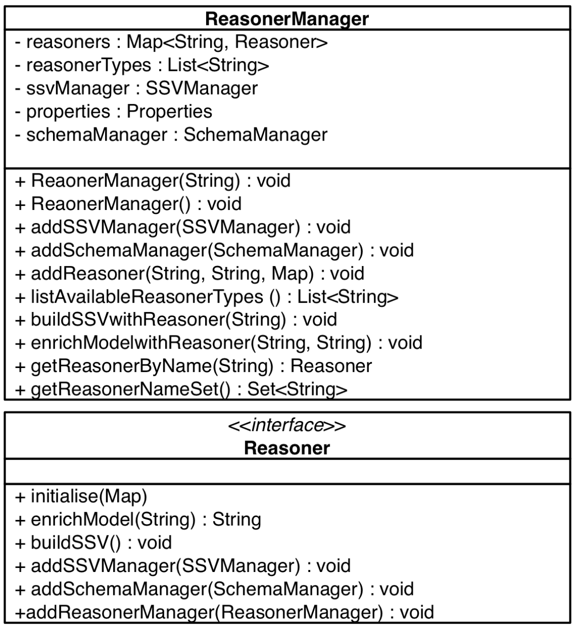

ReasonerManager
- reasoners : Map<String, Reasoner>
- reasonerTypes : List<String>
- ssvManager : SSVManager
- properties : Properties
- schemaManager : SchemaManager
+ ReaonerManager(String) : void
+ ReaonerManager() : void
+ addSSVManager(SSVManager) : void
+ addSchemaManager(SchemaManager) : void
+ addReasoner(String, String, Map) : void
+ listAvailableReasonerTypes () : List<String>
+ buildSSVwithReasoner(String) : void
+ enrichModelwithReasoner(String, String) : void
+ getReasonerByName(String) : Reasoner
+ getReasonerNameSet() : Set<String>
<<interface>>
Reasoner
+ initialise(Map)
+ enrichModel(String) : String
+ buildSSV() : void
+ addSSVManager(SSVManager) : void
+ addSchemaManager(SchemaManager) : void
+addReasonerManager(ReasonerManager) : void

Figure 5.4: Class Diagram Description of the Reasoner Manager & the Reasoner Interface

This section will describe the Reasoner Manager component of the ACP, and discuss the implementation of it and the Reasoner interface which is used in common by all reasoners in the ACP. The Reasoner Manager is designed to be able to load and run Reasoner instances at runtime. The system uses the Java Classloader to load instances of classes. Because of this, classes in the Reasoner must be accessible through the deployment classpath. In order to facilitate discovery, the system configuration file for the Reasoner manager includes a list of available reasoners. This is a manually maintained list, in order to permit the person deploying the ACP instance to control which reasoner classes should be able to be loaded. Each Reasoner can be given a string map of parameters, which is a set of key:value pairs used in configuring the system, and reasoners can have references to the components as necessary. Several instances of the same reasoner can be created, with different parameters, as required.

ReasonerManager

```
public void addReasoner
  (String reasonerName, String reasonerType, Map reasonerParameters)
 throws InstantiationException, IllegalAccessException,
 ClassNotFoundException{
        Class ReasonerTypeClass = Class.forName(reasonerType);
        Reasoner reasoner = (Reasoner) ReasonerTypeClass.newInstance();
        reasoner.initialise(reasonerParameters);
        reasoner.addSchemaManager(schemaManager);
        reasoner.addSSVManager(ssvManager);
        reasoners.put(reasonerName, reasoner);
}
```

The Reasoner Manager, and the Reasoners, access the internal methods of the components, and do not use the Façade ACP class. The Reasoners themselves are implemented in Java. This has the advantage that the Reasoners have full access to the capabilities of Java for engaging in document parsing, string manipulation or making use of external connections to additional local or remote functionality. However, there is little security associated with this model, and badly-written or intentionally malicious reasoner could cause considerable harm in the system. Reasoners can be loaded and called externally, through the Façade class.

Reasoner Interface

The Reasoner Interface is implemented by all Reasoners in the ACP. There are three methods, which classify the functionality of the reasoner. These methods are extremely generic, and the design of the reasoner system relies on the Reasoner using its parameters to specify the details of the operations, where necessary. Each of the components can be connected to the Reasoner, as necessary. Different Reasoners will likely need different access to the parts of the system.

Reasoner

```
public interface Reasoner {
        public void initialise(Map parameters);
        public String enrichModel (String model);
        public void buildSSV();
        public void addSSVManager(SSVManager ssvManager);
        public void addSchemaManager(SchemaManager schemaManager);
        public void addReasonerManager(ReasonerManager reasonerManager);
}
```

5.6 Implementation Walkthrough

In order to provide an understating of the operation of the ACP system, this section will present a simple example scenario of use and demonstrate the action of each of the parts of the implementation. The section begins by presenting the example use cases, and describing some of the possible variations that the system can accommodate. The second section describes the deployment of the participants and the ACP instance. This includes a discussion of the participants, and the platform of operation for the context mediator.

5.6.1 Example Scenario

The 'cold-start' problem describes a situation where a system which is presented with a new user needs to know information about that user. The traditional, manual approach is to ask the user to fill out the details themselves, but this can be a tedious process, particularly if there is a lot of information to communicate. It is desirable for systems to be able to share information about users. However, this has proved to be a significant challenge because of the differences in representation between different systems and institutions.

This scenario envisions a group of students, who are studying computer science. In the course of their studies, the students travel to a different university, to continue their studies. Each student has been making use of eLearning systems in his or her home university to learn about SQL[4], and will be continuing their studies at the new university, which also makes use of eLearning to teach SQL. However, the difficulty that arises is that the two universities make use of different eLearning systems to record their student transcripts.

The traditional solution to this problem is to elicit the student's knowledge through a questionnaire, or, worse, to make a broad assumption of the student's level, based on the level of his classmates in the new university. A better, more personalised method would be to take the transcript that the student has earned in his home university, and make use of it in the new university's eLearning system.

The challenge of this scenario is to be able to create an accurate representation of the student's learning experience in terms that the target application can understand. The profile stored in the previous learning environment holds this information, but it differs both in terms of the language it uses to describe the topics of learning, and in the representation of that knowledge. The ACP is designed to be able to overcome both of these issues[5].

[4]Structured Query Language

[5]*n.b.* the issue of User Model interoperation is itself a wide area of research, and this example solution does not deal with many of the important factors, such as the difficulty associated with creating suitable ontologies from the participating systems. However, it is a representative example

To put this in terms of a context problem, there exists a target application, the new university's eLearning system, which could benefit from external knowledge about the user (their transcript), but which cannot get that information because it is stored on another, previously unknown system, in a format that is different to the target's native format.

This scenario makes several assumptions, these follow:

1. that the two universities trust each other sufficiently to allow each other access to their student learning data
2. that the students are uniquely identified by a username, which is the same in both universities
3. that each of the eLearning Systems (from the home university, and the new university) have an OWL ontology describing their student user models.
4. that there exists a mapping between the two ontologies, which appropriately describes the semantic relationship between the two ontologies
5. that a table has been created which maps the subject labels from each of the eLearning systems to each other. This describes the data relationship.

5.6.2 Service Description

The service environment for this scenario is composed of the Target Application, which is the eLearning system of the new university, and one Source of Context, which is the user model service of the students' home university. The new university IT administrator is able to make use of an ACP instance to transfer the required information. The information transfer requires the source of context to make its ontology model accessible, so that it can be loaded by the ACP, while the target

of the sort of information which context must cover: the information is useful, relevant and due to the wide number of different mapping combinations, extremely difficult to approach within the participant knowledge architectures.

application must be able to share its ontology and also accept the resulting enriched ontology and make use of it.

The two ontologies in this process are outlined below, along with their alignment. The main differences are in the naming of the labels of the different parts of the models. For this example, each ontology has one class, $User/Learner$, which represents the

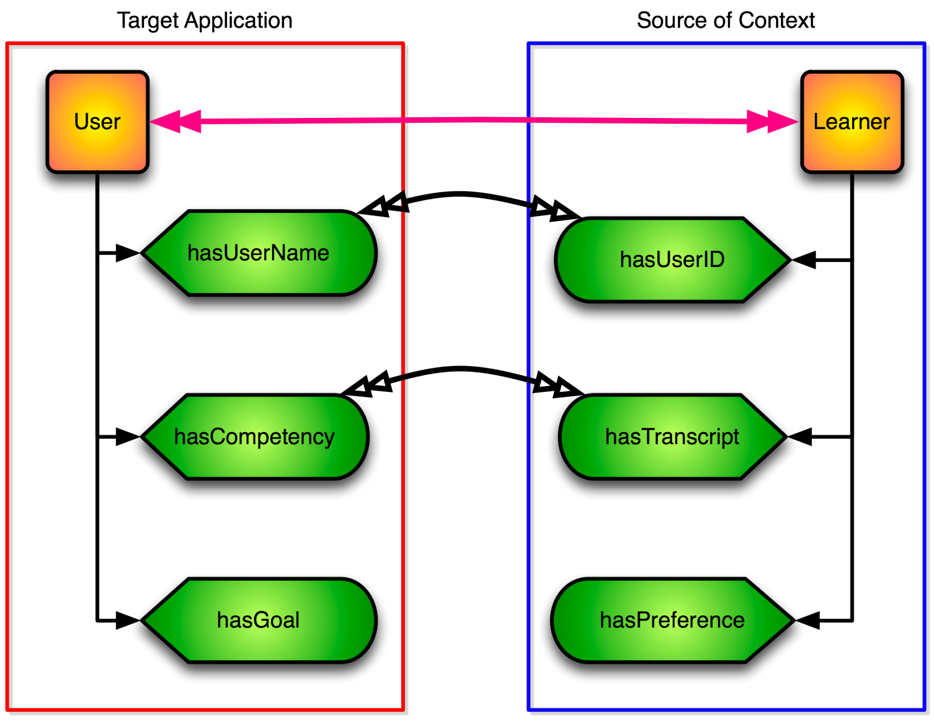


Figure 5.5: The example ontologies. The Target Ontology is boxed in red on the left, the Source Ontology is boxed in blue on the right. Classes are represented by the yellow boxes, datatype properties by the green capsules. The black arrows indicate property data equivalence, while the pink arrow indicates class equivalence.

profile of one student. Each Class has three data properties, which contain the information in the profile. Each student is represented as an instance of the relevant class in each ontology. The $hasUserName$ property in the Target Ontology is the only field which is complete initially in the Target Ontology. This username is used to select the appropriate equivalent instance in the Source Ontology to be transferred.

The bulk of the user model information is held within the $hasTranscript$ property of the Source ontology. Each Instance will have several values stored in this property, each value being a topic that the student has successfully learned, such as 'sql.table.create' for the knowledge of how to create an SQL table.

Two properties, $hasGoal$, and $hasPreferences$, do not have equivalents in the other ontology. $hasPreferences$ is a property which describes the educational preferences of the user, such as the fact that they have a preference for visual learning materials over text-based ones. A novel feature of the ACP is that it attempts to make the existence of this new knowledge available to the target application, which might be able to make use of it. This is achieved by adding the $hasPreferences$ property to the Target ontological schema, and transferring the relevant instance information.

The Operation of the overall process will be governed by a UserModelTool reasoner, which is designed to transfer the information. The UserModelTool implements the Reasoner Interface, and has explicit methods for the different transfer phases. This reasoner is supplemented with a Reasoner called SchemaIntegrator, which is responsible for representing the OWL schema in the Topic Map, and AlignImporter, which is responsible for representing the mapping description as recorded in an INRIA Align RDF file.

5.6.3 Ontology Uplift

This section describes the process of creating a representation of the knowledge of each of the participants' ontologies in the ACP. The first phase of the context integration is to import the ontologies themselves, in their native forms, through the Schema Manager. The Schema Manager can load several ontologies at once, and the Jena library permits different reasoner specifications and different storage models to be used. This allows the system to accommodate the different OWL and RDF dialects available to Jena.

The ACP Facade is instantiated, and the Schema, SSV and reasoner managers are initialised. The main class which calls the ACP facade then instructs the ACP to load the Ontology schema files. These are .owl or .rdf files containing the xml serialisations of the ontologies. It is also possible to load copies of files over the network, if necessary.

Once loaded, the ontologies are accessible to the Reasoners through the SchemaManager methods. The methods allow the listing of classes and properties, and accessing instance values. There are also methods for adding and removing schema and instance elements from the ontology.

In this example, the two ontologies are loaded using the default rule inference profile and in-memory storage. The Ontologies are now ready to be uplifted to the Topic Map.

Once the BuildSSVWithReasoner function is called, the UserModelTool Reasoner loads and initialises the SchemaIntegrator reasoner. This reasoner is given a list of services to integrate, as well as the details of the SSV Topic Map. If it does not already exist, the map is created by the SchemaIntegrator, which also creates type topics used to designate the ontological role of the different entities. These types include OWLClass, OWLProperty, and a Type Topic.

Beginning with the classes, the SchemaIntegrator gets a list of the URIs for the classes from the SchemaManager for each Service. The Reasoner creates a Topic for each Class in the ontology, and adds appropriate type information marking it as an OWL class. In addition, the reasoner creates an Occurrence in the topic, which points to the URI. A service Topic also designates the service membership of the entity.

Once the Classes have been uplifted in this fashion, a similar process occurs for the Properties. The result of this complete process is a Topic Map which contains a topic for each class and property in the map, as well as type topics which describe those entities, and with attached occurrences describing their defining URI. A graphical representation of the uplifted Topics and their sources is shown in **Fig. 5.6**. Note that the complete map would include more type topics (in the middle of the map below) to represent the service which each topic was uplifted from. The transformation in this case has been to create the nodes in the semantic knowledge map: this includes the entities known to the two ontologies, and the type topics necessary to describe

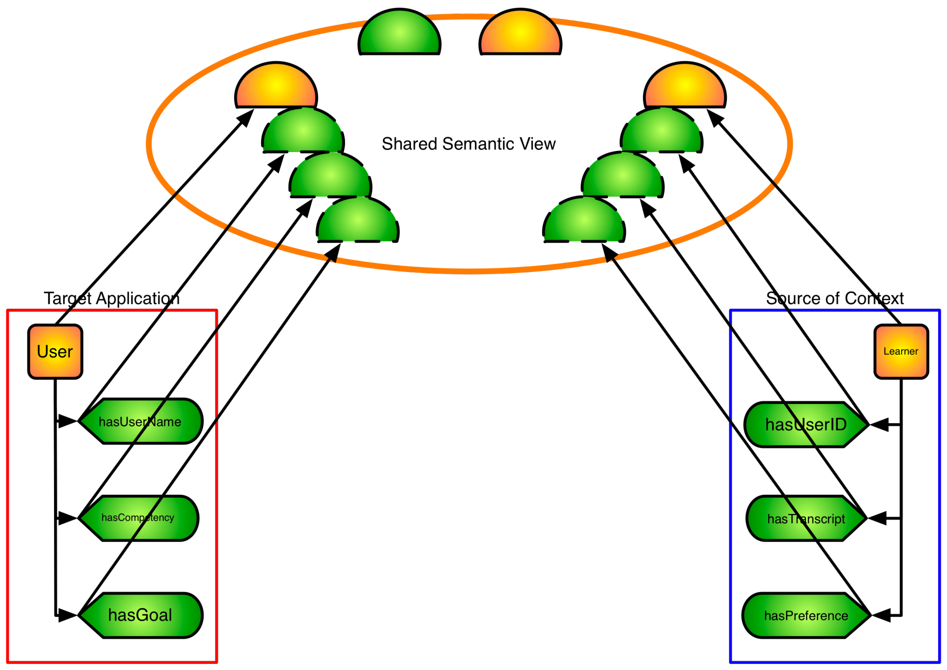


Figure 5.6: The result of the uplift process. The uplifted topics and the ontologies are pictures. The Topics representing properties have dashed lines. Note that additional type information about the service for each topic is not pictured.

them. URIs are guaranteed to be globally unique, while entity names are not, and there is the potential for considerable confusion if the short names of the entities were used, then there is an increased chance of confusion between semantically unrelated entities.

Alignment Import

Once the base map has been established, an alignment file is used to represent the mappings which have been found between the ontologies. The align RDF defines the parties to each mapping, as well as a mapping operation such as = (Equivalence), and a confidence interval of the mapping (usually 1.0). In this example, the mappings are all of the Equivalence type.

The UserModelTool calls the AlignImporter reasoner, which loads the map created by the SchemaIntegrator. It adds its own types to the map, using the previously-established type topic to indicate that these are type topics. The AlignImporter creates topics to describe the Equivalence, Subsumption and inverse subsumption relationships. In addition, a five star model is created, which is used to represent the confidence interval (5 stars = 1.0 confidence).

The AlignImporter Reasoner reads the description file, and for each topic which it can find both members for, it creates an Association between them. The Association Type is set to be the type of operation as described in the Align file, and the Association Topic (the reified description topic) is given the types appropriate to the metadata about the mapping, including the rating and the fact that it came from an Align file. A partial diagram of the alignments is shown in **Fig.**5.7. **Fig.** 5.6. Note that the complete map would include more type topics (in the middle of the map below) to represent the service which each topic was uplifted from. This second

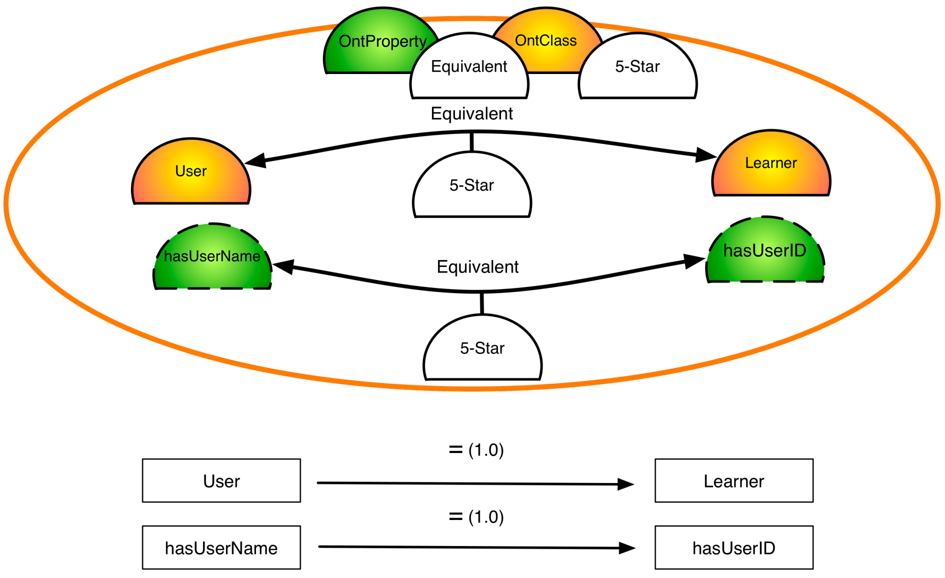


Figure 5.7: A part of the result of the alignment Import Process, which creates the Associations, and their metadata topics for each association.

import phase completes the initial import process. The SSV Map, which is stored in the persistent MySQL storage, can be accessed to perform semantic data transfer, or it can be further added to, for example to add more mapping information, more ontological entities or to be altered in some other fashion by a different reasoner.

Adding New Knowledge

One example of the advantages of this design is in adding new knowledge to the Target Application. This allows the Context Mediator to add new concepts to the Target Application which are found in the sources. The Source ontology has entities which carry information about the learning preferences of the user. This information could be useful to the target learning application, if the system is appropriately able to make use of it. In order to perform this addition, the $hasPreferences$ property is created by the UserModelTool reasoner in the Target Ontology. This expands the awareness of the system of context, by adding not only new information (such as the transcript of the student), but also new knowledge (the existence of user preferences, with the added information about their values). In this example, the UserModelTool creates a new property with the local name of the source property, and the base url from the target. In other ontologies, it might be appropriate to retain the URI of the property.

Once this operation has been completed, then the process of transferring data allows the system to treat the new property in the same way as those which had been in the ontology initially. The advantage of this is that the context has expanded the awareness of the system, and it can therefore take advantage of an improved representation of the user with more descriptive qualities as well as better data.

5.6.4 Data Transfer

With the Topic Map established, the ontological knowledge in the system remains in the local ontologies, as held by the SchemaManager. The inter-schema relationships are represented and characterised in the Topic Map SSV, and are ready to be used to query the sources of context. In this example, the UserModelTool is called through the ACP facade class to Enrich the Target Model. This is done in three phases: the first phase is the identification phase, which is used to locate appropriate information to transfer to the Target ontology. The second phase is the semantic phase, where the information is translated from the semantic entities in the source ontology into the semantics of the target ontology. Finally, the syntactic phase is responsible for translating the data and adding it to the Target Ontology.

Identification Phase

In the UserModelTool parameters, an identifying property is named. This is the property which allows the reasoner to choose which instance in the source is the equivalent one to the target's instance. This permits the system to find corresponding profiles. The relevant property in this case is 'hasUserName' in the Target Ontology, and upon querying the SSV, the 'hasUserID' property is the appropriate equivalent.

In the example, the Target Application's ontology includes an instance with the username 'jsmith'. This value is used to get the Instance URI of the 'jsmith' instance in the Source Ontology. This source instance is then queried, based on the class information for the 'Learner' Class, to get its property values. These values are recorded, and represent the identified additional information in the source ontology. For example, the source instance could contain the values 'programming.java.basic' and 'databases.tables.intro', which provide information on that user's learning history with programming and database tables, and can be copied to the target's equivalent instance.

With the identification phase complete, the set of information to be transferred has been identified and aggregated. It must now be translated into the appropriate properties in the Target ontology.

Semantic Phase

Each of the property value assertions taken from the source ontology is translated as follows: the equivalent property is located in the Topic Map through the association. A new aggregation is then created which includes the instance uri in the target (found by querying the ontology), the property name (found by querying the SSV) and the property value (which is converted using a lookup table built into the UserModelTool reasoner). This creates a new aggregated description of the semantically translated information.

The information now just needs to be added to the Target Ontology. In this simple example, there was no need to look at the metadata for the associations beyond the operation type. In more extended scenarios, it would be possible to choose different mappings based on the association metadata, for example choosing a higher-confidence mapping.

Syntactic Phase

The syntactic phase for this example is relatively simple. The aggregation of semantically translated values is added to the existing instance in the Target ontology, one at a time. The result is that the student's profile has been translated and transferred between the two systems. The enriched ontology can be written as a file, and loaded into the New University's Learning environment. If the Target Application is appropriately able, it can also make use of the added learning preference information to further personalise the student's individual learning experience.

5.6.5 Multiple Sources of Context

In this example, there is one source of context which provides context information to the target application. However, one of the key requirements of context is to be able to integrate a variety of sources of context. The ACP supports directly the import, uplift and alignment of the concepts from several different ontologies from more than one source.

To demonstrate the way the ACP extends to multiple parties, the example scenario is altered to include two additional sources of context, which also contain user profiles relevant to a set of learners. These profiles are from an additional source learning environment which contains other information about the learning history of the student.

In order to integrate the additional source of context, an external alignment description is used which describes the integration pathways between the Target Application and the new Source. The uplift process for adding an additional source of context is identical to the first, where the ontology concepts are imported by an uplift reasoner. As part of the uplift process, the concepts are associated with a service type topic; this type topic is the key to the uniform way in which ACP reasoners can access different service concepts during the integration process. Sources of context do not need to be added together, they can be added individually, as required, at different time intervals.

One approach to incorporating information from several sources would be to call the UserModelTool repeatedly, each time with a different parameter for the chosen service. The reasoner is able to transparently include information from each source. The correct source topics are identified by checking their topic's types, and ensuring they have the correct source service type.

The UserModelTool can be extended to take account of more than one service in a relatively straightforward way by extending its functionality in the identification, semantic and syntactic phases. In the identification phase, the tool can check each of the identified associations for each concept, and, depending on their type, and combine the contents. For example, if the second source service has a *hasLearningHistory* property associated with the *hasCompetency* property of the Target, the result of the identification phase can include instance and property mapping information identified in both sources. The semantic transformation phase can take account of the differences in types between the associations in each source as necessary to create a set of values to be included in the target at the syntactic phase. In the syntactic phase, conflicts can be resolved, for example by only including concepts found in both profiles, or by choosing one profile over another.

An advantage of the ACP architecture is that it places all the concepts from all participating services together at the same level. The relationships of the topics to services occurs through the type of the concept topics. This means that reasoners can access any service's concepts at each phase of the transfer. This has practical uses in resolving complex context relationships, for example in a situation where a set of different user IDs can be stored in an identity service, and used to identify instance mappings between the target and a source of learning data.

5.6.6 More Complex Examples

The approach described here presents a relatively straightforward example of how the ACP imports ontological information, queries knowledge bases and integrates contextual information. The system can accommodate more complex scenarios, and a few examples are presented below.

Complex Data Mappings

More complex data relationships can exist. Instead of a simple lookup table, the information to be transformed can be handled by complex reasoner functionality, even to the point of using an external web service call to make the exchange, though this has an accompanying performance cost. Depending on the complexity of the Topic Map, it is even possible to combine several pieces of information to integrate into one new piece of knowledge for the Target Application.

For example, in a more complex user model exchange, there might be a difference of level of abstraction between the source of context (which operates at a course level) and the target (which operates at a lower, subject competency level). In this case, the alignment is described in **Fig.5.8** During the mapping import phase, a new type called "CourseToSubject" is created. This type indicates that the source is in Course level descriptions, while the Target is at Subject level. This can be added to the mapping description after import into the Shared Semantic View. In addition the "PartialMapping" type is used to describe the relationship between the educational properties.

Assuming that there is a web service that can convert course codes to competencies, the transfer process is then altered as follows: the identification phase chooses values from the "PartialMapping" relationship, but the a reasoner is added to the semantic phase which can translate the course list to the subject list. This allows the subject

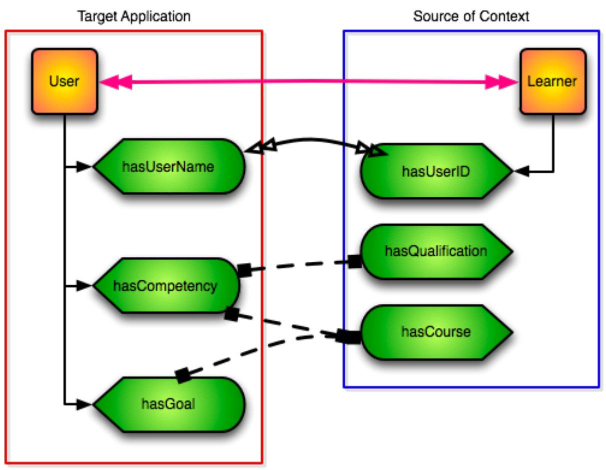


Figure 5.8: Diagram of a more complex example of data relationship resolution. Dashed, Boxed arrows represent partial mappings.

list from the previous courses in the Qualifications to be added to the Target's *hasCompetency* property. In addition, the current completed subjects from the *hasCourse* property can be translated to subject competencies. Finally, the goal of the current *hasCourse* can be translated to the Target's *hasGoal*.

