Agent-Based Ontology Management towards Interoperability

By
Li Li
BSc (Southwest China Normal University, China)
MSc (Southwest China Normal University, China)

Submitted in fulfillment of the requirements for the degree of
Doctor of Philosophy
of Swinburne University of Technology
December 2005

SWINBURNE UNIVERSITY OF TECHNOLOGY

CANDIDATE DECLARATION

I certify that the thesis entitled:

Agent-Based Ontology Management Towards Ontology Interoperability

submitted for the degree of:

Doctor of Philosophy

is the result of my own research, except where otherwise acknowledged, and that this thesis in whole or in part has not been submitted for an award, including a higher degree, to any other university or institution.

Full Name: Li Li

Signed:..... Date:.....

To My Family

Abstract

Ontologies are widely used as data representations for knowledge bases and marking up data on the emerging Semantic Web. Hence, techniques for managing ontology come to the centre of any practical and general solution of knowledge-based systems.

Challenges arise when we look a step further in order to achieve flexibility and scalability of the ontology management. Previous works in ontology management, primarily for *ontology mapping*, *ontology integration* and *ontology evolution*, have exploited only one form or another of ontology management in restrictive settings. However, a distributed and heterogeneous environment makes it necessary for researchers in this field to consider ontology interoperability in order to achieve the vision of the Semantic Web. Several challenges arise when we set our goal to achieve ontology interoperability on the Web. The first one is to decide which software engineering paradigm to employ. The issue of such a paradigm is the core of ontology management when dynamic property is involved. It should make it easy to model complex systems and significantly improve current practice in software engineering. Moreover, it allows the extension of the range of applications that can feasibly be tackled. The second challenge is to exploit frameworks based on the proposed paradigm. Such a framework should make possible flexibility, interactivity, reusability and reliability for systems which are built on it. The third challenge is to investigate suitable mechanisms to cope with *ontology mapping*, *integration* and *evolution* based on the framework. It is known that predefined rules or hypotheses may not apply given that the environment hosting an ontology is changing over time.

Fortunately, agents are being advocated as a next generation model for engineering complex and distributed systems. Also some researchers in this field have given a qualitative analysis to provide a justification for precisely why the agent-based approach is well suited to engineer complex software systems. From a multi-agent perspective, agent technology fits well in developing applications in uncontrolled and distributed environments which require substantial support for change. Agents in multi-agent systems (MAS) are autonomous and can engage in interactions which are essential for any ongoing agents' actions. A MAS approach is thus regarded as an intuitive and suitable way of modelling dynamic systems. Following the above discussion, an agent-based framework for managing ontology in a dynamic environment is developed. The framework has several key characteristics such as flexibility and extensibility that differentiate this research from others. Three important issues of the ontology management are also investigated. It is believed that inter-ontology processes like ontology mapping with logical semantics are foundations of ontology-based applications. Hence, firstly, *ontology mapping* is discussed. Several types of semantic relations are proposed. Following these, the mapping mechanisms are developed. Secondly, based on the previous mapping results, *ontology integration* is developed to provide abstract views for participating organisations in the presence of a variety of ontologies. Thirdly, as an ontology is subject to evolution in its life cycle, there must be some kind of mechanisms to reflect their changes in corresponding interrelated ontologies. *Ontology refinement* is investigated to take ontology evolution into consideration. Process algebra is employed to catch and model information exchanges between ontologies. Agent negotiation strategy is applied to guide corresponding ontologies to react properly.

A prototype is built to demonstrate the above design and functionalities. It is applied to ontologies dealing with the subject of beer (type). This prototype consists of four major types of agents, ranging from `user agent`, `interface agent`, `ontology agent`, and `functionary agent`. Evaluations such as *query*, *consistency checking* are conducted on the prototype. This shows that the framework is not only flexible but also completely workable. All agents derived from the framework exhibit their behaviours appropriately as expected.

Acknowledgments

First, I would like to thank my coordinating supervisor, Professor, Yun Yang, for his helpful supervision and continuous encouragement throughout, his guidance and stimulating discussions during the course of this work. He has always given me timely and constructive feedback for my research ideas, papers and thesis drafts. His suggestions have been helpful, and his comments and questions are very insightful. Without his consistent support, I would not have been able to complete this manuscript. I am lucky and happy to have been able to work with him during my PhD program. I would also like to thank my associate supervisor, Dr. Baolin Wu, for his valuable advice and useful feedback on my work which helped to make it better.

I am grateful to the Faculty of Information and Communication Technologies (FICT) and Swinburne University of Technology, which have offered me Research Scholarships throughout my doctoral program. FICT has provided an excellent environment for learning and experiencing, and sponsored my participation in conferences. At the FICT, I must acknowledge research administration coordinators Mrs. Neroli Finlay and Ms. Charlotte Swain for their tireless efforts to make my life in Swinburne easier.

Thanks to all those who helped me in one way or another during my PhD study: Dr. Jun Yan, Dr. Dongang Yu, Jinjun Chen, Xiaohui Zhao and Shane Grund.

Finally, I am deeply grateful to my family, who have always believed in me and provided love and support over all these years. I would never have achieved what I have without their encouragement. I cannot imagine completing graduate study without their help. I am forever in their debt.

List of Publications

The following is a list of my research papers published in referred conference proceedings during my PhD study at Swinburne University of Technology:

1. Li, L., Yang, Y., and Wu, B., Agent-based Ontology Integration for Ontology-Based Application, in: T. Meyer, M. Orgun (Eds.), *Proc. of Australasian Ontology Workshop* (AOW 2005), the 18th Australian Joint Conference on Artificial Intelligence, Conference in Research and Practice in Information Technology (CRPIT) series by Australian Computer Society, Vol 58, pp. 53-59, Sydney, Australia, December 2005.

2. Li, L., Yang, Y., and Wu, B., Agent-based Ontology Mapping towards Ontology Interoperability, *Proc. of the 18th Australian Joint Conference on Artificial Intelligence* (AI05), LNAI 3809, Springer-Verlag, pp. 843-846, Sydney, Australia, December 2005.

3. Li, L., Wu, B., and Yang, Y., Agent-based Approach for Dynamic Ontology Management, *Proc. of the 9th International Conference on Knowledge-Based Intelligence Information & Engineering Systems* (KES2005), LNCS 3683, Part III, Springer-Verlag, pp. 1-7, Melbourne, Australia, September 2005.

4. Wu, B., Dewan, M., Li, L. and Yang, Y., Supply Chain Protocolling, *IEEE Conference on E-Commerce Technology* (CEC'05), pp. 314-321, München, Germany, July 2005.

5. Li, L., Yang, Y., and Wu, B., Agent-based Approach towards Ontology Refinement in Virtual Enterprises, *Proc. of the 3rd International Conference on*

Active Media Technology (AMT 2005), pp. 250-255, Kagawa, Japan, May 2005.

6. Li, L., Wu, B., and Yang, Y., Ontology-based Matchmaking in e-Marketplace with Web Services, *Proc. of the 6th Asia Pacific Web Conference*, LNCS 3399, Springer-Verlag, pp. 620-631, Shanghai, China, March 2005.

7. Li, L., Wu, B., and Yang, Y., Semantic Mapping with Multi-Agent Systems, *Proc. of IEEE International Conference on e-Technology, e-Commerce and e-Service* (EEE-05), pp. 54-57, Hong Kong, March 2005.

8. Li, L., Wu, B., and Yang, Y., Refinement for Ontology Evolution in Virtual Enterprises, *Proc. of the 4th International Conference on Electronic Business* (ICEB2004), pp. 696-701, Beijing, China, December 2004.

9. Wu, B., Li, L., and Yang, Y., E-Business Value Process Modelling, *Proc. of the 4th International Conference on Electronic Business* (ICEB2004), pp. 408-413, Beijing, China, December 2004.

10. Li, L., Wu, B., and Yang, Y., Developing Ontology in Virtual Enterprises, *Proc. of International Conference on Intelligent Agents, Web Technologies and Internet Commerce*, CD ISBN 1740881893, pp. 337-343, Gold Coast, Australia, July 2004.

11. Li, L., Wu B., and Yang, Y., An Ontology-Oriented Approach for Virtual Enterprises, *Proc. of the 5th Asia Pacific Web Conference*, LNCS 3007, Springer-Verlag, pp. 834-843, Hangzhou, China, April 2004.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Ontology management towards interoperability is a key for managing ontologies based on certain applications. An ontology is *an explicit specification of a conceptualisation* [38]. Ontologies are widely used as data representations for knowledge bases and marking up data on the emerging Semantic Web [11]. Hence, techniques for managing ontology come to the centre of any practical and general solution of knowledge-based systems. The vision of ontology for different purposes, such as Web-enabled applications, Web semantics, information systems, is receiving increasing attention both from academia and industry. In the context of knowledge sharing, an ontology is a specification used for making ontological commitment which is an agreement to use a vocabulary in a way that is consistent with respect to the theory specified by an ontology [37, 38]. The next generation of the WWW, the so-called Semantic Web, is based on using ontologies for annotating content with formal semantics. Ultimately, ontologies enable data/information integration in the context of the semantics. They would widen the accessibility of information by allowing the use of multiple ontologies belonging to diverse organisations. The benefits of using ontologies have been recognised in many areas such as knowledge and content management, e-commerce and the Semantic Web. Approaches based on ontologies have shown the advantages for both information integration and system interoperability including reusability, extensibility, verification analysis and so on.