This might leave significant additional information in the Target Application, if a user's full academic competency list were transferred. This can be mitigated in the reasoner design by limiting the competencies which are transferred. The ACP affords rich data relationship translation both with internal, reasoner derived functional mappings, or from external knowledge.

Several Mapping Descriptions

In the situation where there are several different alignment descriptions, the alignment reasoner can be reused, if the format is applicable. Otherwise, a different alignment import reasoner can be called, which can incorporate alignment descriptions in a different format. This second alignment import can happen at any point after the reasoner concepts have been imported. A key advantage of this is that the UserModelTool reasoner does not need to be aware of the differences between the import formats, they are represented identically in the Topic Map.

For example, two mapping description files might be created to align the ontologies in the example above. The first, Description A, maps *User* to *Learner*, and *hasUserName* to *hasUserID*, both with high confidence. However, Description A maps *hasCompetency* to *hasTranscript* with low confidence. Description B, on the other hand, maps only the properties of the two ontologies, and maps *hasUserName* to *hasUserID*, *hasCompetency* to *hasTranscript*.

When both of these sets of mappings are imported, the import reasoner assigns type information which describes which mappings are drawn from which description. The Transfer Reasoner can therefore choose to transfer property information only where both A and B agree.

The design presented in this chapter can resolve complex differences in representations between the structures of the ontologies. An example of this is the process of transferring a value which is a direct literal related to the class by a data property in the target, but which is related by an object property in the source.

The ACP reasoners can be structured to resolve this kind of semantic gap by mapping object property objects to instances and retrieving the property values. This example is demonstrated in detail in the Evaluation Chapter in the **User Model Transfer** case study.

These examples only represent a sample of the possibilities which this approach provides for. The key advantage to the approach is that the collaboration between different co-ordinated reasoners means that the system can build a rich semantic view of the knowledge and information held in both the target and sources of context.

5.7 Analysis of Implementation & Conclusions

This chapter has presented the implementation of the Design of a semantic-interoperation based context mediator, as defined in the design in the previous chapter. The key requirements from that chapter were as follows:

While the implementation does include the main requirements, one key issue is

Table 5.1: Summary of Requirements Drawn from Design.

#	**Description**	**Implemented in ACP?**
1.	Mediator Model of Interaction	Yes
2.	Ontology Reasoning	Yes[a]
3.	Creation of Knowledge Bases from Ontologies	Yes [a]
4.	Representation of Mappings and Concepts	Yes
5.	Expressing syntactic differences between concepts	Yes [b]
6.	Topic Map Manager	Yes [c]
7.	Expressing semantic differences between concepts	Yes [b]
8.	User-defined reasoning	Yes
9.	Run-time calling of reasoners	Yes
10.	Identification of contextual need	Yes [d]
11.	Resolving semantic differences between concepts	Yes [d]
12.	Resolving syntactic differences between concepts	Yes [d]
13.	Import Mapping Descriptions	Yes [d]
14.	Creation of suitable enriched ontology	Yes [d]

[a]– through Jena
[b]– using Topic Types and Association Reification
[c]– using TM4J
[d]– Using Reasoners

that the SchemaManager is not truly independent of the OWL language. The implementation is tightly bound to the Jena API, and exposes methods which represent the semantics of OWL (Classes, Properties, Individuals). There are two possible paths for resolving this: the first is to accept the notion that the abstract concepts of class, property and individual are present in many ontology languages, and use an abstract factory model to create different instances of the SchemaManager, The second is to remove the methods from the manager, and rely on the reasoners to ascertain the appropriate methods by reference to the ontology manager within the SchemaManager, which provides an abstract factory with indicators to the correct implementations. The great addition of complexity that each of these strategies creates means that both are beyond the scope of this implementation.

One of the key advantages of the system described above is its flexibility. The novel approach to using a programatically-generated, general semantic network (the Topic Map) to describe the Shared Semantic View means that the methods described above for importing and querying the participating ontologies are only one of a wide variety of possible methods. Individually, or in collaboration with each other, different reasoners can tailor the representation of the entities, and their categorisation and description in whatever way best suits their requirements. In addition, because the reasoners are loaded at run-time, it is possible to form new reasoner compositions and to delegate functionality. This reuse relieves some of the complexity of authoring reasoners and using the ACP to achieve context integration.

Chapter 6

Evaluation

The Evaluation Chapter of this thesis begins by discussing the methodological approach to evaluating the design and implementation presented in this thesis. This is followed by two case studies, and an analysis of the findings from those studies. A comparative analysis of the ACP with the State of the Art is then presented, followed by some conclusions.

6.1 Introduction

This chapter presents the evaluation of the context-informed semantic interoperation approach. The evaluation is based on the performance of the ACP implementation in two case studies, as well as a comparative evaluation of the approach with similar systems in the semantic interoperation and context domains.

6.2 Evaluation Methodology

The design of the ACP and of context-informed semantic interoperation is based on the objective of bringing contextual information from one or more sources of context and integrating this external knowledge into the information model of the target application.

The approach is based on the use of ontologies because, as seen in Chapter 2 of this thesis, they appear to describe the shared knowledge space more completely than other methods. Ontologies include both the instance information, and the schematic information that describes what concepts exist in the information space. This means that context integration can become a process of knowledge transfer as well as information transfer. Unlike conventional XML Schema approaches, for example, systems which use ontologies can recognise the conceptual space in a more general way.

By taking the view that context should be integrated without reference to a pre-existing ontology, the challenge of context integration can be viewed as one of semantic interoperation. Context integration can be viewed therefore as a specific case of ontology mediation, where several partially-overlapping ontologies each contain useful information which needs to be incorporated into a target ontology. An effective context mediator should be able to support the following:

- **Resolution of Differences in Schema and Data:** A context mediator should be able to deal with differences in the schemas of the participants' ontologies. The mediator should support the resolution of differences between data in the system. This has two benefits: the first is that it can directly support the interaction of ontologies which are schematically identical but which have data differences. The second is that it allows for the resolution of data level differences which are not conventionally expressible in the schema. This is evaluated qualitatively in the case studies. For example, the issue of how to represent concepts in the map so as to minimise conflicts is discussed in the first cases study[1].

- **Representation of the Schemas and Alignments:** The Context Mediator and its reasoners need to be able to represent and resolve a variety of different alignments, schemas and metadata. This aspect includes questions about the

[1]For example, where concepts are overlapping, but neither subsumes the other. Conventional first-order ontologies have trouble expressing this, but a system with a suitable data resolution feature set can express this relationship.

representation format of the SSV Topic Map, as well as some questions about the interaction between the ACP user-defined reasoners and the native ontology reasoners. One evaluation of this issue is found in the Queries trial of the first Case Study, which demonstrates a simple alignment and one requiring complex resolution of schema differences. The performance of both representations is measured over a small set of instances.

- **Inclusion of multiple participants**: Many of the semantic interoperation techniques reviewed focus on mapping two ontologies, one to the other. Contextual scenarios often depend on interactions between several ontologies in order to be able to adequately represent the variety of information available. A suitable context system should be able to handle pairing more than one ontology with the target. This is principally evaluated in the second case study, through a qualitative analysis of the representation of multiple parties in case study two.

These three measures of effectiveness are broad categories which gather together the requirements and features found in the design and the state of the art surveys undertaken in this thesis.

The key features identified by the design of the system are:

- the use of ontology mediation
- the use of model-based exchange
- the use of external tools to undertake mapping

The study of context revealed the need for a context mediator that could fulfil the following functions:

- The transfer of contextual information, and the
- The inclusion of multiple sources of context
- The use of user-defined reasoning to resolve abstraction differences

The transfer of contextual knowledge is achieved through an ontology mediator, and Chapter 3 defines seven properties of mediation. For the purposes of the case studies, these can be summarised as:

- Related to the articulation
- Related to the information / knowledge transfer

In essence, these properties relate to each other as functional requirements. The requirements form the design specify the need for a system which does not have an a-priori model of context, which can take advantage of mapping tools. The context requirements specify the specific need of a context mediator, and the ontology mediation properties describe the key properties that must be evaluated for effective mediation. This is represented graphically in **Fig 6.1**.

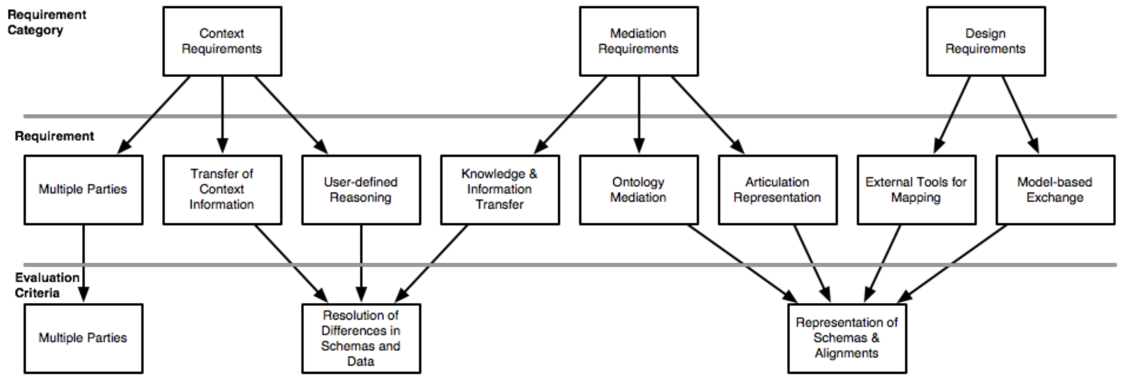


Figure 6.1: Diagram relating the findings of the State of the Art and Design to the Criteria for Evaluation.

6.3 Case Studies

This section outlines the case studies used to evaluate the context-informed approach. The objectives of this evaluation are derived from the overall thesis objectives. The evaluation process focuses on two case studies. Each case study comprises several trials which isolate and evaluate key aspects of the effectiveness of the ACP approach. Each of these key aspects is covered by different parts of the trials in the case studies. Some trials cover one aspect, while other aspects are divided between trials (see **Fig. 6.2**).

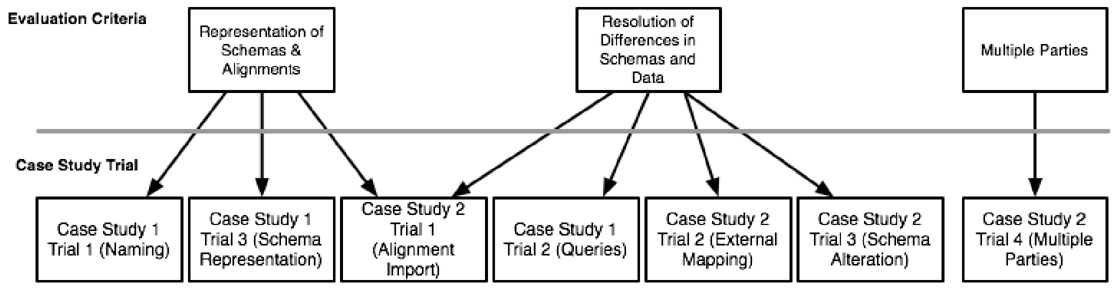

Evaluation Criteria
Representation of Schemas & Alignments
Resolution of Differences in Schemas and Data
Multiple Parties
Case Study Trial
Case Study 1 Trial 1 (Naming)
Case Study 1 Trial 3 (Schema Representation)
Case Study 2 Trial 1 (Alignment Import)
Case Study 1 Trial 2 (Queries)
Case Study 2 Trial 2 (External Mapping)
Case Study 2 Trial 3 (Schema Alteration)
Case Study 2 Trial 4 (Multiple Parties)

Figure 6.2: The Distribution of the Case Study Tasks over the key aspects of the Evaluation.

- The Representation of the ontology concepts is addressed in two case study trials. The Naming trial (1.1) investigates some of the possible conflicts and solutions to representing concepts from different, related ontologies in the same topic space. The Ontology Reasoning trial (1.3) assesses the value of trying to import more information from the ontology about the concepts, which is normally inferred by the ontology reasoner. The import of alignment descriptions (2.1) also introduces an examination of the representation of the class-property relationships.

- The resolution of Data and Schema abstraction differences is resolved over four trials. The Query Trial (1.2) demonstrates the resolution of a schema difference between the alignment of an object type property and a datatype property. This means that the user-defined reasoner uses the ontology data to resolve a schema difference. A variety of different structures are then examined in the External Matching Trial (2.2) in order to show the advantage of using the ACP as part of an ontology matching workflow. The creation of an alignment import reasoner was evaluated as part of the alignment import trial (2.1). Finally, a method for supporting different levels of schema inference capabilities is discussed in the Schema Alteration trial (2.3).

- Multiple Parties (2.4) are examined by importing a set of 'real-world' ontologies provided externally with their alignments. The expressive capability of the Topic Map is demonstrated by the ability of the ACP to represent more than one source of context.

6.3.1 Case Study: User Model Transfer

This section outlines the User Modelling Case Study. Many applications, particularly those which employ personalisation, employ user models to describe their users. These models contain information about the users such as their preferences, a record of their previous interactions with the system and other important information. This information is used by the system to change its behaviour and state in order to better suit the user.

One difficulty with User Modelling is that it is desirable to maintain a consistent user experience across a variety of systems, and there is significant heterogeneity amongst different applications. This Case Study considers applications which have ontologically-expressed user models, and attempts to transfer information between these models, bridging different semantic and syntactic gaps.

Objectives of this Case Study

The objective of this case study concerns the representation of the ontologies & alignment information in the Shared Semantic View Topic Map. There are a number of different ways in which this could be achieved. For example, there is a one-to-one mapping in this system between topics and ontological concepts. It is possible to consider a different scheme, where topics represent the shared concept across different ontologies. In this model, two related concepts are represented by two topics and an association between them. There is no single correct representation; one of the advantages of using a generic and flexible structure such as a Topic Map is that it can be used in many different ways to express the concepts. This case study attempts to define some features of a representation scheme which are suitable for the context integration process.

From an examination of ontologies, there are a few key features which can be included in the Topic Map, which this case study will investigate. The choice between these different features will help to create a representation which remains expressive, but which does not rapidly become too complex to manage effectively. The specific objectives are:

- To examine the representation of Ontology Classes in the Topic Map, their naming and how to retain a reference to their originating definition in the ontology.
- To examine the representation of Ontology Properties in the Topic Map, their naming and how to retain a reference to their originating definition in the ontology.
- To evaluate the representation of the Is-A hierarchy in the Topic Map – is it appropriate to represent the super-class and sub-class relationships in the Topic Map?
- To evaluate the representation of the Has-A relationships in the Topic Map – should classes and properties be associated in the Topic Map?

These considerations help to decide the line between what information is stored in the Topic Map, and what information is retained in the schema manager, and therefore queried using inference from the ontology reasoner.

Methodology & Background of the Case Study

The case study consists of ontologies from two possible eLearning systems. These systems employ user models to describe the educational history and experience of a user, including specific subjects which the user has studied. In this case study, the two models being investigated are based on a model related to the APeLS eLearning system [Conlan and Wade, 2004] and the IMS LIP Learner Information Package Specification [IMS Global Learning Consortium, 2005]. These models were chosen because the APeLS model represents an example of a model in practical use in eLearning systems, while the LIP model is a well-known international standard model for learner profiles.

Neither of the user models chosen have official ontological representations. This is an advantage in this evaluation scenario, as it allowed for the evaluation to include the examination of different variations in the generation of the ontologies that described the knowledge in the models. While this does mean that the ontologies were generated specifically for experimentation with the ACP, this was appropriate given the objectives of the case study.

Input & Metrics

Two versions of each of the two schemas were examined in the course of this case study. In each case, the ontologies were derived from the original schemas and modelled with different features to help assess the considerations associated with a suitable representation. The ontologies were as follows:

1. One Class, Many Properties – these ontologies represent the learner model in the form of a class that represents the learner, with attributes represented as properties of the class instances. This model is schematically simple in the class domain, but is useful for investigating the nature of property queries. The properties-oriented LIP ontology is pictured in **Fig. 6.3**, and the AE ontology structured in this manner is **Fig. 6.4**.

2. Several Disjoint Classes, with Properties split by category – these ontologies represents the concept space of the models by translating the attributes of the model into ontological classes. The values of the attributes are represented by specific properties, which are bound to those classes. This design creates a more complex map of classes which need to be accessed and queried by the reasoners. The class-oriented LIP ontology is pictured in **Fig. 6.5**, and the AE ontology structured in this manner is **Fig. 6.6**.

The evaluation of this case study consists of a set of comparative tests, which examine different approaches to what is represented in the Topic Map, with the intention of answering the objectives of the case study. The tests are as follows:

1. Entity Naming – What topic Labels should the topics representing particular entities have? This test compared full URIs with Local Names

2. Querying Type Hierarchies – is it feasible to attempt to query super/sub-class and super/sub-property relationships from within the map? This was evaluated based on the complexity of the queries to the map required by different representations.

3. Querying Property Values – is it useful to query the Topic Map for the properties of a Class, or is it better to retain those relationships within the schema manager?

These issues are evaluated in terms of the metrics described below. These metrics were chosen to measure the properties of the representation which are suitable for ontology mediation for context.

1. Minimising Redundancy – the management of mediation is at best a partially automatic process. This means that it is desirable to make the maps as simple as possible, so that they are understandable to the administrator who manages them. Additionally, less redundant maps could be more useful and re-usable. Redundancy can be assessed qualitatively by comparing the representations for unnecessarily duplicated information.

2. Maximising expressiveness – one of the key requirements is the ability to address the transformation of information at both a semantic and a syntactic level in the system. The Topic map scheme needs to be able to retain flexibility in expressing data and semantic differences. The expressiveness of the system can be assessed by taking an example which shows a complex difference in schema representations, and measuring the performance cost as compared to a simpler transfer, it can also be examined qualitatively by demonstrating that the system can represent all the necessary features of the concepts and their relationships.

3. Controlling complexity – The formation of queries should be made relatively straightforward and as repeatable as possible, to make the design of re-usable reasoners possible. This can be assessed by measuring the growth of the map as more concepts are included.

4. Performance Degradation – Performance should ideally not degrade too rapidly, particularly on the transfer of data. The degradation as model complexity increases is less important than the degradation as the amount of data passed increases. This is due to the desire to map relatively small ontologies, rather than large, complex schemas.

The evaluation was undertaken in several stages, where different ontologies were loaded into instances of the ACP, and queries were made with the intention of transferring data from the source ontology to a matched instance in the target ontology. The ontology structures are outlined below.

An initial version of the ontologies was generated which has a single class that represents each learner, and the attributes of the learner represented as datatype property values. Note that in the case of the LIP ontology, some simplifications were undertaken, such as removing the detailed components for addresses and parts of names. Details of the eliminated elements can be found in the Appendix to this thesis.

The ontology structures used in this case study were designed to be as small as possible. There were several reasons to design small schemas: the first is that it lowers the complexity of the ontology loading and inference phase in the schema manager, which was not under direct evaluation in this work. The second is that it allowed the assessment of performance to evaluate the particular cost as the complexity of the SSV operations was examined.

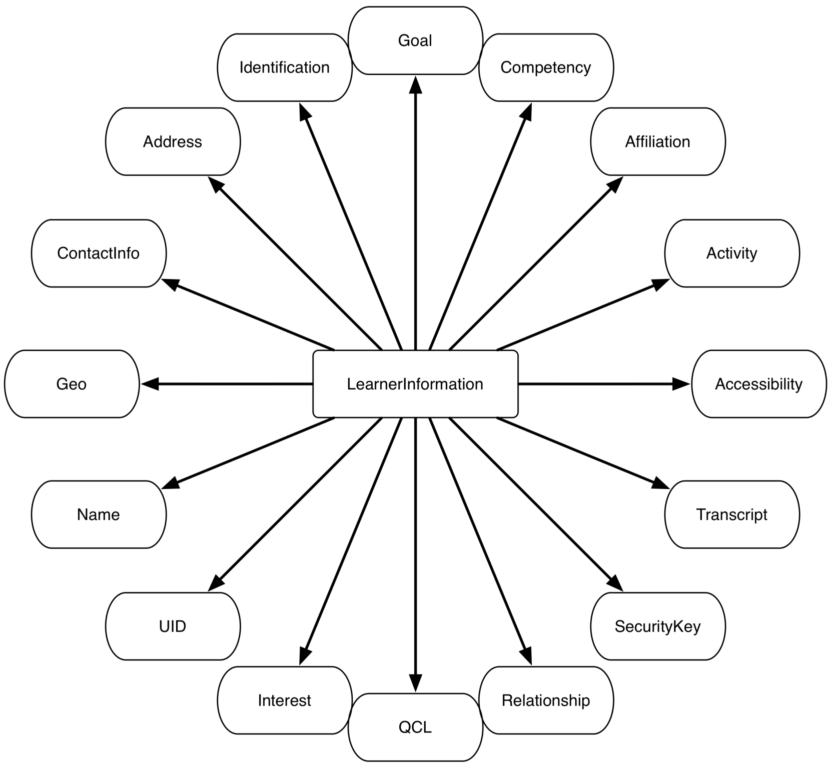


Figure 6.3: The Property-based version of the LIP ontology. Solid boxes indicate ontology classes, dashed boxes indicate Datatype properties.

Evaluation Tasks

In order to evaluate the questions posed for this case study, each of the ontologies was imported into an ACP Topic Map, and queries were formed. A set of test mappings were then created, which were used to evaluate the system when using multiple mapped ontologies. Different combinations of ontologies were evaluated as follows:

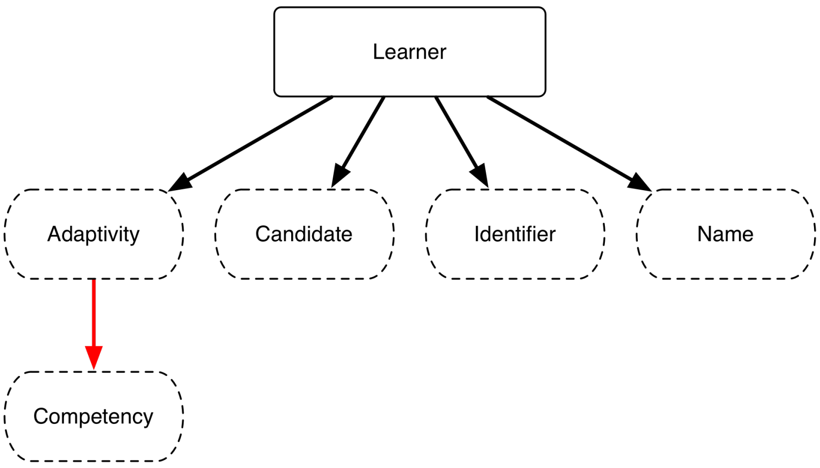


Figure 6.4: The Property-based version of the AE ontology. Solid boxes indicate ontology classes, dashed boxes indicate Datatype properties.

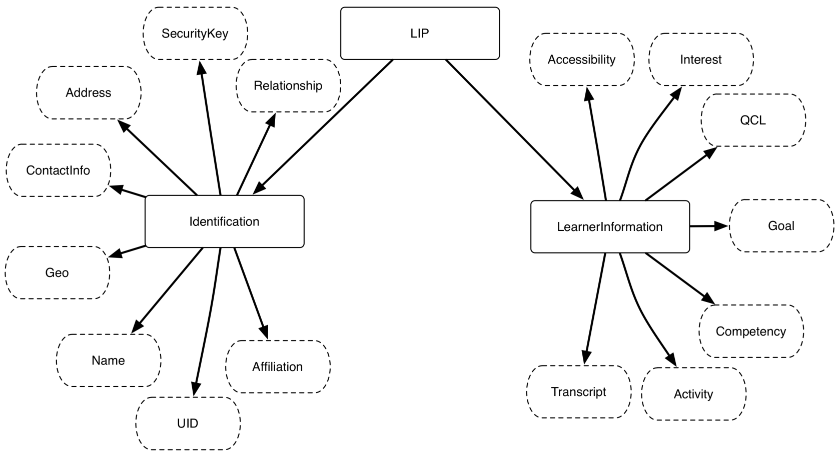


Figure 6.5: The Class-based version of the LIP ontology. Solid boxes indicate ontology classes, dashed boxes indicate Datatype properties.

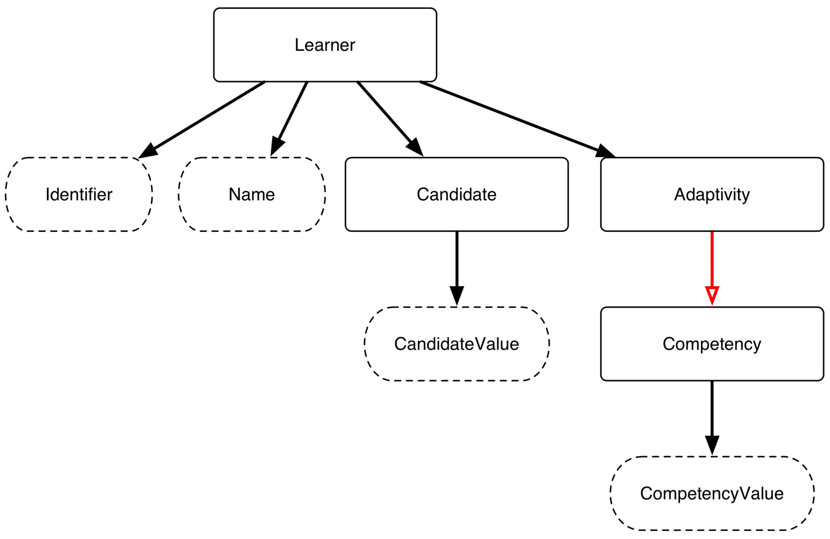


Figure 6.6: The Class-based version of the AE ontology. Solid boxes indicate ontology classes, dashed boxes indicate Datatype properties.

The First Trial was to attempt to import the ontologies and compare different naming schemes for them. The local name for the entity was used to generate a label for the topic, which was generated by the schema manager. The four ontologies were imported in different combinations into the Topic Map, using two different naming schemes. Analysis of the ontology concepts along with previous literature reviewed yielded a set of possible naming conflicts. The objective of the first task was to characterise how the two different naming schemes handled the different naming clashes, and to gain an overall understanding of how ambiguously-named concepts behave in the ACP.

The Second Trial involved comparing the system with different ontological representations. The idea was to show that the system could transfer information across different types of layers of abstraction. This showed the ability of the system to deal with differences in ontological representation, and to be able to transfer information to the Target from ontologies with different structures. A secondary interest was to assess the performance of the system with different numbers of instances and alignment relationships.

The Third Trial involved an examination of the incorporation of inheritance information into the Topic Maps. Different query approaches were then evaluated to determine whether the inclusion of type hierarchy information was useful. The two strategies in this case were either to use the Topic Map associations or the schema reasoner to determine type hierarchy information.

In order to test the is-a relationship, the classes were imported and in one version the Topic Map expressed the has-a relationship, while in the other the ontology schema was queried for the relevant information. The difference between the two maps is that in one case, an association was created based on which properties were associated with the class, based on the domain. In the second case, the reasoner was used to determine which properties were in a particular class.