Based on past experience, for example, CYC [69], a monolith ontology is unlikely to be constructed due to the large scale, individual privacy needs, dynamics and heterogeneity. As a typical example, the proliferation of Internet technology and globalisation of business environments give rise to the advent of dynamic virtual alliances [14] such as virtual organisations (VOs) among complementary organisations. It is hardly conceivable that a single ontology can be applied to all involved parties and applications. In addition, in numerous domains there is always a need to deal with *different purposes* to keep their uniqueness. However, it seems that no satisfactory solutions to semantic mapping exist, because today the vast majority of semantic mappings is still created manually. The slow and expensive manual creation of mappings has now become a serious problem in building ontology-based applications. The problem becomes even more critical when a changeable environment (e.g. the Web) is involved, in which ontologies and ontology-based applications might proliferate and scale up. Also a changeable environment enforces underlying ontologies to evolve over time. Finally, the development of technologies such as the Semantic Web will further fuel deployment of ontologies to enable knowledge sharing and reuse across different sources. Manual mapping is simply not possible for such scales. Given semantic mapping is a fundamental step in numerous ontology-based applications, it is time to think about the development of solutions which are flexible, robust and applicable, to meet the requirements of dynamic changes in an environment. Besides *ontology mapping*, *ontology integration* and *evolution* are also critical and need to be investigated in ontology management.

When heterogeneity, distribution, autonomy and evolution are concerned in ontologies, achieving effective interoperability of ontologies manually seems unlikely. Interactions between agents on demand for the purpose of understanding each other, which is a potential solution to addressing environment change, is treated as a flexible approach and seen as a key technology at the knowledge view level. As understanding in a particular domain is required before any decision is made, it is regarded as a feasible and effective way to find certain semantics in a certain business scenario at run-time. Besides VOs, the very viable and rapid growth of the Web has made it even more prominent than ever before. Addressing ontology

management in a timely fashion and providing a conformance view for participating organisations at an abstract level is thus seen as necessary. Therefore, run-time interactions are needed for ontology management. However, existing knowledge management cannot cope with these interactions efficiently as they may occur at unpredictable times between unpredictable parts of ontologies.

An agent is capable of flexible, autonomous actions in an environment in order to meet its design objectives [147]. A multi-agent system (MAS) is a collection of autonomous agents working together to solve problems that are beyond the overall capability of individual agents. MASs [150, 81] fit well in the situations where dynamic interactions and communication are needed. Agent-based perspective is thus an appropriate approach for distributed ontology management where operations on diversity of ontologies cannot be achieved at design-time. Through agents taking part in the ontology management at run-time, the bottleneck of information understanding on the basis of agents' interaction will be eased to some extent.

1.1 Ontology and Ontology Management towards Interoperability

The concept of an ontology was initially introduced by Aristotle. In the computer science domain, ontologies aim at capturing domain knowledge in a generic way and provide a commonly agreed understanding of a domain, which may be reused and shared across applications and groups [18]. It has become one of the hot topics in computer science and information technology. Gruber's [38] ontology definition becomes the most referenced in the domain of computer science. In the context of knowledge sharing, an ontology means a specification of a conceptualisation. That is, an ontology is a description (specification) of the concepts and relationships that can exist for an agent or a community of agents. It is a shared understanding which is designed to enable communication for a community of agents (human or intelligent agents). It aims at knowledge sharing and reuse [156]. There are other ontology definitions which emerge as a consequence of their authors building and using ontologies. Refer to Guarino et al. [39] for a deep study of this. We follow Gruber's ontology definition throughout this thesis.

Attention to ontologies and related areas has given rise to the advent of enormous ontology languages and tools, and some of these have been used to build ontology-based applications in the areas of knowledge management, e-business process modelling, negotiation in e-commerce, bioinformatics or medicine.

In addition, ontologies provide support in integrating heterogeneous and distributed information sources. This gives them an important role in areas where achieving interoperability is vital.