Results for Trial 1: Naming

The two naming schemes which were evaluated in this trial were, in the first case, the use of the concept name (as defined by the schema's local name), and the full concept URI. The first case study trial found that there were four main ways in which two concepts could be related by their names:

- **Different Names, Same Concept**: is the situation where the names differ, and the concepts are the same. The conflict in this case arises where the naming scheme is too specific for the query, and the fact that the concepts are members of different ontologies leads them to be separated. For example, where the two ontologies imported are the two LIP-based ontologies, with the local naming scheme, it is possible to retrieve the concepts related to the *transcript* concept in both ontologies merely by looking for that name. This allows related concepts to be queried for.

  A concrete version of this situation arises where a reasoner is designed to retrieve all of the competency information from several services, all of which are using LIP-based ontologies. In these cases, the *transcript* concept is the same in each of the ontologies, even if the representation is different. The property-based LIP ontology *transcript* concept has the full URI of `http://kdeg.cs.tcd.ie/ontologies/LIProp#transcript`, while the class-based LIP ontology concept has the full URI `http://kdeg.cs.tcd.ie/ontologies/LIPClass#transcript`. Different queries would be needed to find each of the different full URIs, even though the concepts are the same. Since the short name is the same for both ontologies, one query finds both concepts.

- **Same Name, Same Concept, Different Data**: is where the the names are the same, and the concepts are semantically equivalent, but contain different data. The typical example for this might be in where the two participants in the system use the same ontology, for example, the AE Properties-based ontology. Each of the services in this system has a different set of user accounts, identified by the *identifier* property. For example, the same user named 'Jane Smith' might have the username 'jsmith' in one system, and '4144025' in the other. While conceptually they are similar, the content of the models differs at a data level.

  Neither naming scheme alone is able to resolve this conflict, and it is necessary to query the association between the topics, if any, to resolve the relationship.

  In this case, the difficulty appears when two equivalent participants representing the same real-world concept in different instances of the same ontology. For example, both participants use the Properties-based Adaptive Engine Ontology. The full URI for the *identifier* concept is `http://kdeg.cs.tcd.ie/ontologies/AEProp#identifier`. However, despite using identical software, the two participants have different user lists, and different user names for particular individuals. It is necessary to characterise the relationship between the two at a data level to resolve this problem.

- **Same Name, Same Concept, Same Data**: The third situation is where the names are the same, the concepts are the same, and the data is the same. This situation is the inverse of the first one. In the case of either naming scheme, it is possible to use the name system to query for related data across all the ontological concepts which have the appropriate name.

- **Same Name, Different Concept**: This fourth situation is perhaps the most important. In this case, the names are the same but the concepts are conceptually different. In this case, a query for a particular name will retrieve concepts which are fundamentally unrelated. The case arises where a word with multiple meanings is used in different senses in the different ontologies. A somewhat weak example exists in these ontologies, where the *competency* concept in the AE profile can include some features of the *competency* concept in the LIP profile, but it is equally possible that they would not overlap. This arises because the LIP profile has several fields which could represent the user's knowledge, while the AE profile aggregates only one. For example, if the AE *competency* field includes values such as 'sql.table.select', and a value such as 'java.intermediate', which is a broader qualification that would not be considered a competency in LIP, but rather a transcript value. The use of the URI scheme is more helpful in this case, as the assumption that word-senses are universal across ontologies is diminished by using the full URI.

  There is a semantic difference between the *competency* concept expressed in the Adaptive Engine Ontologies and that of the LIP ontologies. A query which retrieves concepts named *competency* will get two concepts with the same name, but different semantics. This could lead to potential errors as the Adaptive engine concept of competency is broader than that of the LIP profile. This risk is somewhat lessened where the two full URIs are used, since they differ substantially.

The most important conclusion of the examination of names is the need for additional information to characterise the specific relationship between similarly or differently named concepts; names alone do not suffice. However, some decisions can be made about the types of conflict that can occur, and how to avoid difficulties. It seems from the trial that the use of full URIs makes a stronger statement about what concept the Topic refers to, and therefore reduces the likelihood of a semantic ambiguity resulting from a name-based query. One additional advantage of the URI-based scheme is that this is generally the addressing scheme used to access concepts within the schema reasoners, and therefore the names can be used without having to perform additional queries.

In order to be able to appropriately query for data, it is necessary to establish the exact conceptual relationship between the two topics. This must be done using both the topic metadata(such as the type of the topic describing its ontological type[2], and the association information which characterises the semantic and data relationship.

The performance[3] of the system for importing concepts was measured. The four ontologies were each imported in turn into the same Topic Map. The trial was run three times, with different ordering for the ontologies, in order to gain a representative average. The performance was measured on the version of the APC implementation which included the indexed naming system.

Trial	1st Service	2nd Service	3rd Service	4th Service	Total
1	5.892	3.682	4.038	3.642	19.368
2	6.588	4.587	2.933	2.875	18.553
3	6.475	4.006	3.672	3.099	18.933

Table 6.1: Import times for four ontologies in different orders. All times are in seconds. The import of the first service takes longer because it includes the time to create the map.

The use of small ontologies means that these statistics represent an estimate of the minimum time which would be needed to import simple ontologies into the system, having loaded the schema and SSV infrastructure. From the table, it is possible to estimate the time taken to initialise the map and load the inference libraries at approximately two seconds.

[2]OWL URIs do not distinguish between different property or class concepts.

[3]Performance is measured as the time output by using the Java Calendar class to record the epoch time in milliseconds at each stage of the trial. The trials were conducted on a MacBook Pro with Intel Core Duo 2.6GHz, running a virtualised image of Ubuntu 9.04 64-bit with 1984MB of RAM allocated.

Results for Trial 2: Queries

The objective of this trial was to gain an estimate of the performance and effectiveness of the system under different model transfer situations. The trial took place with two ontology combinations. In both cases, the AE ontology was the Target Application, and the LIP ontology was the source of context. In the first ontology combination, the property-oriented AE ontology was enriched with knowledge from the property-oriented LIP ontology. In the second combination, the property-oriented LIP ontology was enriched with knowledge from the class-oriented LIP ontology. Instances were generated for the Source ontologies which included information about the user in the form of concept strings, which approximate the LIP format. An ontology mapping was created. The trial compared the time taken to transfer 1, 5 and 10 instances for 1, 2 and 3 associations requiring data transfer. Each Instance contained the same learning information, so as to provide consistent performance estimates.

In this system, the two ontologies were imported using a SchemaIntegrator Reasoner. This reasoner operates by listing the classes and properties of each service's ontology. Ontology Classes are given the types for OntClass, the service URI. Ontology Properties are given the type OntProperty and the serviceURI containing them. The full URI of the class or Property is attached to the ontology is used both for the BaseName for the Topic, and for the content of the Occurrence, which also has the OWL type. Alignments for these ontologies were manually generated. The alignment for the first property-oriented trial is relatively simple and involves associating the two classes and the equivalent datatype properties. The IDProp type was added to the topics which represent the unique username field in both services. The alignment for the class-oriented trial is more complex, and is shown graphically below (**Fig. 6.7**).

Red lines indicate associations created by the alignment, and callouts represent additional types added as part of the alignment. This represents one of the advantages of using a Topic Map structure for representing the mappings: the full map has been built with different reasoners as needed, with the ability to add metadata for specific purposes as required.

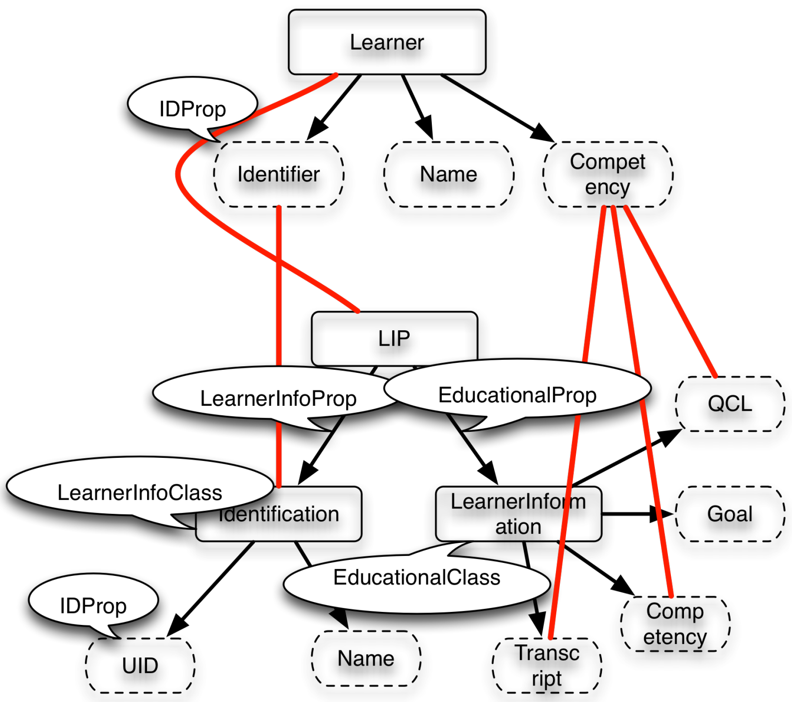


Figure 6.7: Graphical Representation of the Alignment for this Task between the Property-based AE and the Class-based LIP. Solid boxes indicate ontology classes, dashed boxes indicate Datatype properties. Lines indicate alignment mappings, arrows indicate class-property relationships and callout bubbles indicate attached types in the Topic Map.

The first phase in the context information integration is the identification phase. The purpose of the identification phase is to locate which parts of the source ontologies need to be queried to transfer appropriate context information. This includes identifying both the correct properties and classes through the topic associations, and the specific instance data. In both trials, an assumption is made that there exists a property in both service ontologies which uniquely identifies a particular users record (tagged with IDProp). This property is assumed to be unique and, for this example, the same in both systems. It would be a simple extension to the functionality of the reasoner to translate between different usernames. The assumption is also made that the list of blank instances in the target ontology will be filled by the source ontologys instances, which contain educational information. In the Property-oriented Trial, the objective is to identify the equivalent instance in the source ontology which correlates with the instance in the target ontology. This is achieved by looking for the property in the Target Ontology which has the IDProp type and finding its associations. The Source ontology is then queried for the instances with the same ID property value as each of the Targets Ontologies. The educational information is then identified by finding the appropriate equivalence associations in the Topic Map, and recording them for each source service. In the case of the Class-oriented trial, the matter of identifying the correct instance is more complex. The equivalence relationship established between the classes is between the two profile classes. However, the users information in this case is split between two separate instances of different classes. It is necessary to use some extra type information to look up the appropriate instance. This works by first identifying the correct Learner Information Class instance for a particular username. This information is then used to find the appropriate LIP class instance. From this, the system can use the designated educational Property to find the educational information instance, which can be used to transfer the educational data. The educational information property identification is similar to the property-oriented trial.

For example, in the Adaptive Engine Property-based ontology, a user profile for 'jsmith' is contained in the instance with the URI `http://kdeg.cs.tcd.ie/ontologies/AEProp#jsmith`. However, in the LIP Class-oriented ontology, the username information is held in the instance of the *Identification* class with URI `http://kdeg.cs.tcd.ie/ontologies/LIPClass#idjsmith`, and the learner information for that profile is held in an instance of the *LearnerInformation* class with URI `http://kdeg.cs.tcd.ie/ontologies/LIPClass#lijsmith`. The relationship between these two instances is held in *LIP* class instance with URI `http://kdeg.cs.tcd.ie/ontologies/LIPClass#LIPjsmith`. In order to find the learning information held in the *QCL*, *transcript* and *competency* properties of the LIP Profile, the system:

1. identifies the correct *Identification* instance using the fact that the *UID* property has the 'idProp' type in the map.

2. finds the *LIP* instance which points to that *identification* instance using the *hasIdentification* property, which is typed as 'LearnerInfoProp'.

3. finds the *LearnerInformation* property value for that *LIP* instance.

4. identifies the equivalent property associations in the source and maps them to the target (mapping *competency* in the Adaptive Engine ontology to *transcript*, *QCL* and *competency*).

This creates both an instance map and a property map, which can then be used by the semantic phase to retrieve the information.

The second, semantic phase begins with two sets of information: the mapping between the instances of the target and the sources, and the mapping between the target properties and source properties. The system can then iterate through this list and query the source ontologys instances for each instance and gather the information. At this point, the information can also be semantically transformed as necessary. The queries are issued to the schema manager, and so the results come from the inferred ontology knowledgebase.

Continuing the example, this phase generates a list of tuples which include the instance, property and value information for each property value retrieved from the source. The information can be transformed as necessary at this point.

In the final syntactic phase, the list of values for the targets instance properties are passed to the targets ontology. Any other syntactic restrictions that might emerge can be inserted here, such as how particular values are asserted, this includes the choice of property types or the way values are assigned. In regards to the property information, a reasoner is required to be able to resolve the has-a relationships within an OWL ontology. This arises because the instance information can define which classes have what properties. It is therefore similarly difficult to devise a general means for expressing the has-a relationship in the Topic Map.

Finally, the updated ontology is now resident in the schema manager, and can be returned in serialised form to the Target Application, or otherwise used by the context system.

One early difficulty in the performance of the system was that the process of finding the unique identifier from the topic label. There is a many-to-many mapping between names and topics, and the initial algorithm looped through all of the names of all of the topics and sought matches. The degradation of performance in this process was substantial. The implementation of the system was therefore altered to take advantage of the built-in basename index system. This feature was part of the TM4J Topic Map library, and provided a substantial saving in time required to locate topics of a particular name.

Tables 6.2 & 6.3 indicate the average times for the complete process of initialising the system, loading and importing ontologies, generating alignments, calculating instance mappings and transferring instances.

# of Instances	1 attribute	2 attributes	3 attributes
1	14	14.43	14.15
5	13.91	14.12	15.02
10	14.09	14.63	14.69

Table 6.2: Average Transfer Times (in sec.) where the source of context's identifying property was an objecttype property

# of Instances	1 attribute	2 attributes	3 attributes
1	13.52	13.66	13.66
5	13.45	13.76	13.90
10	13.49	14.78	14.08

Table 6.3: Average Transfer Times (in sec.) where the source of context's identifying property was datatype property

There is no significant trend across the different configurations. This would seem to indicate two things: first, for the small scale ontologies used in this example, the mediator's performance does not degrade rapidly either for alignment or schema complexity, or for number of instances. Secondly, the cost of loading the infrastructural libraries of the Topic Map and ontologies seems to dominate the performance cost of the process.

To give a sense of scale of these transfers, they are small by comparison to those more commonly used in ontology mediation (see the case study below). However, they are representative of the size of information which might be transferred in context situations. Context information is usually transferred for particular situations, such as a small group of learners or a particular user. The amount of information to transfer is also likely to be small, for similar reasons.

Results for Trial 3: Schema Representation

In order to be able to attempt to resolve the issue of querying super- and sub-type information within the Topic Map, it was necessary to attempt to establish an algorithm which would satisfactorily resolve type hierarchy questions. However, in the absence of the creation of a reasoner for the Topic Map, it does not appear to be possible to create an appropriate algorithm. There are two difficulties: the first is that the current associations are non-directed, and this means that it is difficult to resolve the hierarchy. The second is that it would be necessary to know in detail the behaviour of the concepts within the map. While ontologies crystallise conceptual spaces, the specific behaviour of their entities depends on the behaviour of the reasoners and the languages which they are written in. It does not appear to be possible to generalise this, particularly within the loose descriptions which Topic Maps provide.

With the strategy of devising a reasoner for the Topic Map discarded, the remaining strategy is to query the reasoners themselves and record the information in the map. However, this is undesirable for two reasons: first because it is a duplication of information which exists within the schema knowledge bases, and second because the information could be subject to change with the arrival of new instances, or it could differ between different parties to the same shared semantic view.

For these reasons it was decided that it is preferable to be able to resolve schema and instance queries within the specific managed knowledge bases within the schema manager. The Topic Map's role in the system is to represent the concepts in the system, their alignments and the metadata needed by the user-defined reasoners. The internal ontological relationships of the concepts are left to the native ontological reasoners.

Overall Findings for this Case Study

The analysis of the naming trial indicates that the use of Full URIs for topic naming is preferable. There is a suitable assumption that the use of the same URI will indicate that there is a *schematic* similarity between two entities from different ontologies. This assumption does not mean that the same data is held in each ontology, but they can be said to refer to the same notional concept. The use of full URIs also help with more general problems, such as conflicting local names, where totally unrelated concepts have the same local name, each a different sense of the same set of letters. The first trial also supports the need for metadata to describe the service membership of particular concepts, as well as querying associations as the method for unambiguously determining the relationship between two topics. Neither naming scheme provides a sufficient description of the shared semantic space.

The second conclusion is that it is desirable to retain as much information as possible in the ontologies themselves. The lack of a Topic Map equivalent to the ontological reasoner meant that it was difficult to be certain that the formation of queries was correct, unless the reasoners were highly specialised. Attempting to represent this information in the Topic Map was redundant, and represented a less efficient method than forming a more appropriate query to the schema manager.

Similarly, the association between properties and classes could be elicited from the ontology schema, but doing so applied conceptual constraints which do not exist in the knowledge base itself. The association between properties and classes in

OWL depends on the instance information available, and while domains and ranges provide some assistance, there are other ways of specifying local type restrictions [Bechhofer et al., 2004] which are not currently queried by the system. For this reason, it is also preferable to retain the has-a relationship information within the ontology, and query it at run-time. This has the added advantage of allowing for change in the event that new instance information appears.

In practice, the only effective way to query for both has-a and is-a relationships in the ontology is using a reasoner, so the choice of representing the associations in the map effectively ends up depending on the reasoner either way. This fact reinforces the desirability of using the schema's native reasoner to determine ontological reasoning when needed, especially for the association between classes and properties, because the use of domains and ranges is entirely optional in specifying properties. This has the consequence that without using a reasoner, a large proportion of important class-property relationships could be missed.

With regard to the is-a relationship, creating associations which describe the type hierarchy in the system greatly increases the complexity of the queries for the map. Currently, the ACP does not make use of role types to define how each member type in the association participates in the system. Including this would allow the system more expressiveness by allowing for identification of which class is the superclass and which is the subclass in the association. However, this would come at the cost of needing to provide types to describe the member roles, and require the system to query those member roles extensively to resolve the type issue. Furthermore, this fails to address the underlying difficulty of being able to say for certain whether a query has completely or appropriately resolved the type hierarchy. Attempting to represent the is-a relationship in the map creates a large number of extra queries which are not even guaranteed to resolve correctly. Based on this,it was found to be preferable to be able to resolve questions of type within the native ontological reasoners, and this can be achieved using the ACP architecture. An added advantage is that it permits different ontologies to be treated with different reasoners, in the case that this is desirable (for example, for different OWL dialects).

The use of the system, and the various attempts to handle topics with similar names led to two interesting conclusions. The first is that the use of topic types must be carefully guarded against unwanted ambiguity. The second is that there was a need for a richer way to describe the relationship between topics, for example to characterise more detailed information about the semantic or syntactic relationship. This led to the development of the reification features which are described in the Design chapter.

The second trial in this case study has also demonstrated one of the key requirements for the ACP's ability to resolve differences of abstraction. The transfer of the identification information from the *hasIdentification* object property in the source is an example of the resolution of an object-to-data property abstraction. The object property contained an instance of the *identification* class, which needed to be aligned with the datatype identifier property in the target. This trial demonstrated that the ACP can transfer information across a complex difference in schema representation, using instance data to determine the instance alignment and retrieve the correct data, and that for small ontologies with few links, there was no significant penalty for resolving the more complex relationship.

The size of the Topic Map for a particular context situation depends on three factors: the size of the ontologies, the number of types, and the number of mappings loaded. On the other hand, the cost of inferring ontologies only depends partially on the number of entities; the complexity of the logic underlying those entities can create very slow or even intractable inference. It is therefore likely that, for most cases, the limiting factor of complexity is the ontological complexity, rather than the Topic Map.

This points to the desirability of better performance in the loading of ontology and Topic Map infrastructure. Part of this cost can be deferred in practical use of the system, because the ACP saves persistent Topic Maps in a database, which means that the ontologies do not need to be imported each time a query is issued. However, overall the current performance would not be suitable for high-responsiveness applications. The high cost of using a reasoner such as Jena, and strategies for selecting other reasoners are beyond the scope of this work, but the issue has been similarly highlighted by others [Lewis et al., 2006].

6.3.2 Case Study: Bibliography Benchmark Ontologies

Objectives of this Case Study

This case study is designed to evaluate the use of the ACP with high-quality, independent ontologies and independently-produced mappings. This shows that the ACP can form part of a mediation workflow that begins with an *a-priori* mapping event, and ends with context information integration using mediation. This is an important advantage of using semantic interoperation because it takes advantage of existing tools for mapping ontologies and contributes to mapping reuse by taking advantage of generic mapping descriptions.

The first objective of the case study is to show that the ACP can fulfil the requirements of a semantic mediator, the ACP must demonstrate the ability to import mapping information from external tools, resolve a variety of schematic differences, and show that it can handle a set of sources of context. Two trials in this case study demonstrate those features on a set of independently-authored ontologies and mappings, one deals with the method of importing mappings and ontologies, and the second with the process of transferring information using the same reasoner in different ontology combinations.

The second objective of the case study is to evaluate, from the perspective of the context mediator, two of the key features of a context-informed system. The first of these features is to examine supporting enriching the target application's schema, in order to allow a highly-dynamic application to take account of new concepts. The second trial demonstrates the process of importing more than one source of context ontology.

Methodology

In order to help show that the ACP is reasonably general in its ontological support, it was decided that a set of independently-authored ontologies and alignment descriptions should be used. The OAEI[4] Workshop provides one of the few high-quality reference ontology sets which include alignment information and which can be used to evaluate a system such as the ACP. The OAEI itself is directed more towards the process of finding and describing ontology matches themselves, so the ACP is not a direct candidate system. The idea of being able to consume the output of a matching process for mediation is, however, a powerful advantage of the semantic interoperation approach.

The subject domain for the OAEI reference ontologies is that of bibliography, and publications. The ontology instances describe various publications, authors and publication venues. The key importance of these ontologies is not their specific subject domain, but rather the fact that they provide an independent test bed for the ACP.

The OAEI reference ontologies are arranged into a series of alignment tests, where two ontologies are given a particular alignment, and differ from each other in various structural ways. The ontology pairs consist of a reference ontology, which is the target application for the ACP, and a source ontology, which is the source of context. The context data in this case is the instance information about the publications. Due to the nature of the OAEI testing, there are several ontology pairs which are not directly relevant to the ACP evaluation. These include tests such as the one where there is no alignment between the ontologies, and various tests where only additional annotations have been removed. The evaluation case study is therefore based on a partial set of the reference pairs, selected to provide coverage of the required features of the ACP. The alignment information for the OAEI ontologies is given in the form of Align RDF description files [Euzenat and Schvaiko, 2007]. These files are sets of alignment pair descriptions which include the source and target ontologies, as well as the full concept URIs, a measure of the confidence of the alignment and the type

[4]Ontology Alignment Evaluation Initiative

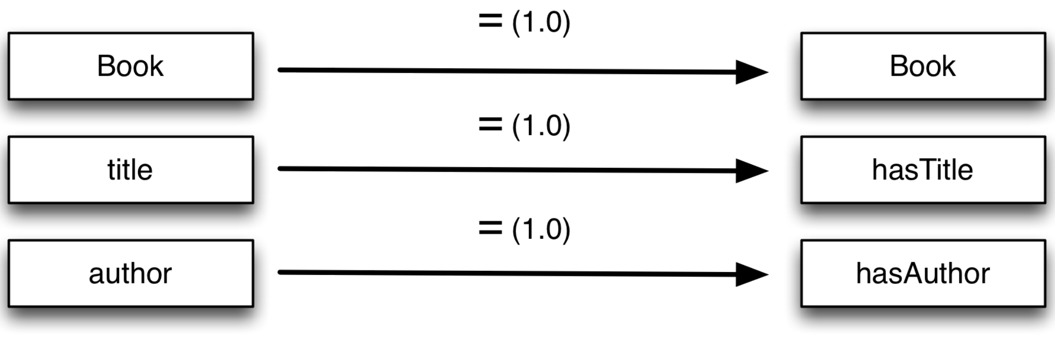


Figure 6.8: Graphical representation of an example mapping. The full URIs have been shortened to class names for clarity.

of relationship. The alignment files are the result of a pooling exercise undertaken by the participants in the workshop [Euzenat et al., 2007], and can be viewed as an agreed 'gold standard' mapping. The mapping RDF files are in Align Level 0 format[5]. An example alignment between the *title* properties in the reference ontology and the MIT Bibtex ontology[6] takes the following form:

```
Refalign.rdf
<map>
<Cell>
<entity1 rdf:resource="http://oaei.ontologymatching.org/2007/benchmarks/101/onto.rdf#title"/>
<entity2 rdf:resource="http://oaei.ontologymatching.org/2007/benchmarks/301/onto.rdf#hasTitle"/>
<measure rdf:datatype="http://www.w3.org/2001/XMLSchema#float">1.0</measure>
<relation>=</relation>
</Cell>
</map>
```

Each alignment file describes a one-to-one mapping between concepts in the two ontologies, referenced by their URIs. This means that there is no indication from the alignment file as to whether it is mapping a Class to a Property, for example. In fact, the alignments in these trials mapped only classes to classes and properties to properties. A partial graphical representation of the *Book* Class and two properties is shown in **Fig. 6.8**.

[5] *c.f.* Chapter 3 for more detail on the Align Format.
[6] one of the 'real world' reference ontologies.

Inputs & Metrics of the Case Study

The Evaluation of the case study will therefore depend on four tasks:

1. To develop ACP Reasoners which can import OWL ontologies and Align RDF mapping descriptions to form a Shared Semantic View. The evaluation of this task included two key metrics: to show that the system could import all of the information from the alignment descriptions in a way that could be extended by other reasoners. Secondly, a question arose about importing class:property pairs, rather than classes themselves. The key metric in this case was the cost in terms of size for the map. This task helps to assess the expressiveness and redundancy of the system.

2. to show that the ACP can effect a variety of different types of context integration, including bridging different schema representations. The key metric here was to be able to show that the system could use the same transfer reasoner to transfer information between different ontology pairs, each with a different alignment description file. This shows the advantage of being able to use the Topic Map to represent the alignment information in the Topic Map, creating more reusable reasoners. The key variations to be covered are:

    (a) **Concept naming variations**: including translated Class and Property names, labels and comments, changes in spellings, naming conventions and other variations.

    (b) **Owl language variations**: including the addition and removal of restrictions, properties and data types.

    (c) **Hierarchical variations**: including hierarchy expansions, hierarchy flattening (to the point of removing all super-class relationships) and inclusion of intermediate classes.

    These tasks demonstrate the expressiveness of the system, as well as its ability to handle differences in schema and data.

3. to demonstrate the notion that the ACP can support different levels of reasoning in the target application, by showing a method for transporting schematic knowledge in the form of new classes or properties to the Target Application's ontology. This helps to assess the ability of the system to express complex knowledge relationships.

4. to import more than one source ontology, to show the ACP can deal with a set of sources of context. The structure of the context model is that it is focused on integrating information across a set of sources to one Target Application, creating a many-sources-to-one-target structure for multiple participants, rather than a full many-to-many mapping structure. This means that it is not necessary to have mappings from source-to-source to be a complete description; source-to-target mappings are sufficient. The key feature in this trial is to show the expressiveness of the ACP allows for different services to be accessed in different parts of the integration, as necessary. This is used to show qualitatively that the ACP can handle multiple parties.

Results of Trial 2.1: Import of Schemas and Mappings

There are two stages to the process of establishing the Shared Semantic View before it can be used to transfer information. The first stage is to import the ontology concepts. The ontology import reasoner which was used in the first case study was reused in this case study, and used the representation schemes which were chosen during that experiment. During the course of this trial, another representation choice was considered for the Topic Map. In the previous experiment, one concept was created for each property and class concept in each ontology, and one association was made which represented the relationship between all instances of a particular class or property. This means that the properties and the classes were considered separately, and the ACP would use the ontology reasoner to determine which properties were used based on the ontological inference. For example, in the reference ontology[7], the

[7]Note, in this experiment, full URI topic naming was employed, but for clarity in this text, the short names are used.

title property represents the title of both *Monograph* and *Manual* class instances. The mapping to the benchmark test ontology described in the RDF maps *title* to *title*, *Monograph* to *Book*, and *Manual* to *Publication*. In the current model (Fig 6.9), the system would query for the equivalent class to *Monograph*, get the property list, find the equivalent to *title* and retrieve the information. However, the question arises as to what to do if the relationship is different in the *Monoograph* to *Book* relationship than the *Manual* to *Publication* relationship? This information can be managed in the Semantic and Syntactic phases of the ACP user-defined reasoner: data relationships can be encoded in the logic of the reasoner. However, there is no indication of this distinction in the Topic Map. The cost of this representation is a map with a greater number of concepts: one for each class:property pair, and an association for each equivalent class:property pair(Fig 6.10). The evaluation of this question was therefore based on the increase in the size of the map, as compared to the model of importing properties and classes separately. The experiment undertaken was to import three ontologies from the benchmark: the reference ontology and two of the 'real world' ontologies provided as part of the benchmark. The class:property pairs were generated by adding a topic for each of the declared properties associated with the classes in the ontology, based on the inference from the default Jena reasoner. Annotation and restriction properties were ignored in both cases. The results can be seen in table 6.4.

	Class and Property Separate	**Class:Property Pairs**
Reference	104	1137
Ebiquity	46	755
MIT BibTeX	60	361

Table 6.4: Total Size of Topic Map in number of Topics for different representations of the Is-A relationship. The number of associations in the map grows proportionally with the size difference of the maps.

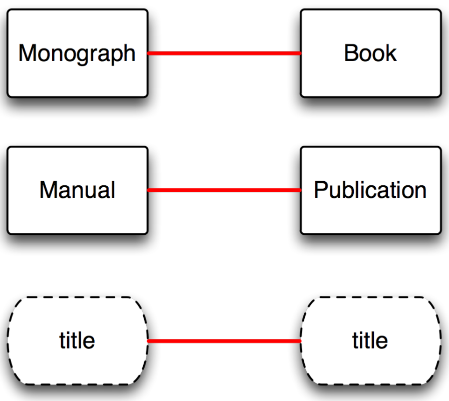


Figure 6.9: Topic Map for Separate Classes and Properties. Red Solid Lines indicate Associations, Classes are Solid Boxes and Properties are Dashed boxes.

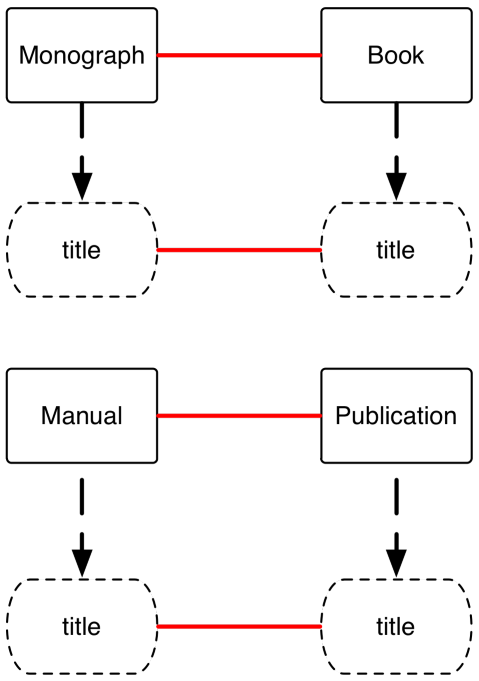


Figure 6.10: Topic Map for Class:Property Pairs, with the dashed arrows indicating the Class:Property Relationship

The result of this trial shows a significant increase in the size and complexity of the Topic Map in the class:property pair case.

Trial 2.2: Resolving Differences in Abstraction Using External Matching Tools

The first case study already showed one example of resolving a key difference in ontological representations: where a difference in class structures meant that information in an instance of an object property needed to be aligned with information in a datatype property. This trial involves the use of the benchmark ontologies from the OAEI alignment set to show that the ACP can resolve other variations in alignment using external alignment information. For this test, a target ontology consisting of the reference ontology with part of its instance data removed was the target application, and instance information from the source ontology in each evaluation test was transferred to show the system's capability. The alignment information was imported from the Align RDF gold standard files. Each alignment test consisted of the reference ontology, a separate test ontology and an associated alignment file. Each test considered either a different variation in the structure of the reference ontology, or a different ontology (in the case of the 'real world' ontologies).

In the trial, 18 ontology alignment tests were performed. The 18 different ontologies which were aligned and transferred are described in table 6.5.

The result for this trial indicates that the ACP could transfer information using the same reasoner across a variety of different ontology combinations. While there was no instance data included in the last four 'Real' ontologies, they required different ontology profiles to load them. This shows an advantage of the ACP model, because it was able to choose a more complex inference profile for one ontology and not another.

Trial 2.3: Supporting different levels of Target Application Ontology Inference

Part of the value of using ontologies to transfer information in the ACP is that they include both schema and data information. Schema inference allows the system to load and use a variety of different ontologies, with different concepts. Schema inference allows the system to answer queries on the nature of the representation of the data, as well as the value of particular data in the ontology.

The fact that the ACP loads the schema information for the ontologies centrally opens up the possibility of altering the schema of the enriched ontology which is returned to the application. This mechanism can, in effect, change the knowledge of the system, as well as its information. Most current applications are not knowledge-driven, and would not likely be able to take advantage of new schema information.

The ability of an application to react to changes in knowledge (schema) and information (instance data) is a measure of its *adaptivity*. There is a scale of

Ontology Code	Concepts Imported?	Alignment Imported?	Instances Transferred?	Notes
101	Yes	Yes	Yes	Identical Ontologies
103	Yes	Yes	Yes	OWL-Lite Generalisation
104	Yes	Yes	Yes	OWL-Lite Restriction
201	Yes	Yes	Yes	Randomised Names
202	Yes	Yes	Yes	Randomised Names, Labels
204	Yes	Yes	Yes	Altered Labels
205	Yes	Yes	Yes	Synonyms
206	Yes	Yes	Yes	French Ontology
207	Yes	Yes	Yes	French Labels
221	Yes	Yes	Yes	Hierarchy Removed
222	Yes	Yes	Yes	Flattened Hierarchy
223	Yes	Yes	Yes	Expanded Hierarchy
224	Yes	Yes	No	No Instances
226	Yes	Yes	Yes	All Data is String Type
301	Yes	Yes	No	Real:BibTeX/MIT
302	Yes	Yes	No	Real: BibTeX/UMBC
303	Yes	Yes	No	Real: Karlsruhe
304	Yes	Yes	No	Real: INRIA

Table 6.5: Table showing different ontology combinations which were executed in the ACP. Instance data was transferred where available.

different possible levels of adaptivity in applications:

- **Non-Dynamic** applications work from a fixed schema, and a defined range of information. The ACP can support these applications by transferring information from external sources, and transforming the information stored in those sources as necessary to fit the fixed nature of the target. An example of this type of transfer is the one shown in the first trial, where missing information is added to a partial profile of a student.
- **Information-Dynamic** applications can take advantage of new data about concepts which they already are aware of. This can be the addition of new instances to the system, for example expanding the list of known publications.
- **Knowledge-Dynamic** applications can take advantage of new knowledge to change their behaviour. In the ontologies, this could mean the addition of new properties, which assert new facts about a particular instance (for example, by adding a new property to a class). An even more adaptive system might be able to take account of new concepts in the ontology, represented by new classes (for example, the addition of the notion of a Movie to the Karlsruhe ontology, which covers only printed publications).

There are different methods for recognising the addition of new knowledge to a schema, depending on the capabilities of the participating systems. Because of this, a full evaluation of this technique is beyond the scope of this work. However, as a proof of concept, two changes were shown in the Karlsruhe bibliography ontology, with the reference ontology as the source. The first was to add the Movie class and its properties to the ontology by using a reasoner which looked for the "TransferClass" type and added it to the target ontology. An equivalence association was then created between the two classes. The properties of the newly-transferred class can then be determined by looking for the appropriate equivalent classes in the target ontology, and adding the new class to their domain[8]. This creates a whole new concept in the

[8]This is optional, and depends on the specific structure of the ontology. Class-Property associations are often inferred from the instance information.

target ontology, which has appropriately reused the properties in the target ontology.

Transfer of properties could be achieved in a similar fashion, with the possible Classes used by particular object properties being recognised using equivalence properties in the Topic Map. The recognition of possible new classes might be achieved in future by taking advantage of the structure of the schema itself. For example, if the $Reference$ classes in two ontologies are equivalent, it might be suitable to query the Topic Map to find subclasses in the source ontology which do not have direct equivalents in the target ontology. These could then be transferred to the target ontology as described above. A similar technique could be applied to properties of classes which do not have equivalents.

Trial 2.4: Multiple Parties

The Structure of the ACP Schema Manager is that it retains a set of services, which are recognised by a unique URI. Each service is associated with a separate ontology, which is stored in and accessed via the schema manager's methods. In the Topic Map, each of the concepts from each service ontology is represented by a topic. As part of the topic metadata, the schema-importing reasoner can attach a type for the concept's service. This allows the Topic Map to maintain separate references to the same concept. For example, one of the benchmark tests imports the default ontology twice. In the Topic Map, this is represented as two different but identical sets of concepts, each typed with a different service type. For example, the two topics representing the *Book* concept in the combined Topic Map would both have the 'OntClass' Type, and each one would have either the 'http://example.com/service1' or 'http://example.com/service2' type depending on which service they are from.

This method of representing concepts means that the ACP is naturally able to extend to more than two parties. In practice, each additional ontology is recognised as a new service type, which has a type topic that is added to the map along with its concepts. The map can therefore grow incrementally as new ontologies are added, and can treat new services like other types which are attached to topics in the map.

The ACP Shared Semantic View manager provides for persistent storage of different maps in the system. This allows reasoner functionality to be broken up and reused, as illustrated with the schema importer and Align RDF importer user-defined reasoners. This means that these reasoners were able to be used for a service environment with more than two parties without alteration.

In the experiment, three ontologies were imported into the Topic Map: one target application and two sources of context. The target application was the benchmark ontology, while the MIT and Ebiquity ontologies were used as sources of context. The three ontologies were imported one by one by the default schema importer, and the alignment reasoner was used to map 296 alignment statements from three description files.

The layout of the services was derived both from the mediation scenario, where several sources feed one target application, and by the nature of the mappings, which were reached by loading the separate one-to-one alignment mapping files for the two source ontologies to the reference ontology.

This configuration suggests at least two approaches to transferring information: The first was to call the information transfer reasoner several times, each time choosing a different source. The second method extended the reasoner to iterate across the services during the identification phase, creating a combined list of entries. While the test alignments and data were not differentiated, this shows that in a real-world system the ACP can gather and compare information from several services and combine it at will. This simple scenario can easily be extended to more complex relationships, for example where information in one service is used to select instances from a second source to be transferred to the Target. A concrete example of this might be where one bibliographic ontology maintains a list of relevant publications for a particular subject, but a second, more general ontology contains the full bibliographic information. The list of entries from the first service could be used to select instances from the second service to populate a complete bibliography in the target application. However, due to the fact that the included ontologies did not include instance data, these approaches were not fully evaluated.

Key Findings for the Case Study

The first finding for this case study is that the advantage of separating the alignments across class:property pairs is that each alignment can then be described differently, allowing different reasoner behaviour for different sets of classes and properties. However, the high increase in map size would seem to rule out the use of class:property pairs. A significant increase in the map size alone would not necessarily be sufficient reason to avoid this technique. There are several possible ways to avoid a large increase. For example, the class:property pair technique could be reserved for specific designated properties which are diversely reused across the ontologies. The main reasons for distinguishing the properties of different class:property alignments come either from the differences in the restrictions of the property, or from some property of the alignments at the data level. The ACP does not currently support property restrictions, and so it would be preferable not to duplicate that information in the map. Instead queries about ontology restrictions could be made from the ontology reasoner, which supports them. The second reason to distinguish alignments is because of differences at a data level. The alignment and schema information provided in this test case does not include that distinction.

While there may be example scenarios where this technique could prove important, for the purposes of this case study and similar scenarios, Classes and Properties were independently imported and correlated using ontology inference.

The result of the second trial with different ontology pairs showed that using the ACP as part of a complete semantic mapping workflow yields useful advantages in being able to reuse ontologies and overcome variations in differences in abstractions. These differences represent a broad range of the possible variations in related ontologies, and the fact that the ACP can bridge those gaps, using the same transfer reasoner with different alignment description, shows the advantage of semantic interoperation as a basis for creating context integration. This demonstrates the advantage of the use of semantic interoperation as a process for establishing context integration pathways. Much of the effort in handling the variation between the different ontologies has been achieved by the use of an external ontology mapping tool, the alignment data for which is imported into the ACP Topic Map.

The transfer included a small (between one and ten) number of property values, some object and some data properties. The object property instances were aligned in the two ontologies.

In this trial, the instance data included in the benchmark ontologies was not extensive, and so the case study test for information transfer was a relatively straightforward one. This arose from the fact that the input information was restricted to that which was provided by the benchmark. The advantage of the ACP, as demonstrated in the first case study, is that it provides a mechanism for creating complex instance mappings, and resolving structural differences such as object-to-data mappings, where necessary.

The third case study objective was fulfilled in trial 2.3, which presented a set of different ways in which applications can accept changes to information and schema, and demonstrated how the ACP could support them.

The alteration of the schema in this fashion can take place for a number of different reasons. The key novelty is the notion that the schema could be altered in this fashion. One of the advantages of this model is that the reasoner creates a topic and associations for the new concept as normal, typed as a native concept to the target ontology. This means that this part of the Identification phase can be constructed so as not to affect later parts of the integration process.

This notion of altering the schema as well as the data of a participant's ontology has the potential to drive complex behavioural changes in an application in response to changes in the type of context information available, as well as its value. The model-exchange method used for context-informed interoperation is key to making this kind of exchange possible.

Finally, from the multi-party trial, the ACP system can be said to be able to support multiple services in a flexible fashion. The additive nature of the maps means that reasoners can be developed that handle single tasks in isolation, which can be run more than once as required, as well as more complex multi-party reasoners which can interact with a variety of types, services and alignments.

6.3.3 Overall Findings for the Case Studies

One of the key features of the ACP is that it permits a wide variety of different approaches to representing ontological information in the Shared Semantic View. The comparison of different features in this evaluation is based on a particular set of criteria, which are derived from the context informed domain.

The need to integrate contextual information provides for certain key assumptions to be made about what is preferable in representing ontology information in the ACP. The first and foremost feature is that the system is designed to provide for context information transfer, this means that the focus is on transferring information. Other applications, such as ontology mergers, are targeted at creating substantive expressions of domain knowledge from peer sources.

This difference means that the ACP does sacrifice precise ontological logic in favour of expressiveness. The assumption is that the Target Application can make use of some, if not all, of the information returned, and so a wide variety of transformations and translations of the data are provided for. The evaluation case studies described above have produced some key findings about the nature of ontology mediation, particularly with the ACP approach. These are outlined below.

Ontology Reasoning

Much of the behaviour and features of the ACP depend on the relationship between the Topic Map and the Ontology Reasoners. The retention of the ontological reasoning features was seen as a key feature in being able to take advantage of the semantic nature of the data representations, but this came at a complexity cost. Topic Maps are not usually accompanied by general reasoners of the type which OWL ontologies enjoy.

The experience of the first case study in particular led to the conclusion that information held within the Topic Map should ideally only be that information which cannot be held within the native ontology.

There is a significant cost to this, which is that the model consistency cannot be guaranteed by the system. The use of the Topic Maps means that the data is taken out of the ontological reasoner's domain, possibly transformed by some reasoner, and then placed in a new ontology. The lack of an reasoner means that careful design of the integration process is required to ensure that the resulting model is valid. This is even more important when changes are made to the schema, as the capabilities of the target application itself must be considered.

The Evaluation of this implementation did not include making use of property cardinality or value restrictions from the OWL reasoner. The extension of the system to handle these would require a matter of extending the schema manager to provide appropriate methods for querying such restrictions. This was beyond the scope of this evaluation of the system, but might provide useful future area for investigation.

To summarise, by using a structure which is not amenable to ontological reasoning, the advantage is that the system can express the information most useful to the effective transfer of information between the ontologies. However, this is at the cost of losing the ability to automatically ensure consistency across the models. This finding is likely the defining one in choosing whether this approach is appropriate for a particular contextual scenario.

Ontology reasoning, particularly with more complex reasoner features, can impose a very high memory cost and time cost on an application using ontologies. This has been observed in this evaluation, and in other literature [Lewis et al., 2006]. The ability of the ACP Schema Manager to load different reasoner profiles as required (for example, full transitive reasoning for case study 1, and the default Jena profile for most of case study 2), means that a pragmatic approach to reasoning can be taken, where the correct reasoner profile can be chosen for each ontology based on the parameters of the integration.

Alignment Generation

In the second case study, the alignments were elicited from the pooled findings of the participating matching algorithms from the alignment workshop. This allowed the case study to be undertaken using both ontologies and alignment descriptions which were generated externally, without the ACP's design in mind.

The combined experience of the two case studies shows that the ACP can take advantage of independently-created alignments, which can be general alignment descriptions, or case-specific and rich alignment relationships such as those described in the user model alignment. It is likely that more complex integration scenarios would require a combination of both approaches, where a set of case-specific types and reasoners is able to operate over a variety of different specific ontology combinations that can be aligned using an external tool.

The process of finding and describing the relationship between ontologies has been divided into steps which are intended to build from a generic description of relationship (matching) to a committed conceptual relationship which can be executed. Some of the properties of the ACP contribute to making this process achievable. The first is that the use of the Topic Map to represent the mapping and mediation information means that information from different sources, such as external alignment descriptions, can be combined into one integrated and persistent structure.

This persistence allows for different information to be contributed by different reasoners, separating concerns both as regards particular ontologies, but also as regards different levels of knowledge and abstraction. As shown in the case studies, this means that the map can represent ontological and data relationships in a variety of ways and presents a unified method for accessing the information. The cost of this flexible model is that the structure of the map is dependent on good behaviour of appropriate reasoners; there is no way to tell if a map is correct or not, except by inspection.

Reasoner Composition & Reuse

The user-defined reasoner interface in the ACP is the means by which the mediation is executed. The structure of reasoners in the ACP reflects the nature of the system: they allow for a high degree of freedom to access different levels of abstraction in order to achieve data integration. In the process of developing the case studies, it became apparent that there were different categories of use for particular reasoners. Some reasoners, such as the one used to import OWL ontologies, were able to fulfil the same role in several different integration scenarios. These utility reasoners benefited from being generic and self-contained. On the other hand, the specific reasoner responsible for transferring user model information in the first case study was specific to that case.

There was an advantage in effort and efficiency in the ability to reuse generic utility reasoners in the ACP. The architecture was specifically designed to support this. This is embodied in the ability of reasoners to call each other in a chain, permitting the system to load different functionality at runtime, and then make use of that loaded reasoner as necessary. For example, in the second case study, the Align RDF importer could be reused to import the relationships from the mapping description file.

The reasoners in the ACP are separated to some extent from the specifics of particular ontologies or concepts by the use of the type system in the Shared Semantic View. Reasoners can add and reuse particular types as necessary to build a coherent picture of the shared semantic view. This is similar to the way which an upper ontology might be used to fix particular semantics within a shared information space. The advantage of the ACP model is that different reasoners can act with complete freedom, at the cost that they do not have a universal agreed set of concepts. The higher integration cost is offset by the expressiveness.

The ability of ACP reasoners to combine semantic and data knowledge as required allows the system to execute some powerful features. For example, the instance alignment discovery process in the User Model case study shows that the system can make use of the power of the mappings it has along with key information to resolve specific semantic gaps automatically. The highly specific nature of the types in that relationship would be difficult to express in a generic ontology, and so reducing the alignment problem to one of local, specific responsibility is appropriate and useful.

6.4 Comparison with the State of the Art

This section will compare the ACP and the context-informed approach as evaluated above with some of the key systems described in the Literature in previous chapters of the thesis. Two principle areas were examined: semantic mediation and its applications in context integration. This section will compare the ACP approach to the ones described by those systems surveyed in the State of the Art under the headings which were used for the analysis of those systems.

6.4.1 Comparison of the ACP to other Context Systems Reviewed

The original intention of the ACP system was to provide a means for integrating context information from external sources into the knowledge of a target system. However, the nature and properties of what constitutes context present a difficult problem, with no likely general solution [Bolchini et al., 2007]. The design of the ACP had a goal of being general, but the reality is that there are limitations to the suitability of the ACP as a context system, as well as advantages. In this section, the ACP is compared with the Context Systems reviewed in Chapter 2, under the same analysis headings as those systems.

Information Model

The key feature which the ACP does not have that is present in other systems in the literature is the Context Model. Both Construct [Dobson et al., 2007] and CoBrA [Chen et al., 2004a] employ upper ontological models which have been created in advance to describe context. These models describe key features for the pervasive domain, such as representations of users, space or time.

One of the key differences in the ACP approach is that it does not link different participants through a pre-existing, shared ontology, but rather through the use of ontology mediation technology it aggregates links and descriptions of concepts in a Shared Semantic View. Each SSV is different, and is created based on the requirements of a specific exchange. The advantage to this approach is that the ACP system is able to handle a wide variety of information and create complex relationships between different concepts. The ACP system imposes few preconceptions on the domain or the content of the participating models, and information can be altered using different chains of reasoners at the knowledge representation, information content or data value levels. This allows for wide semantic gaps to be bridged by the system.

The key advantage of the use of a pre-existing upper ontology is that it provides a clear target for integration, and it allows systems such as CoBrA and Construct to encode detailed knowledge about the concepts within the ontologies, which improves reuse and lowers the integration overhead because the known common entities can be supported by context reasoning information within the context system. However, the experience of CoBrA in particular seems to point to the need for the creation and integration of customised extensions to the upper ontology in order to support different specific applications [Chen et al., 2004b]. The benefit of a pre-shared ontology is therefore not as clear cut as it initially appears to be.

The final decision on the suitability of the semantic mediation approach to context depends therefore on the cost of integration: if an upper ontology and its associated system can appropriately cover the majority of the concept space, and the cost of extending that concept space in the event of change is low enough, then it might be the preferred option. On the other hand, where no upper ontology exists, or where the domain of the system is complex and specific, then the use of the ACP approach provides a means to offer the same or better integration.

Participants

The ACP system is designed to work with participants that can express their information ontologically. In particular, the ACP requires systems to share both the schema and the instance information which they can make available for transfer.

For sources of context, this is not necessarily an enormous burden, but for Target Applications, the requirements can be significantly higher. The need to exchange schema information arises from the fact that no pre-existing schema has been agreed between the participants. Conventional query-based exchanges such as those used by Construct depend on a known schema, and part of the power of the ACP is that it can alter the schema of the target for improved knowledge. The notion of extending applications' knowledge as well as their information is potentially very powerful, as it can permit substantial changes in the way applications behave based on their contextual surroundings. The use of multi-model adaptive techniques [Conlan et al., 2002] in the Target Application is one possible approach which would allow for this kind of behavioural flexibility.

The ACP allows for systems to bring their own knowledge in its own representation to the system, and handles that interaction directly through the process of mediation of ontologies. This is different to the Construct approach, which draws information from ontological and non-ontological sources.

Architecture

The architecture of the ACP is highly centralised. In particular, the system loads the ontologies as well as the mapping information for each participant. The Context-informed approach means that the system is designed to allow the Target Application not to need to form any queries relating to context, but rather the centralised mediator is responsible for making the changes appropriate to the system. It is difficult to see how the ACP could be separated into a distributed architecture, given the highly integrated nature of the schemas within the system.

The value of distribution in the form of agents or nodes is that it allows for lower-power systems to perform context integration. There is little doubt that the ACP is a highly resource-intensive architecture. However, it is possible that the distribution could be achieved by other means: since the ACP takes responsibility for discovery, integration and querying sources of context, the overhead on a target application is lower because it does not need to know about context, but merely exchange with the mediator in its own language. Similarly, if the sources of context are only passing information to the mediator uni-directionally, then the load on them might not be great.

Conclusions

The evaluation of the ACP as a context mediator depends on the assertion that context can be represented ontologically, and that it is desirable for applications to be able to hand off the process of integrating context to an external party. There are several key reasons why this might be an advantage. The most prominent reason to do this is that it allows for a clean separation of concerns for the development of applications. The definition of context used in this thesis is to perceive contextual information as highly variable in form and representation, and for its inclusion to be a matter of sometimes complex transformation and translation. These characteristics lend themselves naturally to the notion of a context mediator with a global view of the systems in a particular situation.

The value of the ACP approach as compared to other context mediators and brokers is then dependent on the nature of the information heterogeneity can accommodate. The ACP differs from other systems surveyed by not attempting to pre-determine the nature of contextual information in the system. This has the advantage of permitting the system integrator total freedom in expressing the nature of the relationship between different concepts. The attendant cost to this is that there is a need for the integrator to develop reasoners for specific requirements. The composition and reuse of reasoners provided for in the ACP goes some way towards addressing this cost.

One advantage of the approach given in the ACP is that the Topic Maps themselves can be a useful representation of the context which a particular application is being used in. The Maps contain the type an entity information necessary to describe the context, and can be inspected and manipulated as semantic structures.

The function of the context brokers can be seen as needing to address knowledge and data, and in this case the ACP does directly support manipulating the information held within the ontologies, based on the type descriptions in the map and the functionality of the reasoners. The ACP does act in a similar way to other context mediators, with the advantage of not prescribing what context 'is'. This was shown in the evaluation by the demonstration that the system can handle the key requirements of context: reconciling differences of abstraction, correlating multiple parties and translating data semantically. A summary table of the key features of each architecture is shown in table 6.6.

	CoBrA	**Construct**	**ACP**
Information Model	SOUPA	When, Where, Who Ontologies	Topic Map
Participants	Agents	Services using queries	Services using Model Exchange
Architecture	Broker-centric	Gossipping Nodes	Mediator-Centric

Table 6.6: Feature Matrix comparing the ACP and other Context Systems

6.4.2 Comparison of the ACP to other Semantic Mediators Reviewed

In attempting to advance the design of context systems, the ACP has been inspired by the trend in research in ontologies into greater Semantic Interoperation. As outlined in Chapter 3, initial attempts at universal global ontologies have gradually given way to mapping expressions and ontology manipulation. Ontology alignment, merger and mediation arise from the recognition of the fact that it is difficult and even arguably undesirable for applications to be required to use the same ontology, and that finding links between those ontologies is a useful approach. In this section, the ACP is compared with the Semantic Mediators reviewed in Chapter 3, under the same analysis headings as those systems.

Nature of Articulation

The ACP is a novel approach to semantic mediation, in that it generates the ontology articulation as a semantic structure (the Topic Map) from the alignment descriptions without a pre-existing model. The approach of using a model to describe the concepts and entities within the ontologies is similar to that of the WSMO model [Roman et al., 2005]. WSMO includes more concepts in its ontology because of the requirements which arise from the web service domain.

Both WSMX[9] and the ACP are similar in that they express their articulation in an external structure which is not the same as the ontology language of the participant. In WSMX, the participating ontologies are mapped to the upper ontology, while in the ACP a separate representation is created which reproduces the concepts in the participant ontologies and maps them. One difference arises from the domain for WSMX: it is focused on the choreography and planning of web services, and data relations are expressed in a separate model. The ACP represents data and conceptual relationships in a unified SSV.

Ontological Linguistic Independence

Of the three mediators, ACP, WSMX and DRAGO, none can be said to be truly linguistically independent. However, given that the DRAGO reasoning algorithm depends on the tableau algorithm, it is probably the least linguistically independent. The ACP's independence arises from the provision of functionality within the Schema Manager for the relevant ontology language, while the linguistic independence of participants in the WSMX environment depends on the provision of a suitable importer and parser. However, the authors of DRAGO indicate that it is also possible to import ontologies into the DRAGO environment with an appropriate parser.

[9]the execution environment for WSMO.

Judging linguistic independence is therefore somewhat unclear. The ACP provides a way to extend its facility to maintain knowledgebases in their native reasoner environment, but this depends on integration effort and the availability of suitable libraries. On the other hand, DRAGO and WSMX import the ontological information, which may be more independent if the representations in question can be satisfactorily imported.

Mapping Representation

In the ACP, mappings are represented using the associations between topics in the map. DRAGO represents mappings using the C-OWL language, while WSMX derives its mappings from the WSML mappings to WSMO. The DRAGO representation is probably the most constrained, in that it is restricted to the associations which conform to the first-order logic of the tableau algorithm. Where WSML cannot adequately describe a relationship, mediators, which can execute different functions within the choreography, can be employed to perform data or other alterations. The ACP has a similar notion of user-defined reasoners to the WSMX mediators, but they are not directly represented in the SSV, instead they operate on the mapping types, schema information and data values in the Topic Map and the participating ontologies.

The ACP sits between the two systems in the spectrum of mapping representation. It has a single unified view of the semantic relationships, unlike WSMX, but on the other hand the mapping includes a wide variety of different types and relationships, while DRAGO has a pure C-OWL representation.

Nature of Internal Representation

The three systems have rather different approaches to the issue of internally representing participating ontologies within the system. While WSMX imports the ontological concepts into separate representations, DRAGO retains the distributed knowledge bases with local reasoning and distributes the integration. The ACP blends the two approaches by importing a representation of the concepts to describe the mappings, but querying the natively reasoned ontologies for information.

The value of the ACP approach is that it reduces the complexity of the mapping representation and frees it from constraints of logical reasoning. The unified SSV can express a wider spectrum of qualities about different mappings because it is tied to a particular reasoner.

Data Transformation

DRAGO does not express data transformation, or execute such transformations. WSMX has two approaches: where the information is expressed in WSML, the system can execute data transformations as described. For information not expressible in the rule language, mediators can be employed to make other changes.

The ACP takes a different approach, because it does not have the ability to decide on domain-derived operations, the ACP uses mediator-like user-defined reasoners for all execution. The only distinctions are in the reasoner interface, which can be seen to have importation, SSV manipulation and model enrichment functionality. Unlike WSMX mediators, which are separated, ACP reasoners are not contained in their functionality. ACP reasoners are also permitted to call one another, to allow them to chain functionality between different specific reasoners, to lower the complexity and improve reuse. Rather than a rule language, the ACP reasoners are directly implemented in Java. This has disadvantages for security and ease-of-authoring but is significantly more powerful and allows reasoners to perform powerful operations.

Knowledge Transformation

The ACP is different from both of the other systems in the way it addresses the ontological schemas of the participating ontologies. DRAGO makes use of conventional T-Box reasoning, which the ACP can take advantage of in the Schema Manager. WSMX can use WSML importation to reason over the schema. A novel feature of the ACP and the context-informed approach is the notion that the target application's schema itself could be manipulated to add new or different concepts to the system. This is a powerful notion, in that it allows the system not only to take new information from its surroundings, but also to offer the knowledge of new concepts to the target application. While the potential for this is substantial, it is not without cost in that it requires a target application capable of taking advantage of new schematic knowledge.

Mapping Importation

All three systems are similar, in that they can import different mapping descriptions into their internal format. The ACP and WSMX can import and modify their mappings for data concerns, which are not relevant to DRAGO. With regards to instance mapping, DRAGO imports an external instance mapping description, while this evaluation has shown that the ACP can determine those mappings programatically, thanks to its ability to examine data and schema information.

Conclusions

A summary table of the different comparisons made in this section is in table 6.7. The key advantage of the ACP is that it combines part of the native reasoning of DRAGO with the flexible custom reasoning of WSMO.

DRAGO's method for mediation arises from a fully inferred distributed tableaux algorithm. While this method is highly automated, it is limited because it cannot make use of information below the schema level. On the other hand, both the ACP and WSMO need user-defined reasoners/mediators to perform the transfer. The advantage of this custom reasoning is that it also allows for data transformation.

The key differences between WSMO and ACP lie first in the fact that WSMO uses a more complete upper ontology, which describes the web service domain principally, and second in the separation of data an schema. The WSMO ontology is focused on the schematic information, and using a sequence of mediators to perform the exchange. On the other hand, the ACP uses its SSV to represent a more complex mix of schema and data information. The advantage to this is in the ability for different reasoners to co-operate.

Neither of the systems seem to feature schema alteration in the way described in the ACP.

6.5 Conclusions

This chapter has presented the evaluation of the ACP implementation and the context-informed approach. Based on the evaluation methodology, there were three key requirements for a semantic mediator to be able to fulfil for it to be an effective context mediator. These were: handling representational differences in the participant ontologies, handling data differences in the participants ontologies and handling multiple parties in the integration. The nature of the system is that, as a semantic mediation system, the effectiveness of the system on a particular set of ontology features is the same, regardless of the specific domain of those ontologies. This means that if the system is effective for certain features with one set of ontologies, it is likely to be effective with similarly structured ontologies in another domain. It is this which allows the conclusion to be drawn about the effectiveness of the ACP based on the evaluation tasks. The key advantage of the use of Topic Maps is that they are a flexible and expressive method of creating representations of both the aligned ontology concepts and all the necessary metadata to describe the alignment at the schema and data levels. This integrated view means that each reasoner can manage its own types and can access all the information in the ontology to the reasoners at any point through the schema manager.

The innovation in the approach therefore arises in two ways: first, the ACP provides an improved way of exchanging information between the Target Application and the sources of context. The model based approach has demonstrated in this evaluation that it can include new information in the schema offered to the target application, as well as data which has been converted to the representation needed by the target. The second innovation is in the representation of the shared semantic view, which

#	**Property**	**WSMO**	**DRAGO**	**ACP**
1	Nature of Articulation	WSMO Ontology + WSML mapping	C-OWL Bridging relationships	Topic Map
2	Linguistic Independence	Dependent on OWL and C-OWL	Mediation provides conversion to internal formats	Dependent on SchemaManager integration
3	Mapping Representation	WSML Rules	Bridge Rules + Instance Correspondence	Topic Map Types & Associations
4	Nature of internal Representation	WSMO Ontology	DDL Knowledgebase	SchemaManager native ontology reasoner
5	Data Transformation	Mediation + WSML Rules	None	Reasoner Interface allows data manipulation
6	Knowledge Transformation	Mediation + WSML Rules	T-Box Reasoning	Ontology Reasoning and schema manipulation through reasoners
7	Mapping Importation	Mediation	OWL-C or converting parser	Converting parser in form of reasoner

Table 6.7: Ontology Mediation Systems Compared.

allows both data and schema information to be represented separately from the native ontological inference. This has advantages both in allowing for transferring data across complex semantic and syntactic differences, and, as shown in the evaluation, to create user-defined reasoning which is flexible and reusable.

In conclusion, the ACP seems to represent a novel approach to ontology mediation which supports effectively context-informed semantic interoperation. The solution is reasonably general, within the constraints described. There is still considerable exploration potential for this kind of mediator, and it is by no means the best approach in all cases. However, the notion of new knowledge and model-oriented exchange seem to hold the potential for justifying the considerable effort which any context system requires to deploy it by supporting applications in responding to their external knowledge with substantial behavioural changes.

Chapter 7

Conclusion

This chapter presents the conclusions of the thesis. The chapter begins by examining the original objectives of the work and assesses the degree to which they have been fulfilled. The contribution of the work, including contribution to the state of the art, is then listed. The work presented in this thesis on context-informed semantic interoperation creates significant opportunities for further development, particularly as semantic technologies mature. Some possible routes for this investigation are discussed in the Future Work section.

7.1 Objectives & Achievements

The central research question of this thesis asked whether it was possible to use a semantic interoperation-based context mediator to improve a target application's knowledge, and whether such a mediator could support different types of knowledge enrichment.

The primary objective of the thesis was to design an architecture for providing innovation in context-informed semantic interoperation. This architecture was intended to support the notion that contextual information could be gathered from a wide variety of sources which were not known at the time of design for the target application. This objective was achieved through the creation of a context mediator which could represent and query ontological knowledge bases and express richly-described alignments between different concepts. By choosing not to create a complete model for context in advance, it was proposed that the system was more flexible, and allowed for a concentration of effort on the specific challenges of a particular context scenario. Further effort could be saved by the use of general tools for semantic interoperation, such as ontology matching tools, to undertake parts of the integration effort.

The Context mediator described in this thesis differs from those that have been created before by using a lightweight semantic network (a Topic Map) to provide an active, queryable structure to represent the Shared Semantic View. This structure supports the semantic interoperation approach by combining information from several sources to build the shared view, rather than linking a pre-existing ontology. This design arose from the results of completing the second thesis objective, that of a state of the art review that reflected both Semantic Interoperation and Context.

Semantic interoperation in this context mediator is based on a view of information transfer that has three phases:

1. The **Identification** phase, where appropriate information is discovered in the sources of context.
2. The **Semantic Transformation** phase, which executes the query and performs any alterations to the information to be transferred.
3. The **Syntactic Translation** phase, which takes the data from the Semantic Transformation phase and arranges it appropriately for addition to the Target Ontology.

This model was successfully applied to fulfil the objective of improving the knowledge within the target application in a variety of different ways. The evaluation of the ACP[1] implementation successfully demonstrated that the framework could bridge a variety of differences between ontologies and transfer information to a target ontology.

[1]Adaptive Contextual Portal

The data transferred could be selected and arranged on the basis of both schema and instance data queries in the sources.

The creation of the shared semantic view and the querying and alteration of ontologies was executed by a set of user-defined reasoners, which were loaded on demand by the system. These reasoners were reusable, and their operations were based on input from the schema and the shared semantic view. This allowed some of the reasoners to function across different integration scenarios (for example, the ontology uplift reasoner was used in both case studies to import OWL ontologies).

The final objective of the thesis was to gain an understanding of the advantages and disadvantages of context informed semantic interoperation. The evaluation of the system showed that the ACP could support different types of target application, from adding information about known instances in the target application up to the alteration of the schema to add new classes, properties and instances. This notion of altering the application's schematic knowledge, in addition to its instance information presents a powerful potential for context, by allowing sufficiently dynamic applications to take account of new types of information made available by context.

The use of the Topic Map for the shared semantic view creates a flexible, expressive representation of the integration between different ontologies. The shared semantic view provides user-defined reasoners with access to a collaborative environment to query and retrieve information at all levels of the transfer (instance data to schema representation) and to record new metadata as required. This feature makes the ACP capable of highly complex and rich context integration even for relatively simple applications, by allowing a chain of reasoners to perform extensive semantic and syntactic alteration to the data.

The ontological nature of the approach might be considered to have a performance and modelling complexity disadvantage in comparison to key:value based methods for managing context. However, there are several features which the ontological approach provides which cannot be easily reproduced in other methods. The first of these is the structured description of knowledge and information, which allows for complex data relationships and for the transfer of schematic knowledge. The second factor is that the use of semantic interoperation means that the participants can control complexity of their ontologies independently. Simple systems can therefore emit information in simple ontologies, and more complex ontologies can be used where they are necessary. This is a particular advantage of the semantic interoperation approach.

The second key disadvantage of the ACP as a context mediator is that, without a reference ontology for context, the integration process can be a complex and difficult one. Not only do the ontology alignments need to be found somehow, but appropriate reasoners need to be built. This cost is partially reduced by the ability to reuse reasoners, and the use of external, pre-existing tools to assist in the creation of the shared semantic view. The second counter-point to this cost arises from the need for those systems which do employ pre-existing ontologies to provide for application-specific extensions. The cost of creating new ontological concepts (or even complete lower-level ontologies) is likely to incur the same or greater effort than creating the semantic interoperation in the ACP.

In comparison to other Semantic Interoperation tools, the ACP has a key disadvantage compared to purely ontological approaches because of its low degree of inference. The use of the Topic Map means that there is a need for user-defined reasoners to supplement the ontological inference. The counterpoint to this is that there is greatly increased expressiveness, and the ability to use the data to find things like instance alignments automatically. The ACP can operate on both the instance information and schema representation levels, and use any of that information in its custom reasoning, while the fully ontological approaches are restricted to schema information reasoning.

7.2 Contribution to the State of the Art

The major contribution of this thesis is Context-informed semantic interoperation, an approach for managing context without a pre-existing ontology which is specifically describing what context information. This approach differs from previous work in that it uses semantic interoperation combined with custom reasoning to solve abstraction differences across different sources of context to enrich the knowledge of the target application. The approach is also novel in that it uses a whole-model exchange mechanism, which can expand the schema of appropriate target applications. This means that the target applications can be enriched with new types of knowledge, derived from context, as well as improved information about the knowledge that they already have. This contrasts with traditional context, which has been focused on the transfer of information from external sources through fixed knowledge representation models.

The minor contribution of this thesis is a novel approach to the design of a semantic mediator based on the use of Topic Maps. This is novel in using the Topic Map as a flexible representation of the content of the ontologies and their relationships, while retaining native ontological reasoning for queries, as appropriate. This semantic mediator also differs from previous systems by combining both ontological and custom reasoning. The use of Topic Map technology is not absolutely required for this approach to work, but Topic Maps demonstrate the principal capabilities needed to support the form of ontology mediation provided by the approach.

Evidence of the contribution to the state of the art comes in the form of three publications:

- O'Connor, A. and Wade, V. (2006) Informing context to support adaptive services. in *AH'06 Proceedings of the Fourth International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems*, pp. 366-369. Springer, Berlin/Heidelberg
- Dagger, D., O'Connor, A., Lawless, S., Walsh, E., Wade, V. (2007) Service Oriented eLearning Platforms: From Monolithic Systems to Flexible Services, IEEE Internet Computing Special Issue on Distance Learning
- O'Connor, A. and Wade, V (2008) Semantic Interoperation to Support Context in *5th International IEEE Workshop on Management of Ubiquitous Communications and Services, Held as part of IEEE NOMS 2008*, Salvador

Bahia, Brazil

These represent the publications so far, but it is intended that the results of the evaluation and other collaborative research will be published in appropriate venues in the near future.

7.3 Future Work

There are two likely principal directions for this work. The first deals with the improvement of the mediator itself, by adding and extending features both to the ontological and shared semantic view components. These improvements reflect the advances in the standards and technologies that govern those components, as well as richer feature sets that arise from experience with the system.

The current ACP implementation in particular is closely associated with OWL, and significant evolution in the OWL standard, particularly in the new major version, OWL2. OWL2 has already begun to appear in experimental form in some of the tools, and the standard is under development in the W3C.

There are several important issues relating to the use and management of mappings in the ACP. Topic Maps provide several more annotation methods, including role types for associations, which might be applicable in certain use scenarios.

In terms of applying the technology, the next stage is perhaps to develop some adaptive applications capable of taking full advantage of enriched models with improved schema and data from the context environment. One area of application for this is in the 'Linked Open Data' domain, which is a pragmatic approach to making structured information available on the web.

These areas are only a part of the large set of challenges which remain in this area, and it is likely that as knowledge of the system, and of semantics in general, improves, that new avenues will become apparent.

7.3.1 Supporting Ontology Features

The first extension which might be desirable in the ACP would be to examine using reasoners to take account of different annotation properties and restriction properties in ontologies. In the current OWL ontologies, restrictions can include information about what class instances a particular object property can refer to, or the cardinality of instances (how many of a particular thing an instance can have). These properties allow for the system to check the details of model consistency and coherence between the instances and the schema.

Model consistency in the ACP is currently a matter for the design of the reasoners. A richer ability to process property annotations would be of considerable use in the syntactic phase, in making sure that new additions comply with the consistency requirements of the model.

In addition to this extension, there are several features of the new OWL2 standard which point to useful opportunities to improve the ACP. OWL2 includes substantial additions to the standard, and there are numerous extensions and corrections to the semantics and structure of OWL2 Ontologies [Grau et al., 2008]. Many of these changes will likely be handled by the fact that the ACP can load OWL2 ontologies in an appropriate native reasoner. There are four features of OWL2 that would appear to have significant impact on semantic interoperation. Two of these relate to the structure or behaviour of OWL2 schema information, and two relate to the way data in OWL2 models can be checked and instanced.

The two features of OWL2 that might prove useful for the way reasoners interact with the schema are the addition of 'Qualified Cardinality Restrictions' (Q.C.R.) and new types of 'relational expressivity' for properties. QCR affect the reasoners because they allow property restrictions to include more complex cardinality requirements for consistency, which must be accounted for if the ACP reasoners are to be able to account for model consistency. New relational expressions in OWL2 include 'part-whole' property relationships, and properties of properties. Both of these features could change how information propagates across a shared semantic model, and the mechanism for determining appropriate alignment descriptions for these relationships will be needed.

On the data side, OWL2 includes two features which mean that the schema provides a much richer description of data. The first is the ability to specify datatype value restrictions (for example maximum and minimum values for integer datatypes). The second feature is inspired by database structures and provides for a particular property to be designated as a ‘Key’, which uniquely identifies an instance in the ontology. In both of these cases, there is an opportunity to try and extract some of the features of a specific ontology automatically as part of the user-defined reasoner process, which could have great advantage in reusing reasoners (a key property in particular could be useful in examples such as the first case study of the evaluation of the system).

The benefit of developing the ACP towards OWL2 will depend on how many of the new features are used and how they are implemented in real ontologies. The OWL2 standard includes an attempt to address the problem of large, complex ontologies inference tasks through the OWL2-EL profile, which aims to provide consistency, subsumption and expression checking in polynomial time. This could have a particular advantage in large ontologies. Similarly, there are profile to support large instance knowledge bases (OWL2-QL) and rule-based inferencing (OWL2-RL) [Motik et al., 2009]. The use of these profiles in the ACP might provide better performance in certain cases.

7.3.2 Extending the Shared Semantic View

In the current implementation of the ACP, there is a general assumption that associations are between two topics. The Topics are described by their types, and the association is described by its type and its reifying topic. One area of metadata not explored in the ACP is the notion of different Association Role Types. These would allow the ACP to describe the different participants in an association using the type system (ie with Topics to describe each role). The advantage of this model is two-fold. First, it would allow associations to characterise its members depending on their role in the association, and second it would provide a means for exploring associations with more than two members. This could be used to create 'Operator Topics', which could describe logical relationships between concepts. One example of this would be a concatenation operator, where a target concept is associated with the sum of strings from two source topics. Rich roles would allow the reasoners to perform operations based on the memberships of those associations.

The second area of interest would be in the use of the map of associations themselves. The reification process creates a set of topics which describe the articulation of the ontology. From the map's perspective, these topics are no different from those which represent concepts. It might therefore be useful to investigate creating associations and maps between the reifying topics. One example might be to create dependency relationships between associations, so that if one association is discarded or changed, the implications of that change can be propagated across the map.

7.3.3 Linked Open Data

Linked Open Data is concerned with making structured, machine-readable data available using web technologies and an agreed set of best practices, such as those in [Bizer et al., 2007]. The data exposed as linked open data has several key properties: the first is that it is published in an RDF data model, and the second is that the data is linked to other relevant information through RDF links. A third aspect is that it is 'open', which means that the data is made available over the web using open, agreed standards.

Resources in LOD are identified using URIs, particularly http URIs. A key feature of Linked Data is that the schema vocabularies used to structure the data should reuse, to the greatest extent possible, existing terms. The most prominent vocabularies in LD include:

- Dublin Core, which describes documents. [Nilsson et al., 2008]
- Friend of a Friend, which describes individuals and their friends. [Brickley and Miller, 2005]
- Semantically Interlinked Online Communities, which describes user-generated content, such as forum posts. [Breslin et al., 2005]
- An RDF binding of Creative Commons, which is a model for licensing the distribution and reuse of works. [Abelson et al., 2008]

In general, Linked Data is intended to be data-oriented: the vocabularies used to structure the data are relatively simple in of themselves, and the focus is on providing large amounts of data in a highly accessible manner. This notion of a highly-interlinked, data-oriented web of services would appear to offer significant opportunities for the context-informed approach, first as an application domain for the technology, and second in supporting some particular aspects in creating better Linked Data infrastructures.

There is already a very large body of research into the use cases for the Semantic Web[2]. Linked Data provides a rich source of external contextual knowledge which could be used as a basis for using contextual enrichment of adaptive applications to solve challenges in these use cases. There are large Linked Data repositories in the areas of general encyclopædic knowledge (DBPedia [Auer et al., 2007], YAGO [Suchanek et al., 2007]), Movies (linkedMDB [Hassanzadeh and Consens, 2009]), and academic publications (DBLP RKB Explorer [Glaser and Millard, 2007]). These repositories are themselves highly interconnected to each other, and to other Linked

[2]For example, see the W3C list at `http://www.w3.org/2001/sw/sweo/public/UseCases/`

Data repositories, and the objective of using the ACP is connect this linked, public, open data with local, specific and possibly private ontological data.

Context-informed systems can take advantage of a blend of the relevant linked open data, as well as local private knowledge and other sources of context to create a semantically-rich knowledge representation for the user's target application. For example, in an adaptive learning application, a linked-data equipped, context-informed eLearning application might be able to find rich descriptions of subject concepts from DBPedia, as well as suggesting relevant additional reading material from DBLP. Adaptive schema reasoning on the part of the target application might even make this possible with a low overhead on the part of the adaptive application developer.

Another area of interest for using linked data with the ACP would be in supporting federated linked data queries. It is common to use the SPARQL query language [Prud'Hommeaux et al., 2006] to access the linked data information, normally in one particular repository per query. The notion of federated SPARQL has been advanced as a means for combining information from diverse repositories and offering it as a query endpoint [Prud'hommeaux, 2007]. Some distributed query engines exist already [Quilitz and Leser, 2008]. There are some interesting applications for the ACP in this area. The first is as a means for serving federated queries. In this case, the target application for the ACP might be an ontology representing the resulting federated concept space. The alignments in the shared semantic view provide integration pathways between the different linked data repositories.

By representing the integration pathways in an explicit fashion, and by providing dynamic reasoner support, it might also be possible to explore the notion of federated write-backs to linked data. At the moment, linked data is principally a read-access model, with limited support for updating data through the UPDATE SPARQL command. The ACP architecture might provide a means for writing back the result of federated reasoning to the sources. This would be an interesting inversion of the current ACP model: in this case one integrated source would feed a number of separate targets.

These notions combine to suggest the creation of a query-oriented version of the ACP, where direct schema loading and reasoner access are replaced, at least in part, by on-demand SPARQL query generation. This has applications in both example scenarios. In implementation terms, the ACP might load a representation of the vocabulary of each linked data source into the shared semantic view, but instead of querying the ontology locally, the schema manager would translate the information need into a SPARQL query.

For serving federated linked data, the ACP could be as a loosely-coupled method for saving federation descriptions. Each federation could be saved as a different Topic Map, created and served on demand. Currently, the performance of the ACP is not appropriate for this application, but with a more performance-focused implementation, and with a reduced Topic Map feature set, it might be possible to achieve satisfactory performance.

SPARQL commands bear a resemblance to the SQL queries used in databases, and for creating federations it might be useful to consider two new commands, CREATE, and ALTER. The CREATE command could be used in SPARQL to specify new vocabularies, while ALTER could be used to add or modify federated schema concepts in the linked data repository. The current architecture of many Linked Data repositories would not support this directly, as they are often based on database export methods [Bizer and Cyganiak, 2006]. However, the ACP supports the notion that both the schema and data of ontologies are alterable, and might provide a basis for investigating the effectiveness of this model of a read/write data web. Federated write-back has typically been a major challenge, so it is likely that this approach would only work in certain specific cases, but it could nonetheless prove useful for solving some problems.

7.3.4 Further Implementation and Experimentation

The system presented in this thesis was evaluated on a case-study basis, based on key properties drawn from both context and ontology mediation requirements. One area of future work would be in designing and deploying the ACP with real services, in order to examine the practicalities of the context-informed approach.

The key considerations for the creation of such a system are:

1. **Ontological Quality:** there is a requirement for the sources of context to expose their knowledge in a useful way as ontological knowledge. The system can gather information across web service protocols, and can load updates on demand. However, the nature of the web protocol makes call-back difficult. In addition, there will need to be a balance for the developer between the cost of updating ontologies and the likely information gain. It might be preferable for a source of context to be chosen which exposes location names for a user, rather than rapidly-moving co-ordinates.

2. **Knowledge Structure of the Target Application:** designing applications which can take full advantage of new knowledge from the context mediator requires the application to have an open knowledge model. Adaptive applications represent one popular approach to solving open-knowledge problems, and an effective method for creating strategies for behaviour which can react to new knowledge. For example, where an adaptive application operates by comparing user model properties to content model elements, the addition of new entities, such as language, to both models demonstrates a clear information gain.

Finally, such systems will likely require better tool support for developing effective agreement between parties. This can be achieved in part through the use of general tools for ontology mapping, but a more user-friendly interface would also be a useful piece of future work.

Bibliography

[Abelson et al., 2008] Abelson, H., Adida, B., Linksvayer, M., and Yergler, N. (2008). ccREL: The Creative Commons Rights Expression Language. *Creative Commons Wiki.*

[Abowd et al., 1999] Abowd, G., Dey, A., Brown, P., Davies, N., Smith, M., and Steggles, P. (1999). Towards a Better Understanding of Context and Context-Awareness. *LECTURE NOTES IN COMPUTER SCIENCE*, pages 304–307.

[Ahmed, 2004] Ahmed, K. (2004). Tm4j developer's guide. Technical report, The TM4J Project.

[Albertoni and De Martino, 2008] Albertoni, R. and De Martino, M. (2008). Asymmetric and context-dependent semantic similarity among ontology instances. *Journal on Data Semantics X.*

[Auer et al., 2007] Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., and Ives, Z. (2007). Dbpedia: A nucleus for a web of open data. *Lecture Notes in Computer Science*, 4825:722.

[Aumueller et al., 2005] Aumueller, D., Do, H., Massmann, S., and Rahm, E. (2005). Schema and ontology matching with COMA++. *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, pages 906–908.

[Baldauf et al., 2007] Baldauf, M., Dustdar, S., and Rosenberg, F. (2007). A survey on context-aware systems. *International Journal of Ad Hoc and Ubiquitous Computing*, 2(4):263–277.

[Bardram et al., 2003] Bardram, J., Kjær, R., and Pedersen, M. (2003). Context-aware user authentication-supporting proximity-based login in pervasive computing. *Lecture Notes in Computer Science*, pages 107–123.

[Bechhofer, 2003] Bechhofer, S. (2003). Owl reasoning examples. Technical report, University of Manchester.

[Bechhofer et al., 2004] Bechhofer, S., van Harmelen, F., Jim Hendler, J., Horrocks, I., L. McGuinness, D. L., Patel-Schneider, P. F., and Andrea Stein, L. (2004). Owl web ontology language reference. Technical report, W3C.

[Bizer and Cyganiak, 2006] Bizer, C. and Cyganiak, R. (2006). D2R server–publishing relational databases on the semantic web. In *5th International Semantic Web Conference*.

[Bizer et al., 2007] Bizer, C., Cyganiak, R., and Heath, T. (2007). How to publish linked data on the web. *Retrieved June*, 20:2008.

[Bolchini et al., 2007] Bolchini, C., Curino, C. A., Quintarelli, E., Schreiber, F. A., and Tanca, L. (2007). A data-oriented survey of context models. *SIGMOD Rec.*, 36(4):19–26.

[Bouquet et al., 2003] Bouquet, P., Giunchiglia, F., van Harmelen, F., Serafini, L., and Stuckenschmidt, H. (2003). C-owl: Contextualising ontologies. In *Proceedings of the 2nd International Semantic Web Conference*.

[Breslin et al., 2005] Breslin, J., Harth, A., Bojars, U., and Decker, S. (2005). Towards semantically-interlinked online communities. In *The 2nd European Semantic Web Conference (ESWC'05), Heraklion, Greece, Proceedings, LNCS*, volume 3532, pages 500–514. Springer.

[Brickley and Miller, 2005] Brickley, D. and Miller, L. (2005). FOAF vocabulary specification. *Namespace Document*, 3.

[Brønstead et al., 2007] Brønstead, J., Hansen, K., and Ingstrup, M. (2007). A survey of service composition mechanisms in ubiquitous computing. In *Second*

Workshop on Requirements and Solutions for Pervasive Software Infrastructures, UbiComp 2007 Workshops Proceedings.

[Brusilovsky, 2001] Brusilovsky, P. (2001). Adaptive hypermedia. *User modeling and user-adapted interaction*, 11(1):87–110.

[Carroll et al., 2004] Carroll, J., Dickinson, I., Dollin, C., Reynolds, D., Seaborne, A., and Wilkinson, K. (2004). Jena: implementing the semantic web recommendations. In *Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*, pages 74–83. ACM New York, NY, USA.

[Ceri et al., 2007] Ceri, S., Daniel, F., Matera, M., and Facca, F. (2007). Model-driven development of context-aware Web applications. *ACM Transactions on Internet Technology (TOIT)*, 7(1).

[Chaari et al., 2007] Chaari, T., Ejigu, D., Laforest, F., and Scuturici, V.-M. (2007). A comprehensive approach to model and use context for adapting applications in pervasive environments. *Journal of Systems and Software.*

[Chen and Kotz, 2000] Chen, G. and Kotz, D. (2000). A Survey of Context-Aware Mobile Computing Research.

[Chen et al., 2004a] Chen, H., Finin, T., and Joshi, A. (2004a). A Context Broker for Building Smart Meeting Rooms. Technical Report DTIC Research Report ADA439472, Defense Technical Information Center.

[Chen et al., 2004b] Chen, H., Perich, F., Chakraborty, D., Finin, T., and Joshi, A. (2004b). Intelligent agents meet semantic web in a smart meeting room. In *AAMAS '04: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 854–861, Washington, DC, USA. IEEE Computer Society.

[Chen et al., 2004c] Chen, H., Perich, F., Finin, T., and Joshi, A. (2004c). SOUPA: standard ontology for ubiquitous and pervasive applications. *Mobile and Ubiquitous Systems: Networking and Services, 2004. MOBIQUITOUS 2004. The First Annual International Conference on*, pages 258–267.

[Choi et al., 2006] Choi, N., Song, I.-Y., and Han, H. (2006). A survey on ontology mapping. *ACM SIGMOD Record*, 35(3).

[Clear et al., 2006] Clear, A., Knox, S., Ye, J., Coyle, L., Dobson, S., and Nixon, P. (2006). Integrating multiple contexts and ontologies in a pervasive computing framework. *Contexts and Ontologies: Theory, Practice and Applications, Riva Del Garda, Italy*, pages 20–25.

[Conlan and Wade, 2004] Conlan, O. and Wade, V. (2004). Evaluation of APeLS-an adaptive eLearning service based on the multi-model, metadata-driven approach. *Lecture notes in computer science*, pages 291–295.

[Conlan et al., 2002] Conlan, O., Wade, V., Bruen, C., and Gargan, M. (2002). Multi-model, metadata driven approach to adaptive hypermedia services for personalized elearning. *Lecture Notes in Computer Science*, pages 100–111.

[Coutaz and Rey, 2002] Coutaz, J. and Rey, G. (2002). Foundations for a Theory of Contextors. *Computer Aided Design of User Interfaces, Springer Verlag, June.*

[Cristani and Cuel, 2005] Cristani, M. and Cuel, R. (2005). A survey on ontology creation methodologies. *International Journal on Semantic Web & Information Systems*, 1(2):49–69.

[Da and Zhang, 2004] Da, T. and Zhang, Q. (2004). A middleware for building context-aware mobile services. In *Vehicular Technology Conference, 2004. VTC 2004-Spring. 2004 IEEE 59th*, volume 5.

[de Graauw, 2002] de Graauw, M. (2002). Business maps: Topic maps go b2b! *O'Reilly XML.com.*

[Dey, 2001] Dey, A. (2001). Understanding and Using Context. *Personal and Ubiquitous Computing*, 5(1):4–7.

[Dobson et al., 2007] Dobson, S., Nixon, P., Coyle, L., Neely, S., Stevenson, G., and Williamson, G. (2007). Construct: An Open Source Pervasive Systems Platform.

In *Consumer Communications and Networking Conference, 2007. CCNC 2007. 2007 4th IEEE*, pages 1203–1204.

[Economist, 2003] Economist, T. (2003). The sentient office is coming. *The Economist.*

[Engmann and Massmann, 2007] Engmann, D. and Massmann, S. (2007). Instance matching with coma++. In *BTW 2007 Workshop: Model Management und Metadaten-Verwaltung.*

[Euzenat et al., 2007] Euzenat, J., Isaac, A., Meilicke, C., Shvaiko, P., Stuckenschmidt, H., Švàb, O., Svàtek, V., van Hage, W. R., and Yatskevich, M. (2007). Results of the ontology alignment evaluation initiative 2007. In *Proceedings of ISWC+ASWC Workshop on Ontology Matching.*

[Euzenat and Schvaiko, 2007] Euzenat, J. and Schvaiko, P. (2007). *Ontology Matching.* Springer.

[Feier et al., 2005] Feier, C., Roman, D., Polleres, A., Domingue, J., Stollberg, M., and Fensel, D. (2005). Towards intelligent web services: The web service modeling ontology (WSMO). *International Conference on Intelligent Computing (ICIC).*

[Glaser and Millard, 2007] Glaser, H. and Millard, I. (2007). Rkb explorer: Application and infrastructure. *Proceedings of Semantic Web Challenge.*

[Grau et al., 2008] Grau, B. C., Horrocks, I., Motik, B., Parsia, B., Patel-Schneider, P., and Sattler, U. (2008). Owl 2: The next step for owl. *Web Semantics: Science, Services and Agents on the World Wide Web*, 6(4):309 – 322. Semantic Web Challenge 2006/2007.

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

[Gu et al., 2005] Gu, T., Pung, H. K., and Zhang, D. Q. (2005). A service-oriented middleware for building context-aware services. *Journal of Network and Computer Applications.*

[Guha, 1991] Guha, R. (1991). *Contexts: A Formalization and Some Applications.* University Microfilms.

[Guttman, 2001] Guttman, E. (2001). Autoconfiguration for IP Networking: Enabling Local Communication. *IEEE INTERNET COMPUTING*, pages 81–86.

[Hassanzadeh and Consens, 2009] Hassanzadeh, O. and Consens, M. P. (2009). Linked movie data base. In *Proceedings of the WWW2009 workshop on Linked Data on the Web (LDOW2009).*

[Heuer and Schmidt, 2008] Heuer, L. and Schmidt, J. (2008). Tmapi 2.0. In *TMRA 2008, Fourth International Conference on Topic Maps Research and Applications.*

[Hofer et al., 2003] Hofer, T., Schwinger, W., Pichler, M., Leonhartsberger, G., Altmann, J., and Retschitzegger, W. (2003). Context-awareness on mobile devices-the hydrogen approach. In *System Sciences, 2003. Proceedings of the 36th Annual Hawaii International Conference on*, page 10.

[Horrocks et al., 2004] Horrocks, I., Patel-Schneider, P. F., Boley, H., Tabet, S., Grosof, B., and Dean, M. (2004). Swrl: A semantic web rule language combining owl and ruleml. Member submission, W3C.

[Hughes and Ashpole, 2004] Hughes, T. C. and Ashpole, B. C. (2004). The semantics of ontology alignment. In *I3CON. Information Interpretation and Integration Conference.*

[IMS Global Learning Consortium, 2005] IMS Global Learning Consortium, I. (2005). Learner information package. `http://www.imsglobal.org/profiles/`.

[Jian et al., 2005] Jian, N., Hu, W., Cheng, G., and Qu, Y. (2005). Falcon-ao: Aligning ontologies with falcon. *log*, 1:2.

[JTC1:SC34, 2002] JTC1:SC34 (2002). ISO/IEC 13250 topic maps. Technical report, ISO/IEC.

[Judd and Steenkiste, 2003] Judd, G. and Steenkiste, P. (2003). Providing contextual information to pervasive computing applications. In *Pervasive Computing* *and Communications, 2003.(PerCom 2003). Proceedings of the First IEEE International Conference on*, pages 133–142.

[Kalfoglou and Schorlemmer, 2003] Kalfoglou, Y. and Schorlemmer, M. (2003). Ontology mapping: the state of the art. *The Knowledge Engineering Review*, 18(1):1–31.

[Kalyanpur et al., 2006] Kalyanpur, A., Parsia, B., Sirin, E., Grau, B., and Hendler, J. (2006). Swoop: A web ontology editing browser. *Web Semantics: Science, Services and Agents on the World Wide Web*, 4(2):144–153.

[Kruschwitz and Al-Bakour, 2005] Kruschwitz, U. and Al-Bakour, H. (2005). Users Want More Sophisticated Search Assistants: Results of a Task-Based Evaluation. *JOURNAL-AMERICAN SOCIETY FOR INFORMATION SCIENCE AND TECHNOLOGY*, 56(13):1377.

[Lewis et al., 2006] Lewis, D., Keeney, J., O Sullivan, D., and Guo, S. (2006). Towards a managed extensible control plane for knowledge-based networking. *Lecture Notes in Computer Science*, 4269:98.

[Lonsdale et al., 2004] Lonsdale, P., Baber, C., and Sharples, M. (2004). A context awareness architecture for facilitating mobile learning. *Learning with Mobile Devices: Research and Development.*

[Mark, 2002] Mark, G. (2002). Extreme collaboration. *Communications of the ACM*, 45(6):89–93.

[Martin, 2004] Martin, D. (2004). *OWL-S: Semantic Markup for Web Services.* W3C, `http://www.w3.org/Submission/2004/SUBM-OWL-S-20041122/`.

[McGuinness and van Harmelen, 2002] McGuinness, D. L. and van Harmelen, F. (2002). Web ontology language (owl lite, owl dl, and owl full) feature synopsis version 1.0. Technical report, W3C.

[Mitra et al., 2005] Mitra, P., Wiederhold, G., and Decker, S. (2005). Dealing with semantic interoperation of data. *Local to Global Data Interoperability - Challenges and Technologies, 2005*, pages 134–144.

[Motik et al., 2009] Motik, B., Cuenca Grau, B., Horrocks, I., Wu, Z., Fokoue, A., and Lutz, C. (2009). Owl2 profiles. `http://www.w3.org/2007/OWL/wiki/Profiles#OWL_2_EL`.

[Mrissa et al., 2008] Mrissa, M., Thiran, P., Ghedira, C., Benslimane, D., and Maamar, Z. (2008). Using context to enable semantic mediation in web service communities. In *Proceedings of the 2008 international workshop on Context enabled source and service selection, integration and adaptation: organized with the 17th International World Wide Web Conference (WWW 2008)*. ACM New York, NY, USA.

[Niles and Pease, 2001] Niles, I. and Pease, A. (2001). Towards a standard upper ontology. In *FOIS '01: Proceedings of the international conference on Formal Ontology in Information Systems*, pages 2–9, New York, NY, USA. ACM.

[Niles and Pease, 2003] Niles, I. and Pease, A. (2003). Linking Lexicons and Ontologies: Mapping WordNet to the Suggested Upper Merged Ontology. *Proceedings of the IEEE International Conference on Information and Knowledge Engineering*, pages 412–416.

[Niles and Pease, 2004] Niles, I. and Pease, A. (2004). Sumo to owl rdf mapping. `http://www.ontologyportal.org/translations/SUMO.owl.txt`.

[Nilsson et al., 2008] Nilsson, M., Powell, A., Johnston, P., and Naeve, A. (2008). Expressing Dublin Core metadata using the Resource Description Framework (RDF). *DCMI Recommendation.*

[Noy et al., 2008] Noy, N. F., de Coronado, S., Solbrig, H., Fragoso, G., Hartel, F. W., and Musen, M. A. (2008). Representing the nci thesaurus in owl dl: Modeling tools help modeling languages. *Applied Ontology*, 3(3):173–190.

[OASIS, 2004] OASIS (2004). Oasis uddi specifications version 3. Technical report, OASIS, `http://uddi.org/pubs/uddi_v3.htm`.

[O'Connor, 2005] O'Connor, A. (2005). Mechanisms for context-informed adaptive hypermedia. Master's thesis, University of Dublin, Trinity College. TCD CS Technical Report; TCD-CS-2005-15.

[Oh and Woo, 2005] Oh, Y. and Woo, W. (2005). A Unified Application Service Model for ubiHome by Exploiting Intelligent Context-Awareness. *LECTURE NOTES IN COMPUTER SCIENCE*, 3598:192.

[O'Sullivan, 2006] O'Sullivan, D. (2006). *The OISIN Framework: Ontology Interoperation in Support of Semantic Interoperation.* PhD thesis, Department of Computer Science, University of Dublin, Trinity College.

[Pepper, 2000] Pepper, S. (2000). The TAO of Topic Maps. *Proceedings of XML Europe.*

[Portland Pattern Repository, 2009] Portland Pattern Repository (2009). Facade pattern. *Portland Pattern Repository.*

[Pradhan, 2000] Pradhan, S. (2000). Semantic location. *Personal Ubiquitous Comput.*, 4(4):213–216.

[Prud'hommeaux, 2007] Prud'hommeaux, E. (2007). Federated SPARQL.

[Prud'Hommeaux et al., 2006] Prud'Hommeaux, E., Seaborne, A., et al. (2006). SPARQL query language for RDF. *W3C working draft*, 4.

[Qian, 1993] Qian, X. (1993). Semantic interoperation via intelligent mediation. *Research Issues in Data Engineering, 1993: Interoperability in Multidatabase Systems, 1993. Proceedings RIDE-IMS '93., Third International Workshop on*, pages 228–231.

[Quilitz and Leser, 2008] Quilitz, B. and Leser, U. (2008). Querying distributed rdf data sources with sparql. *Lecture Notes in Computer Science*, 5021:524.

[Ratto et al., 2003] Ratto, M., Shapiro, R., Truong, T., and Griswold, W. (2003). The activeclass project: Experiments in encouraging classroom participation. In *Computer support for collaborative learning*, pages 477–486. Citeseer.

[Redhat Middleware, 2004] Redhat Middleware, L. (2004). Hibernate - relational persistence for idiomatic java. Technical report, Hibernate.

[Reed and Lenat, 2002] Reed, S. and Lenat, D. (2002). Mapping ontologies into cyc. In *Proc. AAAI Conference 2002 Workshop on Ontologies for the Semantic Web.*

[Roman et al., 2005] Roman, D., Keller, U., Lausen, H., de Bruijn, J., Lara, R., Stollberg, M., Polleres, A., Feier, C., Bussler, C., and Fensel, D. (2005). Web service modeling ontology. *Applied Ontology*, 1(1):77–106.

[Russell and Norvig, 1995] Russell, S. J. and Norvig, P. (1995). *Artificial intelligence: a modern approach.* Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

[Schilit et al., 1994] Schilit, B., Adams, N., and Want, R. (1994). Context-aware computing applications. In *Mobile Computing Systems and Applications, 1994. Proceedings., Workshop on*, pages 85–90.

[Schmidt et al., 1999] Schmidt, A., Beigl, M., and Gellersen, H. (1999). There is more to context than location. *Computers & Graphics*, 23(6):893–901.

[Serafini and Tamilin, 2005] Serafini, L. and Tamilin, A. (2005). Drago: Distributed reasoning architecture for the semantic web. *Proc. of the Second European Semantic Web Conference (ESWC'05)*, pages 361–376.

[Serafini and Tamilin, 2007] Serafini, L. and Tamilin, A. (2007). Instance migration in heterogeneous ontology environments. *LECTURE NOTES IN COMPUTER SCIENCE*, 4825:452.

[Shadbolt et al., 2006] Shadbolt, N., Hall, W., and Bernes-Lee, T. (2006). The semantic web revisited. *IEEE Intelligent Systems*, 21(2):96–101.

[Sowa, 2000] Sowa, J. F. (2000). *Knowledge representation: logical, philosophical and computational foundations.* Brooks/Cole Publishing Co., Pacific Grove, CA, USA.

[Spyns et al., 2002] Spyns, P., Meersman, R., and Jarrar, M. (2002). Data modelling versus ontology engineering. *SIGMOD Rec.*, 31(4):12–17.

[Steggles and Gschwind, 2005] Steggles, P. and Gschwind, S. (2005). The Ubisense smart space platform. In *Adjunct Proceedings of the Third International Conference on Pervasive Computing*, volume 191.

[Strang and Linnhoff-popien, 2004] Strang, T. and Linnhoff-popien, C. (2004). A context modeling survey. In *Workshop on Advanced Context Modelling, Reasoning and Management, UbiComp 2004 - The Sixth International Conference on Ubiquitous Computing, Nottingham/England*.

[Strassner et al., 2008] Strassner, J., Liu, Y., Jiang, M., Zhang, J., van der Meer, S., Foghlu, M., Fahy, C., and Donnelly, W. (2008). Modelling context for autonomic networking. In *Network Operations and Management Symposium Workshops, 2008. NOMS Workshops 2008. IEEE*.

[Suchanek et al., 2007] Suchanek, F., Kasneci, G., and Weikum, G. (2007). Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web*, pages 697–706. ACM New York, NY, USA.

[Sun Microsystems, Inc., 2004] Sun Microsystems, Inc. (2004). Jsr-000176 j2se 5.0 (tiger) release contents. Technical report, Sun Microsystems, Inc.

[Thomson et al., 2003] Thomson, G., Richmond, M., Terzis, S., and Nixon, P. (2003). An approach to dynamic context discovery and composition. *Proceedings of UbiSys*, 3.

[Wang et al., 2002] Wang, X., Dong, J., Chin, C., Hettiarachchi, S., and Zhang, D. (2002). Semantic Space: an infrastructure for smart spaces. *Computing*, 1(2):67–74.

[Ye et al., 2007] Ye, J., Coyle, L., Dobson, S., and Nixon, P. (2007). Ontology-based models in pervasive computing systems. *The Knowledge Engineering Review*, 22(04):315–347.

[Zhu et al., 2005] Zhu, F., Mutka, M., and Ni, L. (2005). Service Discovery in Pervasive Computing Environments. *IEEE PERVASIVE COMPUTING*, pages 81–90.

Appendix A

Ontology Information

A.1 Introduction

This chapter includes information about the ontologies used to in the evaluation of the work presented in this thesis.

A.2 User Model Case Study

A.2.1 LIP Description

The IMS LIP Information Model Core Data defines the following categories[1]:

- Identification: Biographic and demographic data relevant to learning;
- Goal: Learning, career and other objectives and aspirations;
- Qualifications, Certifications and Licenses (qcl): Qualifications, certifications and licenses granted by recognized authorities;
- Activity: Any learning-related activity in any state of completion. Could be

[1]These definitions are reproduced from the Specification at `http://www.imsglobal.org/profiles/lipinfo01.html`

self-reported. Includes formal and informal education, training, work experience, and military or civic service;

- Transcript: A record that is used to provide an institutionally-based summary of academic achievement. The structure of this record can take many forms;
- Interest: Information describing hobbies and recreational activities;
- Competency: Skills, knowledge, and abilities acquired in the cognitive, affective, and/or psychomotor domains;
- Affiliation: Membership of professional organizations, etc. Membership of groups is covered by the IMS Enterprise specification;
- Accessibility: General accessibility to the learner information as defined through language capabilities, disabilities, eligibilities and learning preferences including cognitive preferences (e.g. issues of learning style), physical preferences (e.g. a preference for large print), and technological preferences (e.g. a preference for a particular computer platform);
- Securitykey: The set of passwords and security keys assigned to the learner for transactions with learner information systems and services.
- Relationship: The set of relationships between the core components. The core structures do not have within them identifiers that link to the core structures. Instead all of these relationships are captured in a single core structure thereby making the links simpler to identify and manage.

The LIP standard permits the inclusion of the core data models, as well as information from within those core models. For the purposes of the ontologies designed below, the accessibility model was not included, and the remaining properties were included based on finding a representative set of properties from the examples given with the standard[2].

[2]`http://www.imsglobal.org/profiles/`

A.2.2 Ontology XML files

These ontologies were created using the SWOOP ontology tool [Kalyanpur et al., 2006].

A.2.3 Properties-Oriented Version of the AE Ontology

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE rdf:RDF [
  <!ENTITY AEProp "http://kdeg.cs.tcd.ie/ontologies/AEProp#">
  <!ENTITY owl "http://www.w3.org/2002/07/owl#">
  <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#">
  <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#">
  <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#">
]>
<rdf:RDF xml:base="http://kdeg.cs.tcd.ie/ontologies/AEProp"
         xmlns:AEProp="&AEProp;"
         xmlns:owl="&owl;"
         xmlns:rdf="&rdf;"
         xmlns:rdfs="&rdfs;">
<!-- Ontology Information -->
  <owl:Ontology rdf:about=""
                rdfs:label="AEProp"
                owl:versionInfo="0.1"/>
<!-- Classes -->
  <owl:Class rdf:about="#Learner"
             rdfs:label="Learner"/>
<!-- Annotation Properties -->
  <owl:AnnotationProperty rdf:about="&rdfs;label"/>
  <owl:AnnotationProperty rdf:about="&owl;versionInfo"/>
<!-- Datatype Properties -->
  <owl:DatatypeProperty rdf:about="#adaptivity"
                        rdfs:label="adaptivity"/>
  <owl:DatatypeProperty rdf:about="#candidate"
                        rdfs:label="candidate"/>
  <owl:DatatypeProperty rdf:about="#competency"
                        rdfs:label="competency">
    <rdfs:subPropertyOf rdf:resource="#adaptivity"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:about="#identifier"
                        rdfs:label="identifier"/>
  <owl:DatatypeProperty rdf:about="#name"
                        rdfs:label="name"/>
<!-- Instances -->
```

```
  <AEProp:Learner rdf:about="#user1"
                  rdfs:label="user1">
    <AEProp:identifier rdf:datatype="&xsd;string">user1</AEProp:identifier>
  </AEProp:Learner>
  <AEProp:Learner rdf:about="#user10"
                  rdfs:label="user10">
    <AEProp:identifier rdf:datatype="&xsd;string">user10</AEProp:identifier>
  </AEProp:Learner>
  <AEProp:Learner rdf:about="#user2"
                  rdfs:label="user2">
    <AEProp:identifier rdf:datatype="&xsd;string">user2</AEProp:identifier>
  </AEProp:Learner>
  <AEProp:Learner rdf:about="#user3"
                  rdfs:label="user3">
    <AEProp:identifier rdf:datatype="&xsd;string">user3</AEProp:identifier>
  </AEProp:Learner>
  <AEProp:Learner rdf:about="#user4"
                  rdfs:label="user4">
    <AEProp:identifier rdf:datatype="&xsd;string">user4</AEProp:identifier>
  </AEProp:Learner>
  <AEProp:Learner rdf:about="#user5"
                  rdfs:label="user5">
    <AEProp:identifier rdf:datatype="&xsd;string">user5</AEProp:identifier>
  </AEProp:Learner>
  <AEProp:Learner rdf:about="#user6"
                  rdfs:label="user6">
    <AEProp:identifier rdf:datatype="&xsd;string">user6</AEProp:identifier>
  </AEProp:Learner>
  <AEProp:Learner rdf:about="#user7"
                  rdfs:label="user7">
    <AEProp:identifier rdf:datatype="&xsd;string">user7</AEProp:identifier>
  </AEProp:Learner>
  <AEProp:Learner rdf:about="#user8"
                  rdfs:label="user8">
    <AEProp:identifier rdf:datatype="&xsd;string">user8</AEProp:identifier>
  </AEProp:Learner>
  <AEProp:Learner rdf:about="#user9"
                  rdfs:label="user9">
    <AEProp:identifier rdf:datatype="&xsd;string">user9</AEProp:identifier>
  </AEProp:Learner>
</rdf:RDF>
```

A.2.4 Class-Oriented Version of the AE Ontology

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE rdf:RDF [
  <!ENTITY AEClass "http://kdeg.cs.tcd.ie/ontologies/AEClass">
  <!ENTITY owl "http://www.w3.org/2002/07/owl#">
  <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#">
  <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#">
  <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#">
]>
<rdf:RDF xml:base="&AEClass;"
         xmlns:owl="&owl;"
         xmlns:rdf="&rdf;"
         xmlns:rdfs="&rdfs;">

<!-- Ontology Information -->
  <owl:Ontology rdf:about=""
                owl:versionInfo="0.1"/>

<!-- Classes -->
  <owl:Class rdf:about="#Adaptivity"
             rdfs:label="Adaptivity"/>
  <owl:Class rdf:about="#Candidate"
             rdfs:label="Candidate"/>
  <owl:Class rdf:about="#Competency"
             rdfs:label="Competency">
    <rdfs:subClassOf rdf:resource="#Adaptivity"/>
  </owl:Class>

  <owl:Class rdf:about="#Learner"
             rdfs:label="Learner"/>

<!-- Annotation Properties -->
  <owl:AnnotationProperty rdf:about="&rdfs;label"/>
  <owl:AnnotationProperty rdf:about="&owl;versionInfo"/>

<!-- Datatype Properties -->
  <owl:DatatypeProperty rdf:about="#candidateValue"
                        rdfs:label="candidateValue"/>
  <owl:DatatypeProperty rdf:about="#competencyValue"
                        rdfs:label="competencyValue"/>
  <owl:DatatypeProperty rdf:about="#identifier"
                        rdfs:label="identifier"/>
  <owl:DatatypeProperty rdf:about="#name"
                        rdfs:label="name"/>
<!-- Object Properties -->
  <owl:ObjectProperty rdf:about="#hasAdaptivity"
```

```
                     rdfs:label="hasAdaptivity"/>
  <owl:ObjectProperty rdf:about="#hasCandidate"
                     rdfs:label="hasCandidate"/>
</rdf:RDF>
```

A.2.5 Class-Oriented Version of the LIP Ontology

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE rdf:RDF [
  <!ENTITY LIPClass "http://kdeg.cs.tcd.ie/ontologies/LIPClass#">
  <!ENTITY owl "http://www.w3.org/2002/07/owl#">
  <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#">
  <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#">
  <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#">
]>
<rdf:RDF xml:base="http://kdeg.cs.tcd.ie/ontologies/LIPClass"
         xmlns:LIPClass="&LIPClass;"
         xmlns:owl="&owl;"
         xmlns:rdf="&rdf;"
         xmlns:rdfs="&rdfs;">

<!-- Ontology Information -->
  <owl:Ontology rdf:about=""
                rdfs:label="LIPCLass"
                owl:versionInfo="0.1"/>

<!-- Classes -->
  <owl:Class rdf:about="#Identification"
             rdfs:label="Identification"/>
  <owl:Class rdf:about="#LIP"
             rdfs:label="LIP"/>
  <owl:Class rdf:about="#LearnerInformation"
             rdfs:label="LearnerInformation"/>

<!-- Annotation Properties -->
  <owl:AnnotationProperty rdf:about="&rdfs;label"/>
  <owl:AnnotationProperty rdf:about="&owl;versionInfo"/>

<!-- Datatype Properties -->
  <owl:DatatypeProperty rdf:about="#accessibility"
                        rdfs:label="accessibility"/>
  <owl:DatatypeProperty rdf:about="#activity"
                        rdfs:label="activity"/>
  <owl:DatatypeProperty rdf:about="#address"
                        rdfs:label="address"/>
  <owl:DatatypeProperty rdf:about="#affiliation"
                        rdfs:label="affiliation"/>
  <owl:DatatypeProperty rdf:about="#competency"
                        rdfs:label="competency"/>
  <owl:DatatypeProperty rdf:about="#contactinfo"
                        rdfs:label="contactinfo"/>
  <owl:DatatypeProperty rdf:about="#geo"
```

```
                      rdfs:label="geo"/>
  <owl:DatatypeProperty rdf:about="#goal"
                      rdfs:label="goal"/>
  <owl:DatatypeProperty rdf:about="#interest"
                      rdfs:label="interest"/>
  <owl:DatatypeProperty rdf:about="#name"
                      rdfs:label="name"/>
  <owl:DatatypeProperty rdf:about="#qcl"
                      rdfs:label="qcl"/>
  <owl:DatatypeProperty rdf:about="#relationship"
                      rdfs:label="relationship"/>
  <owl:DatatypeProperty rdf:about="#securitykey"
                      rdfs:label="securitykey"/>
  <owl:DatatypeProperty rdf:about="#transcript"
                      rdfs:label="transcript"/>
  <owl:DatatypeProperty rdf:about="#uid"
                      rdfs:label="uid"/>

<!-- Object Properties -->
  <owl:ObjectProperty rdf:about="#hasIdentification"
                    rdfs:label="hasIdentification"/>
  <owl:ObjectProperty rdf:about="#hasLearnerInformation"
                    rdfs:label="hasLearnerInformation"/>

<!-- Instances -->
    <!-- Instance Begins -->
  <LIPClass:Identification rdf:about="#iduser1"
                         rdfs:label="iduser1">
    <LIPClass:name rdf:datatype="&xsd;string">name</LIPClass:name>
    <LIPClass:uid rdf:datatype="&xsd;string">user1</LIPClass:uid>
  </LIPClass:Identification>

  <LIPClass:LearnerInformation rdf:about="#liuser1"
                             rdfs:label="liuser1">
    <LIPClass:competency rdf:datatype="&xsd;string">competency1</LIPClass:competency>
    <LIPClass:competency rdf:datatype="&xsd;string">competency2</LIPClass:competency>
    <LIPClass:competency rdf:datatype="&xsd;string">competency3</LIPClass:competency>
    <LIPClass:competency rdf:datatype="&xsd;string">competency4</LIPClass:competency>
    <LIPClass:competency rdf:datatype="&xsd;string">competency5</LIPClass:competency>
    <LIPClass:interest rdf:datatype="&xsd;string">sql</LIPClass:interest>
    <LIPClass:qcl rdf:datatype="&xsd;string">sql.concept1</LIPClass:qcl>
    <LIPClass:qcl rdf:datatype="&xsd;string">sql.concept2</LIPClass:qcl>
    <LIPClass:qcl rdf:datatype="&xsd;string">sql.concept3</LIPClass:qcl>
    <LIPClass:qcl rdf:datatype="&xsd;string">sql.concept4</LIPClass:qcl>
    <LIPClass:qcl rdf:datatype="&xsd;string">sql.concept5</LIPClass:qcl>
    <LIPClass:transcript rdf:datatype="&xsd;string">relational.concept1</LIPClass:transcript>
    <LIPClass:transcript rdf:datatype="&xsd;string">relational.concept2</LIPClass:transcript>
    <LIPClass:transcript rdf:datatype="&xsd;string">relational.concept3</LIPClass:transcript>
    <LIPClass:transcript rdf:datatype="&xsd;string">relational.concept4</LIPClass:transcript>
    <LIPClass:transcript rdf:datatype="&xsd;string">relational.concept5</LIPClass:transcript>
  </LIPClass:LearnerInformation>

  <LIPClass:LIP rdf:about="#user1"
                rdfs:label="user1">
    <LIPClass:hasIdentification rdf:resource="#iduser1"/>
    <LIPClass:hasLearnerInformation rdf:resource="#liuser1"/>
  </LIPClass:LIP>
      <!-- Instance Begins -->
  <LIPClass:Identification rdf:about="#iduser2"
                           rdfs:label="iduser2">
    <LIPClass:name rdf:datatype="&xsd;string">name</LIPClass:name>
    <LIPClass:uid rdf:datatype="&xsd;string">user2</LIPClass:uid>
  </LIPClass:Identification>

  <LIPClass:LearnerInformation rdf:about="#liuser2"
                               rdfs:label="liuser2">
    <LIPClass:competency rdf:datatype="&xsd;string">competency1</LIPClass:competency>
    <LIPClass:competency rdf:datatype="&xsd;string">competency2</LIPClass:competency>
    <LIPClass:competency rdf:datatype="&xsd;string">competency3</LIPClass:competency>
    <LIPClass:competency rdf:datatype="&xsd;string">competency4</LIPClass:competency>
    <LIPClass:competency rdf:datatype="&xsd;string">competency5</LIPClass:competency>
    <LIPClass:interest rdf:datatype="&xsd;string">sql</LIPClass:interest>
    <LIPClass:qcl rdf:datatype="&xsd;string">sql.concept1</LIPClass:qcl>
    <LIPClass:qcl rdf:datatype="&xsd;string">sql.concept2</LIPClass:qcl>
    <LIPClass:qcl rdf:datatype="&xsd;string">sql.concept3</LIPClass:qcl>
    <LIPClass:qcl rdf:datatype="&xsd;string">sql.concept4</LIPClass:qcl>
    <LIPClass:qcl rdf:datatype="&xsd;string">sql.concept5</LIPClass:qcl>
    <LIPClass:transcript rdf:datatype="&xsd;string">relational.concept1</LIPClass:transcript>
    <LIPClass:transcript rdf:datatype="&xsd;string">relational.concept2</LIPClass:transcript>
    <LIPClass:transcript rdf:datatype="&xsd;string">relational.concept3</LIPClass:transcript>
    <LIPClass:transcript rdf:datatype="&xsd;string">relational.concept4</LIPClass:transcript>
    <LIPClass:transcript rdf:datatype="&xsd;string">relational.concept5</LIPClass:transcript>
  </LIPClass:LearnerInformation>

  <LIPClass:LIP rdf:about="#user2"
                rdfs:label="user2">
    <LIPClass:hasIdentification rdf:resource="#iduser2"/>
    <LIPClass:hasLearnerInformation rdf:resource="#liuser2"/>
  </LIPClass:LIP>
      <!-- Instance Begins -->
  <LIPClass:Identification rdf:about="#iduser3"
                           rdfs:label="iduser3">
    <LIPClass:name rdf:datatype="&xsd;string">name</LIPClass:name>
    <LIPClass:uid rdf:datatype="&xsd;string">user3</LIPClass:uid>
```

```
</LIPClass:Identification>

<LIPClass:LearnerInformation rdf:about="#liuser3"
                             rdfs:label="liuser3">
  <LIPClass:competency rdf:datatype="&xsd;string">competency1</LIPClass:competency>
  <LIPClass:competency rdf:datatype="&xsd;string">competency2</LIPClass:competency>
  <LIPClass:competency rdf:datatype="&xsd;string">competency3</LIPClass:competency>
  <LIPClass:competency rdf:datatype="&xsd;string">competency4</LIPClass:competency>
  <LIPClass:competency rdf:datatype="&xsd;string">competency5</LIPClass:competency>
  <LIPClass:interest rdf:datatype="&xsd;string">sql</LIPClass:interest>
  <LIPClass:qcl rdf:datatype="&xsd;string">sql.concept1</LIPClass:qcl>
  <LIPClass:qcl rdf:datatype="&xsd;string">sql.concept2</LIPClass:qcl>
  <LIPClass:qcl rdf:datatype="&xsd;string">sql.concept3</LIPClass:qcl>
  <LIPClass:qcl rdf:datatype="&xsd;string">sql.concept4</LIPClass:qcl>
  <LIPClass:qcl rdf:datatype="&xsd;string">sql.concept5</LIPClass:qcl>
  <LIPClass:transcript rdf:datatype="&xsd;string">relational.concept1</LIPClass:transcript>
  <LIPClass:transcript rdf:datatype="&xsd;string">relational.concept2</LIPClass:transcript>
  <LIPClass:transcript rdf:datatype="&xsd;string">relational.concept3</LIPClass:transcript>
  <LIPClass:transcript rdf:datatype="&xsd;string">relational.concept4</LIPClass:transcript>
  <LIPClass:transcript rdf:datatype="&xsd;string">relational.concept5</LIPClass:transcript>
</LIPClass:LearnerInformation>

<LIPClass:LIP rdf:about="#user3"
              rdfs:label="user3">
  <LIPClass:hasIdentification rdf:resource="#iduser3"/>
  <LIPClass:hasLearnerInformation rdf:resource="#liuser3"/>
</LIPClass:LIP>
    <!-- Instance Begins -->
<LIPClass:Identification rdf:about="#iduser4"
                         rdfs:label="iduser4">
  <LIPClass:name rdf:datatype="&xsd;string">name</LIPClass:name>
  <LIPClass:uid rdf:datatype="&xsd;string">user4</LIPClass:uid>
</LIPClass:Identification>

<LIPClass:LearnerInformation rdf:about="#liuser4"
                             rdfs:label="liuser4">
  <LIPClass:competency rdf:datatype="&xsd;string">competency1</LIPClass:competency>
  <LIPClass:competency rdf:datatype="&xsd;string">competency2</LIPClass:competency>
  <LIPClass:competency rdf:datatype="&xsd;string">competency3</LIPClass:competency>
  <LIPClass:competency rdf:datatype="&xsd;string">competency4</LIPClass:competency>
  <LIPClass:competency rdf:datatype="&xsd;string">competency5</LIPClass:competency>
  <LIPClass:interest rdf:datatype="&xsd;string">sql</LIPClass:interest>
  <LIPClass:qcl rdf:datatype="&xsd;string">sql.concept1</LIPClass:qcl>
  <LIPClass:qcl rdf:datatype="&xsd;string">sql.concept2</LIPClass:qcl>
  <LIPClass:qcl rdf:datatype="&xsd;string">sql.concept3</LIPClass:qcl>
  <LIPClass:qcl rdf:datatype="&xsd;string">sql.concept4</LIPClass:qcl>
  <LIPClass:qcl rdf:datatype="&xsd;string">sql.concept5</LIPClass:qcl>
  <LIPClass:transcript rdf:datatype="&xsd;string">relational.concept1</LIPClass:transcript>
  <LIPClass:transcript rdf:datatype="&xsd;string">relational.concept2</LIPClass:transcript>
  <LIPClass:transcript rdf:datatype="&xsd;string">relational.concept3</LIPClass:transcript>
  <LIPClass:transcript rdf:datatype="&xsd;string">relational.concept4</LIPClass:transcript>
  <LIPClass:transcript rdf:datatype="&xsd;string">relational.concept5</LIPClass:transcript>
</LIPClass:LearnerInformation>

<LIPClass:LIP rdf:about="#user4"
              rdfs:label="user4">
  <LIPClass:hasIdentification rdf:resource="#iduser4"/>
  <LIPClass:hasLearnerInformation rdf:resource="#liuser4"/>
</LIPClass:LIP>
    <!-- Instance Begins -->
<LIPClass:Identification rdf:about="#iduser5"
                         rdfs:label="iduser5">
  <LIPClass:name rdf:datatype="&xsd;string">name</LIPClass:name>
  <LIPClass:uid rdf:datatype="&xsd;string">user5</LIPClass:uid>
</LIPClass:Identification>

<LIPClass:LearnerInformation rdf:about="#liuser5"
                             rdfs:label="liuser5">
  <LIPClass:competency rdf:datatype="&xsd;string">competency1</LIPClass:competency>
  <LIPClass:competency rdf:datatype="&xsd;string">competency2</LIPClass:competency>
  <LIPClass:competency rdf:datatype="&xsd;string">competency3</LIPClass:competency>
  <LIPClass:competency rdf:datatype="&xsd;string">competency4</LIPClass:competency>
  <LIPClass:competency rdf:datatype="&xsd;string">competency5</LIPClass:competency>
  <LIPClass:interest rdf:datatype="&xsd;string">sql</LIPClass:interest>
  <LIPClass:qcl rdf:datatype="&xsd;string">sql.concept1</LIPClass:qcl>
  <LIPClass:qcl rdf:datatype="&xsd;string">sql.concept2</LIPClass:qcl>
  <LIPClass:qcl rdf:datatype="&xsd;string">sql.concept3</LIPClass:qcl>
  <LIPClass:qcl rdf:datatype="&xsd;string">sql.concept4</LIPClass:qcl>
  <LIPClass:qcl rdf:datatype="&xsd;string">sql.concept5</LIPClass:qcl>
  <LIPClass:transcript rdf:datatype="&xsd;string">relational.concept1</LIPClass:transcript>
  <LIPClass:transcript rdf:datatype="&xsd;string">relational.concept2</LIPClass:transcript>
  <LIPClass:transcript rdf:datatype="&xsd;string">relational.concept3</LIPClass:transcript>
  <LIPClass:transcript rdf:datatype="&xsd;string">relational.concept4</LIPClass:transcript>
  <LIPClass:transcript rdf:datatype="&xsd;string">relational.concept5</LIPClass:transcript>
</LIPClass:LearnerInformation>

<LIPClass:LIP rdf:about="#user5"
              rdfs:label="user5">
  <LIPClass:hasIdentification rdf:resource="#iduser5"/>
  <LIPClass:hasLearnerInformation rdf:resource="#liuser5"/>
</LIPClass:LIP>
    <!-- Instance Begins -->
<LIPClass:Identification rdf:about="#iduser6"
                         rdfs:label="iduser6">
  <LIPClass:name rdf:datatype="&xsd;string">name</LIPClass:name>
  <LIPClass:uid rdf:datatype="&xsd;string">user6</LIPClass:uid>
</LIPClass:Identification>

<LIPClass:LearnerInformation rdf:about="#liuser6"
                             rdfs:label="liuser6">
  <LIPClass:competency rdf:datatype="&xsd;string">competency1</LIPClass:competency>
  <LIPClass:competency rdf:datatype="&xsd;string">competency2</LIPClass:competency>
  <LIPClass:competency rdf:datatype="&xsd;string">competency3</LIPClass:competency>
  <LIPClass:competency rdf:datatype="&xsd;string">competency4</LIPClass:competency>
  <LIPClass:competency rdf:datatype="&xsd;string">competency5</LIPClass:competency>
  <LIPClass:interest rdf:datatype="&xsd;string">sql</LIPClass:interest>
  <LIPClass:qcl rdf:datatype="&xsd;string">sql.concept1</LIPClass:qcl>
  <LIPClass:qcl rdf:datatype="&xsd;string">sql.concept2</LIPClass:qcl>
  <LIPClass:qcl rdf:datatype="&xsd;string">sql.concept3</LIPClass:qcl>
  <LIPClass:qcl rdf:datatype="&xsd;string">sql.concept4</LIPClass:qcl>
  <LIPClass:qcl rdf:datatype="&xsd;string">sql.concept5</LIPClass:qcl>
  <LIPClass:transcript rdf:datatype="&xsd;string">relational.concept1</LIPClass:transcript>
  <LIPClass:transcript rdf:datatype="&xsd;string">relational.concept2</LIPClass:transcript>
  <LIPClass:transcript rdf:datatype="&xsd;string">relational.concept3</LIPClass:transcript>
  <LIPClass:transcript rdf:datatype="&xsd;string">relational.concept4</LIPClass:transcript>
  <LIPClass:transcript rdf:datatype="&xsd;string">relational.concept5</LIPClass:transcript>
</LIPClass:LearnerInformation>

<LIPClass:LIP rdf:about="#user6"
              rdfs:label="user6">
  <LIPClass:hasIdentification rdf:resource="#iduser6"/>
  <LIPClass:hasLearnerInformation rdf:resource="#liuser6"/>
</LIPClass:LIP>
    <!-- Instance Begins -->
<LIPClass:Identification rdf:about="#iduser7"
                         rdfs:label="iduser7">
  <LIPClass:name rdf:datatype="&xsd;string">name</LIPClass:name>
  <LIPClass:uid rdf:datatype="&xsd;string">user7</LIPClass:uid>
</LIPClass:Identification>

<LIPClass:LearnerInformation rdf:about="#liuser7"
                             rdfs:label="liuser7">
  <LIPClass:competency rdf:datatype="&xsd;string">competency1</LIPClass:competency>
  <LIPClass:competency rdf:datatype="&xsd;string">competency2</LIPClass:competency>
  <LIPClass:competency rdf:datatype="&xsd;string">competency3</LIPClass:competency>
  <LIPClass:competency rdf:datatype="&xsd;string">competency4</LIPClass:competency>
  <LIPClass:competency rdf:datatype="&xsd;string">competency5</LIPClass:competency>
  <LIPClass:interest rdf:datatype="&xsd;string">sql</LIPClass:interest>
  <LIPClass:qcl rdf:datatype="&xsd;string">sql.concept1</LIPClass:qcl>
  <LIPClass:qcl rdf:datatype="&xsd;string">sql.concept2</LIPClass:qcl>
  <LIPClass:qcl rdf:datatype="&xsd;string">sql.concept3</LIPClass:qcl>
  <LIPClass:qcl rdf:datatype="&xsd;string">sql.concept4</LIPClass:qcl>
  <LIPClass:qcl rdf:datatype="&xsd;string">sql.concept5</LIPClass:qcl>
  <LIPClass:transcript rdf:datatype="&xsd;string">relational.concept1</LIPClass:transcript>
  <LIPClass:transcript rdf:datatype="&xsd;string">relational.concept2</LIPClass:transcript>
  <LIPClass:transcript rdf:datatype="&xsd;string">relational.concept3</LIPClass:transcript>
  <LIPClass:transcript rdf:datatype="&xsd;string">relational.concept4</LIPClass:transcript>
  <LIPClass:transcript rdf:datatype="&xsd;string">relational.concept5</LIPClass:transcript>
</LIPClass:LearnerInformation>

<LIPClass:LIP rdf:about="#user7"
              rdfs:label="user7">
  <LIPClass:hasIdentification rdf:resource="#iduser7"/>
  <LIPClass:hasLearnerInformation rdf:resource="#liuser7"/>
</LIPClass:LIP>
    <!-- Instance Begins -->
<LIPClass:Identification rdf:about="#iduser8"
                         rdfs:label="iduser8">
  <LIPClass:name rdf:datatype="&xsd;string">name</LIPClass:name>
  <LIPClass:uid rdf:datatype="&xsd;string">user8</LIPClass:uid>
</LIPClass:Identification>

<LIPClass:LearnerInformation rdf:about="#liuser8"
                             rdfs:label="liuser8">
  <LIPClass:competency rdf:datatype="&xsd;string">competency1</LIPClass:competency>
  <LIPClass:competency rdf:datatype="&xsd;string">competency2</LIPClass:competency>
  <LIPClass:competency rdf:datatype="&xsd;string">competency3</LIPClass:competency>
  <LIPClass:competency rdf:datatype="&xsd;string">competency4</LIPClass:competency>
  <LIPClass:competency rdf:datatype="&xsd;string">competency5</LIPClass:competency>
  <LIPClass:interest rdf:datatype="&xsd;string">sql</LIPClass:interest>
  <LIPClass:qcl rdf:datatype="&xsd;string">sql.concept1</LIPClass:qcl>
  <LIPClass:qcl rdf:datatype="&xsd;string">sql.concept2</LIPClass:qcl>
  <LIPClass:qcl rdf:datatype="&xsd;string">sql.concept3</LIPClass:qcl>
  <LIPClass:qcl rdf:datatype="&xsd;string">sql.concept4</LIPClass:qcl>
  <LIPClass:qcl rdf:datatype="&xsd;string">sql.concept5</LIPClass:qcl>
  <LIPClass:transcript rdf:datatype="&xsd;string">relational.concept1</LIPClass:transcript>
  <LIPClass:transcript rdf:datatype="&xsd;string">relational.concept2</LIPClass:transcript>
  <LIPClass:transcript rdf:datatype="&xsd;string">relational.concept3</LIPClass:transcript>
  <LIPClass:transcript rdf:datatype="&xsd;string">relational.concept4</LIPClass:transcript>
  <LIPClass:transcript rdf:datatype="&xsd;string">relational.concept5</LIPClass:transcript>
</LIPClass:LearnerInformation>

<LIPClass:LIP rdf:about="#user8"
              rdfs:label="user8">
  <LIPClass:hasIdentification rdf:resource="#iduser8"/>
  <LIPClass:hasLearnerInformation rdf:resource="#liuser8"/>
</LIPClass:LIP>
    <!-- Instance Begins -->
```

```
<LIPClass:Identification rdf:about="#iduser9"
                         rdfs:label="iduser9">
  <LIPClass:name rdf:datatype="&xsd;string">name</LIPClass:name>
  <LIPClass:uid rdf:datatype="&xsd;string">user9</LIPClass:uid>
</LIPClass:Identification>

<LIPClass:LearnerInformation rdf:about="#liuser9"
                             rdfs:label="liuser9">
  <LIPClass:competency rdf:datatype="&xsd;string">competency1</LIPClass:competency>
  <LIPClass:competency rdf:datatype="&xsd;string">competency2</LIPClass:competency>
  <LIPClass:competency rdf:datatype="&xsd;string">competency3</LIPClass:competency>
  <LIPClass:competency rdf:datatype="&xsd;string">competency4</LIPClass:competency>
  <LIPClass:competency rdf:datatype="&xsd;string">competency5</LIPClass:competency>
  <LIPClass:interest rdf:datatype="&xsd;string">sql</LIPClass:interest>
  <LIPClass:qcl rdf:datatype="&xsd;string">sql.concept1</LIPClass:qcl>
  <LIPClass:qcl rdf:datatype="&xsd;string">sql.concept2</LIPClass:qcl>
  <LIPClass:qcl rdf:datatype="&xsd;string">sql.concept3</LIPClass:qcl>
  <LIPClass:qcl rdf:datatype="&xsd;string">sql.concept4</LIPClass:qcl>
  <LIPClass:qcl rdf:datatype="&xsd;string">sql.concept5</LIPClass:qcl>
  <LIPClass:transcript rdf:datatype="&xsd;string">relational.concept1</LIPClass:transcript>
  <LIPClass:transcript rdf:datatype="&xsd;string">relational.concept2</LIPClass:transcript>
  <LIPClass:transcript rdf:datatype="&xsd;string">relational.concept3</LIPClass:transcript>
  <LIPClass:transcript rdf:datatype="&xsd;string">relational.concept4</LIPClass:transcript>
  <LIPClass:transcript rdf:datatype="&xsd;string">relational.concept5</LIPClass:transcript>
</LIPClass:LearnerInformation>

<LIPClass:LIP rdf:about="#user9"
              rdfs:label="user9">
  <LIPClass:hasIdentification rdf:resource="#iduser9"/>
  <LIPClass:hasLearnerInformation rdf:resource="#liuser9"/>
</LIPClass:LIP>
    <!-- Instance Begins -->
<LIPClass:Identification rdf:about="#iduser10"
                         rdfs:label="iduser10">
  <LIPClass:name rdf:datatype="&xsd;string">name</LIPClass:name>
  <LIPClass:uid rdf:datatype="&xsd;string">user10</LIPClass:uid>
</LIPClass:Identification>

<LIPClass:LearnerInformation rdf:about="#liuser10"
                             rdfs:label="liuser10">
  <LIPClass:competency rdf:datatype="&xsd;string">competency1</LIPClass:competency>
  <LIPClass:competency rdf:datatype="&xsd;string">competency2</LIPClass:competency>
  <LIPClass:competency rdf:datatype="&xsd;string">competency3</LIPClass:competency>
  <LIPClass:competency rdf:datatype="&xsd;string">competency4</LIPClass:competency>
  <LIPClass:competency rdf:datatype="&xsd;string">competency5</LIPClass:competency>
  <LIPClass:interest rdf:datatype="&xsd;string">sql</LIPClass:interest>
  <LIPClass:qcl rdf:datatype="&xsd;string">sql.concept1</LIPClass:qcl>
  <LIPClass:qcl rdf:datatype="&xsd;string">sql.concept2</LIPClass:qcl>
  <LIPClass:qcl rdf:datatype="&xsd;string">sql.concept3</LIPClass:qcl>
  <LIPClass:qcl rdf:datatype="&xsd;string">sql.concept4</LIPClass:qcl>
  <LIPClass:qcl rdf:datatype="&xsd;string">sql.concept5</LIPClass:qcl>
  <LIPClass:transcript rdf:datatype="&xsd;string">relational.concept1</LIPClass:transcript>
  <LIPClass:transcript rdf:datatype="&xsd;string">relational.concept2</LIPClass:transcript>
  <LIPClass:transcript rdf:datatype="&xsd;string">relational.concept3</LIPClass:transcript>
  <LIPClass:transcript rdf:datatype="&xsd;string">relational.concept4</LIPClass:transcript>
  <LIPClass:transcript rdf:datatype="&xsd;string">relational.concept5</LIPClass:transcript>
</LIPClass:LearnerInformation>

<LIPClass:LIP rdf:about="#user10"
              rdfs:label="user10">
  <LIPClass:hasIdentification rdf:resource="#iduser10"/>
  <LIPClass:hasLearnerInformation rdf:resource="#liuser10"/>
</LIPClass:LIP>
    <!-- Instance Begins -->
<LIPClass:Identification rdf:about="#iduser10"
                         rdfs:label="iduser10">
  <LIPClass:name rdf:datatype="&xsd;string">name</LIPClass:name>
  <LIPClass:uid rdf:datatype="&xsd;string">user10</LIPClass:uid>
</LIPClass:Identification>

<LIPClass:LearnerInformation rdf:about="#liuser10"
                             rdfs:label="liuser10">
  <LIPClass:competency rdf:datatype="&xsd;string">competency1</LIPClass:competency>
  <LIPClass:competency rdf:datatype="&xsd;string">competency2</LIPClass:competency>
  <LIPClass:competency rdf:datatype="&xsd;string">competency3</LIPClass:competency>
  <LIPClass:competency rdf:datatype="&xsd;string">competency4</LIPClass:competency>
  <LIPClass:competency rdf:datatype="&xsd;string">competency5</LIPClass:competency>
  <LIPClass:interest rdf:datatype="&xsd;string">sql</LIPClass:interest>
  <LIPClass:qcl rdf:datatype="&xsd;string">sql.concept1</LIPClass:qcl>
  <LIPClass:qcl rdf:datatype="&xsd;string">sql.concept2</LIPClass:qcl>
  <LIPClass:qcl rdf:datatype="&xsd;string">sql.concept3</LIPClass:qcl>
  <LIPClass:qcl rdf:datatype="&xsd;string">sql.concept4</LIPClass:qcl>
  <LIPClass:qcl rdf:datatype="&xsd;string">sql.concept5</LIPClass:qcl>
  <LIPClass:transcript rdf:datatype="&xsd;string">relational.concept1</LIPClass:transcript>
  <LIPClass:transcript rdf:datatype="&xsd;string">relational.concept2</LIPClass:transcript>
  <LIPClass:transcript rdf:datatype="&xsd;string">relational.concept3</LIPClass:transcript>
  <LIPClass:transcript rdf:datatype="&xsd;string">relational.concept4</LIPClass:transcript>
  <LIPClass:transcript rdf:datatype="&xsd;string">relational.concept5</LIPClass:transcript>
</LIPClass:LearnerInformation>

<LIPClass:LIP rdf:about="#user10"
              rdfs:label="user10">
  <LIPClass:hasIdentification rdf:resource="#iduser10"/>
  <LIPClass:hasLearnerInformation rdf:resource="#liuser10"/>
  </LIPClass:LIP>
</rdf:RDF>
```

A.2.6 Properties-Oriented Version of the LIP Ontology

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE rdf:RDF [
  <!ENTITY LIPProp "http://kdeg.cs.tcd.ie/ontologies/LIPProp#">
  <!ENTITY owl "http://www.w3.org/2002/07/owl#">
  <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#">
  <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#">
  <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#">
]>
<rdf:RDF xml:base="http://kdeg.cs.tcd.ie/ontologies/LIPProp"
         xmlns:LIPProp="&LIPProp;"
         xmlns:owl="&owl;"
         xmlns:rdf="&rdf;"
         xmlns:rdfs="&rdfs;">

<!-- Ontology Information -->
  <owl:Ontology rdf:about=""
                rdfs:label="LIPProp"
                owl:versionInfo="0.1"/>

<!-- Classes -->
  <owl:Class rdf:about="#LearnerInformation"
             rdfs:label="LearnerInformation"/>

<!-- Annotation Properties -->
  <owl:AnnotationProperty rdf:about="&rdfs;label"/>
  <owl:AnnotationProperty rdf:about="&owl;versionInfo"/>

<!-- Datatype Properties -->
  <owl:DatatypeProperty rdf:about="#accessibility"
                        rdfs:label="accessibility"/>
  <owl:DatatypeProperty rdf:about="#activity"
                        rdfs:label="activity"/>
  <owl:DatatypeProperty rdf:about="#address"
                        rdfs:label="address"/>
  <owl:DatatypeProperty rdf:about="#affiliation"
                        rdfs:label="affiliation"/>
  <owl:DatatypeProperty rdf:about="#competency"
                        rdfs:label="competency"/>
  <owl:DatatypeProperty rdf:about="#contactinfo"
                        rdfs:label="contactinfo"/>
  <owl:DatatypeProperty rdf:about="#geo"
                        rdfs:label="geo"/>
  <owl:DatatypeProperty rdf:about="#goal"
                        rdfs:label="goal"/>
  <owl:DatatypeProperty rdf:about="#identification"
```

```
                      rdfs:label="identification"/>
  <owl:DatatypeProperty rdf:about="#interest"
                      rdfs:label="interest"/>
  <owl:DatatypeProperty rdf:about="#name"
                      rdfs:label="name"/>
  <owl:DatatypeProperty rdf:about="#qcl"
                      rdfs:label="qcl"/>
  <owl:DatatypeProperty rdf:about="#relationship"
                      rdfs:label="relationship"/>
  <owl:DatatypeProperty rdf:about="#securitykey"
                      rdfs:label="securitykey"/>
  <owl:DatatypeProperty rdf:about="#transcript"
                      rdfs:label="transcript"/>
  <owl:DatatypeProperty rdf:about="#uid"
                      rdfs:label="uid"/>

<!-- Instances -->
  <LIPProp:LearnerInformation rdf:about="#user1"
                            rdfs:label="user1">
    <LIPProp:competency rdf:datatype="&xsd;string">sql.competency1</LIPProp:competency>
    <LIPProp:competency rdf:datatype="&xsd;string">sql.competency2</LIPProp:competency>
    <LIPProp:competency rdf:datatype="&xsd;string">sql.competency3</LIPProp:competency>
    <LIPProp:competency rdf:datatype="&xsd;string">sql.competency4</LIPProp:competency>
    <LIPProp:competency rdf:datatype="&xsd;string">sql.competency5</LIPProp:competency>
    <LIPProp:transcript rdf:datatype="&xsd;string">relational.concept1</LIPProp:transcript>
    <LIPProp:transcript rdf:datatype="&xsd;string">relational.concept2</LIPProp:transcript>
    <LIPProp:transcript rdf:datatype="&xsd;string">relational.concept3</LIPProp:transcript>
    <LIPProp:transcript rdf:datatype="&xsd;string">relational.concept4</LIPProp:transcript>
    <LIPProp:transcript rdf:datatype="&xsd;string">relational.concept5</LIPProp:transcript>
    <LIPProp:interest rdf:datatype="&xsd;string">sql</LIPProp:interest>
    <LIPProp:qcl rdf:datatype="&xsd;string">sql.concept1</LIPProp:qcl>
    <LIPProp:qcl rdf:datatype="&xsd;string">sql.concept2</LIPProp:qcl>
    <LIPProp:qcl rdf:datatype="&xsd;string">sql.concept3</LIPProp:qcl>
    <LIPProp:qcl rdf:datatype="&xsd;string">sql.concept4</LIPProp:qcl>
    <LIPProp:qcl rdf:datatype="&xsd;string">sql.concept5</LIPProp:qcl>
    <LIPProp:uid rdf:datatype="&xsd;string">user1</LIPProp:uid>
  </LIPProp:LearnerInformation>

  <LIPProp:LearnerInformation rdf:about="#user2"
                            rdfs:label="user2">
    <LIPProp:competency rdf:datatype="&xsd;string">sql.competency1</LIPProp:competency>
    <LIPProp:competency rdf:datatype="&xsd;string">sql.competency2</LIPProp:competency>
    <LIPProp:competency rdf:datatype="&xsd;string">sql.competency3</LIPProp:competency>
    <LIPProp:competency rdf:datatype="&xsd;string">sql.competency4</LIPProp:competency>
    <LIPProp:competency rdf:datatype="&xsd;string">sql.competency5</LIPProp:competency>
    <LIPProp:interest rdf:datatype="&xsd;string">sql</LIPProp:interest>
    <LIPProp:transcript rdf:datatype="&xsd;string">relational.concept1</LIPProp:transcript>
    <LIPProp:transcript rdf:datatype="&xsd;string">relational.concept2</LIPProp:transcript>
    <LIPProp:transcript rdf:datatype="&xsd;string">relational.concept3</LIPProp:transcript>
    <LIPProp:transcript rdf:datatype="&xsd;string">relational.concept4</LIPProp:transcript>
    <LIPProp:transcript rdf:datatype="&xsd;string">relational.concept5</LIPProp:transcript>
    <LIPProp:qcl rdf:datatype="&xsd;string">sql.concept1</LIPProp:qcl>
    <LIPProp:qcl rdf:datatype="&xsd;string">sql.concept2</LIPProp:qcl>
    <LIPProp:qcl rdf:datatype="&xsd;string">sql.concept3</LIPProp:qcl>
    <LIPProp:qcl rdf:datatype="&xsd;string">sql.concept4</LIPProp:qcl>
    <LIPProp:qcl rdf:datatype="&xsd;string">sql.concept5</LIPProp:qcl>
    <LIPProp:uid rdf:datatype="&xsd;string">user2</LIPProp:uid>
  </LIPProp:LearnerInformation>

  <LIPProp:LearnerInformation rdf:about="#user3"
                              rdfs:label="user3">
    <LIPProp:competency rdf:datatype="&xsd;string">sql.competency1</LIPProp:competency>
    <LIPProp:competency rdf:datatype="&xsd;string">sql.competency2</LIPProp:competency>
    <LIPProp:competency rdf:datatype="&xsd;string">sql.competency3</LIPProp:competency>
    <LIPProp:competency rdf:datatype="&xsd;string">sql.competency4</LIPProp:competency>
    <LIPProp:competency rdf:datatype="&xsd;string">sql.competency5</LIPProp:competency>
    <LIPProp:interest rdf:datatype="&xsd;string">sql</LIPProp:interest>
    <LIPProp:transcript rdf:datatype="&xsd;string">relational.concept1</LIPProp:transcript>
    <LIPProp:transcript rdf:datatype="&xsd;string">relational.concept2</LIPProp:transcript>
    <LIPProp:transcript rdf:datatype="&xsd;string">relational.concept3</LIPProp:transcript>
    <LIPProp:transcript rdf:datatype="&xsd;string">relational.concept4</LIPProp:transcript>
    <LIPProp:transcript rdf:datatype="&xsd;string">relational.concept5</LIPProp:transcript>
    <LIPProp:qcl rdf:datatype="&xsd;string">sql.concept1</LIPProp:qcl>
    <LIPProp:qcl rdf:datatype="&xsd;string">sql.concept2</LIPProp:qcl>
    <LIPProp:qcl rdf:datatype="&xsd;string">sql.concept3</LIPProp:qcl>
    <LIPProp:qcl rdf:datatype="&xsd;string">sql.concept4</LIPProp:qcl>
    <LIPProp:qcl rdf:datatype="&xsd;string">sql.concept5</LIPProp:qcl>
    <LIPProp:uid rdf:datatype="&xsd;string">user3</LIPProp:uid>
  </LIPProp:LearnerInformation>

  <LIPProp:LearnerInformation rdf:about="#user4"
                              rdfs:label="user4">
    <LIPProp:competency rdf:datatype="&xsd;string">sql.competency1</LIPProp:competency>
    <LIPProp:competency rdf:datatype="&xsd;string">sql.competency2</LIPProp:competency>
    <LIPProp:competency rdf:datatype="&xsd;string">sql.competency3</LIPProp:competency>
    <LIPProp:competency rdf:datatype="&xsd;string">sql.competency4</LIPProp:competency>
    <LIPProp:competency rdf:datatype="&xsd;string">sql.competency5</LIPProp:competency>
    <LIPProp:interest rdf:datatype="&xsd;string">sql</LIPProp:interest>
    <LIPProp:transcript rdf:datatype="&xsd;string">relational.concept1</LIPProp:transcript>
    <LIPProp:transcript rdf:datatype="&xsd;string">relational.concept2</LIPProp:transcript>
    <LIPProp:transcript rdf:datatype="&xsd;string">relational.concept3</LIPProp:transcript>
    <LIPProp:transcript rdf:datatype="&xsd;string">relational.concept4</LIPProp:transcript>
    <LIPProp:transcript rdf:datatype="&xsd;string">relational.concept5</LIPProp:transcript>
    <LIPProp:qcl rdf:datatype="&xsd;string">sql.concept1</LIPProp:qcl>
    <LIPProp:qcl rdf:datatype="&xsd;string">sql.concept2</LIPProp:qcl>
    <LIPProp:qcl rdf:datatype="&xsd;string">sql.concept3</LIPProp:qcl>
    <LIPProp:qcl rdf:datatype="&xsd;string">sql.concept4</LIPProp:qcl>
    <LIPProp:qcl rdf:datatype="&xsd;string">sql.concept5</LIPProp:qcl>
    <LIPProp:uid rdf:datatype="&xsd;string">user4</LIPProp:uid>
  </LIPProp:LearnerInformation>

  <LIPProp:LearnerInformation rdf:about="#user5"
                              rdfs:label="user5">
    <LIPProp:competency rdf:datatype="&xsd;string">sql.competency1</LIPProp:competency>
    <LIPProp:competency rdf:datatype="&xsd;string">sql.competency2</LIPProp:competency>
    <LIPProp:competency rdf:datatype="&xsd;string">sql.competency3</LIPProp:competency>
    <LIPProp:competency rdf:datatype="&xsd;string">sql.competency4</LIPProp:competency>
    <LIPProp:competency rdf:datatype="&xsd;string">sql.competency5</LIPProp:competency>
    <LIPProp:interest rdf:datatype="&xsd;string">sql</LIPProp:interest>
    <LIPProp:transcript rdf:datatype="&xsd;string">relational.concept1</LIPProp:transcript>
    <LIPProp:transcript rdf:datatype="&xsd;string">relational.concept2</LIPProp:transcript>
    <LIPProp:transcript rdf:datatype="&xsd;string">relational.concept3</LIPProp:transcript>
    <LIPProp:transcript rdf:datatype="&xsd;string">relational.concept4</LIPProp:transcript>
    <LIPProp:transcript rdf:datatype="&xsd;string">relational.concept5</LIPProp:transcript>
    <LIPProp:qcl rdf:datatype="&xsd;string">sql.concept1</LIPProp:qcl>
    <LIPProp:qcl rdf:datatype="&xsd;string">sql.concept2</LIPProp:qcl>
    <LIPProp:qcl rdf:datatype="&xsd;string">sql.concept3</LIPProp:qcl>
    <LIPProp:qcl rdf:datatype="&xsd;string">sql.concept4</LIPProp:qcl>
    <LIPProp:qcl rdf:datatype="&xsd;string">sql.concept5</LIPProp:qcl>
    <LIPProp:uid rdf:datatype="&xsd;string">user5</LIPProp:uid>
  </LIPProp:LearnerInformation>

  <LIPProp:LearnerInformation rdf:about="#user6"
                              rdfs:label="user6">
    <LIPProp:competency rdf:datatype="&xsd;string">sql.competency1</LIPProp:competency>
    <LIPProp:competency rdf:datatype="&xsd;string">sql.competency2</LIPProp:competency>
    <LIPProp:competency rdf:datatype="&xsd;string">sql.competency3</LIPProp:competency>
    <LIPProp:competency rdf:datatype="&xsd;string">sql.competency4</LIPProp:competency>
    <LIPProp:competency rdf:datatype="&xsd;string">sql.competency5</LIPProp:competency>
    <LIPProp:interest rdf:datatype="&xsd;string">sql</LIPProp:interest>
    <LIPProp:transcript rdf:datatype="&xsd;string">relational.concept1</LIPProp:transcript>
    <LIPProp:transcript rdf:datatype="&xsd;string">relational.concept2</LIPProp:transcript>
    <LIPProp:transcript rdf:datatype="&xsd;string">relational.concept3</LIPProp:transcript>
    <LIPProp:transcript rdf:datatype="&xsd;string">relational.concept4</LIPProp:transcript>
    <LIPProp:transcript rdf:datatype="&xsd;string">relational.concept5</LIPProp:transcript>
    <LIPProp:qcl rdf:datatype="&xsd;string">sql.concept1</LIPProp:qcl>
    <LIPProp:qcl rdf:datatype="&xsd;string">sql.concept2</LIPProp:qcl>
    <LIPProp:qcl rdf:datatype="&xsd;string">sql.concept3</LIPProp:qcl>
    <LIPProp:qcl rdf:datatype="&xsd;string">sql.concept4</LIPProp:qcl>
    <LIPProp:qcl rdf:datatype="&xsd;string">sql.concept5</LIPProp:qcl>
    <LIPProp:uid rdf:datatype="&xsd;string">user6</LIPProp:uid>
```

```
</LIPProp:LearnerInformation>

<LIPProp:LearnerInformation rdf:about="#user7"
                            rdfs:label="user7">
  <LIPProp:competency rdf:datatype="&xsd;string">sql.competency1</LIPProp:competency>
  <LIPProp:competency rdf:datatype="&xsd;string">sql.competency2</LIPProp:competency>
  <LIPProp:competency rdf:datatype="&xsd;string">sql.competency3</LIPProp:competency>
  <LIPProp:competency rdf:datatype="&xsd;string">sql.competency4</LIPProp:competency>
  <LIPProp:competency rdf:datatype="&xsd;string">sql.competency5</LIPProp:competency>
  <LIPProp:interest rdf:datatype="&xsd;string">sql</LIPProp:interest>
  <LIPProp:transcript rdf:datatype="&xsd;string">relational.concept1</LIPProp:transcript>
  <LIPProp:transcript rdf:datatype="&xsd;string">relational.concept2</LIPProp:transcript>
  <LIPProp:transcript rdf:datatype="&xsd;string">relational.concept3</LIPProp:transcript>
  <LIPProp:transcript rdf:datatype="&xsd;string">relational.concept4</LIPProp:transcript>
  <LIPProp:transcript rdf:datatype="&xsd;string">relational.concept5</LIPProp:transcript>
  <LIPProp:qcl rdf:datatype="&xsd;string">sql.concept1</LIPProp:qcl>
  <LIPProp:qcl rdf:datatype="&xsd;string">sql.concept2</LIPProp:qcl>
  <LIPProp:qcl rdf:datatype="&xsd;string">sql.concept3</LIPProp:qcl>
  <LIPProp:qcl rdf:datatype="&xsd;string">sql.concept4</LIPProp:qcl>
  <LIPProp:qcl rdf:datatype="&xsd;string">sql.concept5</LIPProp:qcl>
  <LIPProp:uid rdf:datatype="&xsd;string">user7</LIPProp:uid>
</LIPProp:LearnerInformation>

<LIPProp:LearnerInformation rdf:about="#user8"
                            rdfs:label="user8">
  <LIPProp:competency rdf:datatype="&xsd;string">sql.competency1</LIPProp:competency>
  <LIPProp:competency rdf:datatype="&xsd;string">sql.competency2</LIPProp:competency>
  <LIPProp:competency rdf:datatype="&xsd;string">sql.competency3</LIPProp:competency>
  <LIPProp:competency rdf:datatype="&xsd;string">sql.competency4</LIPProp:competency>
  <LIPProp:competency rdf:datatype="&xsd;string">sql.competency5</LIPProp:competency>
  <LIPProp:interest rdf:datatype="&xsd;string">sql</LIPProp:interest>
  <LIPProp:transcript rdf:datatype="&xsd;string">relational.concept1</LIPProp:transcript>
  <LIPProp:transcript rdf:datatype="&xsd;string">relational.concept2</LIPProp:transcript>
  <LIPProp:transcript rdf:datatype="&xsd;string">relational.concept3</LIPProp:transcript>
  <LIPProp:transcript rdf:datatype="&xsd;string">relational.concept4</LIPProp:transcript>
  <LIPProp:transcript rdf:datatype="&xsd;string">relational.concept5</LIPProp:transcript>
  <LIPProp:qcl rdf:datatype="&xsd;string">sql.concept1</LIPProp:qcl>
  <LIPProp:qcl rdf:datatype="&xsd;string">sql.concept2</LIPProp:qcl>
  <LIPProp:qcl rdf:datatype="&xsd;string">sql.concept3</LIPProp:qcl>
  <LIPProp:qcl rdf:datatype="&xsd;string">sql.concept4</LIPProp:qcl>
  <LIPProp:qcl rdf:datatype="&xsd;string">sql.concept5</LIPProp:qcl>
  <LIPProp:uid rdf:datatype="&xsd;string">user8</LIPProp:uid>
</LIPProp:LearnerInformation>

<LIPProp:LearnerInformation rdf:about="#user9"
                            rdfs:label="user9">
  <LIPProp:competency rdf:datatype="&xsd;string">sql.competency1</LIPProp:competency>
```

```
    <LIPProp:competency rdf:datatype="&xsd;string">sql.competency2</LIPProp:competency>
    <LIPProp:competency rdf:datatype="&xsd;string">sql.competency3</LIPProp:competency>
    <LIPProp:competency rdf:datatype="&xsd;string">sql.competency4</LIPProp:competency>
    <LIPProp:competency rdf:datatype="&xsd;string">sql.competency5</LIPProp:competency>
    <LIPProp:interest rdf:datatype="&xsd;string">sql</LIPProp:interest>
    <LIPProp:transcript rdf:datatype="&xsd;string">relational.concept1</LIPProp:transcript>
    <LIPProp:transcript rdf:datatype="&xsd;string">relational.concept2</LIPProp:transcript>
    <LIPProp:transcript rdf:datatype="&xsd;string">relational.concept3</LIPProp:transcript>
    <LIPProp:transcript rdf:datatype="&xsd;string">relational.concept4</LIPProp:transcript>
    <LIPProp:transcript rdf:datatype="&xsd;string">relational.concept5</LIPProp:transcript>
    <LIPProp:qcl rdf:datatype="&xsd;string">sql.concept1</LIPProp:qcl>
    <LIPProp:qcl rdf:datatype="&xsd;string">sql.concept2</LIPProp:qcl>
    <LIPProp:qcl rdf:datatype="&xsd;string">sql.concept3</LIPProp:qcl>
    <LIPProp:qcl rdf:datatype="&xsd;string">sql.concept4</LIPProp:qcl>
    <LIPProp:qcl rdf:datatype="&xsd;string">sql.concept5</LIPProp:qcl>
    <LIPProp:uid rdf:datatype="&xsd;string">user9</LIPProp:uid>
  </LIPProp:LearnerInformation>

  <LIPProp:LearnerInformation rdf:about="#user10"
                              rdfs:label="user10">
    <LIPProp:competency rdf:datatype="&xsd;string">sql.competency1</LIPProp:competency>
    <LIPProp:competency rdf:datatype="&xsd;string">sql.competency2</LIPProp:competency>
    <LIPProp:competency rdf:datatype="&xsd;string">sql.competency3</LIPProp:competency>
    <LIPProp:competency rdf:datatype="&xsd;string">sql.competency4</LIPProp:competency>
    <LIPProp:competency rdf:datatype="&xsd;string">sql.competency5</LIPProp:competency>
    <LIPProp:interest rdf:datatype="&xsd;string">sql</LIPProp:interest>
    <LIPProp:transcript rdf:datatype="&xsd;string">relational.concept1</LIPProp:transcript>
    <LIPProp:transcript rdf:datatype="&xsd;string">relational.concept2</LIPProp:transcript>
    <LIPProp:transcript rdf:datatype="&xsd;string">relational.concept3</LIPProp:transcript>
    <LIPProp:transcript rdf:datatype="&xsd;string">relational.concept4</LIPProp:transcript>
    <LIPProp:transcript rdf:datatype="&xsd;string">relational.concept5</LIPProp:transcript>
    <LIPProp:qcl rdf:datatype="&xsd;string">sql.concept1</LIPProp:qcl>
    <LIPProp:qcl rdf:datatype="&xsd;string">sql.concept2</LIPProp:qcl>
    <LIPProp:qcl rdf:datatype="&xsd;string">sql.concept3</LIPProp:qcl>
    <LIPProp:qcl rdf:datatype="&xsd;string">sql.concept4</LIPProp:qcl>
    <LIPProp:qcl rdf:datatype="&xsd;string">sql.concept5</LIPProp:qcl>
    <LIPProp:uid rdf:datatype="&xsd;string">user10</LIPProp:uid>
  </LIPProp:LearnerInformation>
</rdf:RDF>
```

Appendix B

Glossary of Terms

Articulation: In Semantic Interoperation, the articulation consists of the representation of the mappings combined with the references to the mapped ontology concepts.

Context: External information which is relevant to a target application, but which the application cannot normally access.

Context Mediator: An application which is able to analyse ontological knowledge and broker information from sources of context to enrich a target application

Participant: A service which is either a Target Application or a Source of Context.

Ontology Mapping: the process of relating symbols and axioms in two ontologies which share the same domain of discourse. [Kalfoglou and Schorlemmer, 2003]

Ontology Alignment: The process of using relations taken from an external ontology to describe the links between the symbols in the ontologies. [Kalfoglou and Schorlemmer, 2003]

Ontology Mediation: The process of transferring information from one ontology to another, while keeping the ontologies separate.

Schema Manager: The ACP component responsible for loading and providing access to ontologies in the system. In the ACP, this component is a set of methods which are backed by the Jena Ontology Framework.

Shared Semantic View: The representation of the concepts within the Target Application and the Sources of Context, along with a description of the links between them. In the ACP, this is represented by a Topic Map.

Reasoner: The ACP contains custom reasoners, which are Java applications loaded on demand to perform tasks such as uplifting ontologies, transferring information between ontologies or altering the Shared Semantic View.

Source of Context: A service or knowledge base which contains some context information.

Target Application: A service or application which is to be enriched with context knowledge.