With the advent of ontologies, especially given that a monolith ontology is not feasible for technical and non-technical reasons, the issues have increased because of very close relationships between ontologies and the Semantic Web. Actually ontologies are the backbone of the Semantic Web. Frequently, in business, every organisation conforming to a same ontology is infeasible. Ontology heterogeneity arises because different sources have different vocabularies and semantics. Moreover, the proliferation of Internet technology and globalisation of business environments has given rise to the advent of a variety of ontologies which are far beyond expectations, whilst the changing environment enforces underlying ontologies to evolve over time. That achievement of some goals in business requires more than individual capabilities and knowledge, and makes it necessary for different organisations to work together to maximise their own profits. All these make a continuing demand on dealing with ontology operations on the fly rather than defined in advance. Distributed ontologies from varied sources also add the requirement of interoperability, thus system interoperability in the context of the Web is now reduced to ontology interoperability. Dynamic ontology mapping is seen as a promising approach to tackling heterogeneous ontologies in order to pave the way to ontology interoperability and eventually to achieving the goals of. In addition, if visions such as the Semantic Web are ever going to become a reality, it will be necessary to provide as much automated support as possible to the task of mapping different ontologies. On the other hand, an ontology evolves throughout its life cycle. Ontology refinement can be regarded as encouraging to taking ontology evolution into account whenever needed. In short, ontology mapping is fundamental to further ontology managements such as ontology integration (e.g. a global view of available ontologies at an abstract level in order to take advantage of existing resources) and

refinement (e.g. coping with ontology evolution).

Ontology management aims to provide flexible mechanisms to cope with dynamic mapping and ontology integration as well as ontology changing within ontology-based applications.

1.2 Key Research Issues in Ontology Management

As stated in Section 1.1, it is necessary to consider ontology interoperability in the context of the Web. The Web is an application that can operate on global computer networks (*http://www.december.com/web/develop/character.html*). In this sense, the emergence of the Web has made the environment of ontologies and ontology-based applications change rapidly, which in turn has made ontology integration more difficult than ever before. In order to identify the characteristics of problems, let us take a closer look at ontologies and ontology-based applications on the Web. The following are dominant characteristics of the environment:

- Available ontologies are scattered on the Web which makes manual gathering of relevant information for a particular problem nearly impossible.
- The number of organisations (with different ontologies) may change dynamically through organisations coming or leaving the environment randomly. Furthermore, which ontology is available along with a particular organisation is uncertain.
- Ontologies are prone to adaptation and evolution.
- Processes of different organisations operate concurrently to modify the environment in ways over which an organisation has no control.
- Different organisations demand efficient interaction as these interactions may occur at unpredictable times, for unpredictable reasons, between unpredictable organisations. An organisation thus needs to consider *synchronising* or *coordinating* its actions with those of other processes in the environment.

In short, the typical characteristics of ontologies and ontology-based applications on the Web can be identified as: distributed (namely ontologies scattered in different places), non-deterministic[1] (in practice, almost all realistic environments must be regarded as non-deterministic), heterogeneous (ontologies may have various modelling methods and different representations), and highly dynamic (the number of organisations/ontologies changes dramatically, the ontology evolves over time, with many processes concurrently modifying the environment in ways over which an organisation has no control, heterogeneous systems need to interact by spanning organisational boundaries and operating effectively within changing environments). However, existing systems or tools in this field so far have been treated mainly as being static, so it is unlikely that they can tackle ontologies and ontology-based applications efficiently and effectively in the above environment. Actually, existing software development techniques are concerned with what are known as *functional* systems[2]. Systems developed by using classic software engineering paradigms are inherently too simple to fulfill the tasks of an open environment. On the contrary, *reactive* systems[3] [120] are capable of responding rapidly to changes in the environment [150]. From a MAS perspective, agents are autonomous and flexible with major characteristics of reactivity, pro-activity and high interactive ability [150]. MASs are able to operate flexibly and rationally in a variety of environmental circumstances, given their abilities to perceive changes in the environment and respond in a timely fashion. MAS's ability to provide robustness and efficiency and solve problems in which data or control is distributed, makes the MAS perfectly suited for scalable, distributed environments. Agent technology is thus ideally suited to model ontology integration on the Web.

Agent technologies are every promising to take on the responsibilities of carrying out actions with the *interference* of other processes. In this research, an agent-oriented view is ushered in and highlighted in order to meet the requirements of

[1] A deterministic environment is one in which any action has a single guaranteed effect - there is no uncertainty about the state that will result from performing an action.

[2] A *functional* system is one that simply takes some input, performs some computation over this input, and eventually produces some output.

[3] Typically, the main role of *reactive* systems is to maintain an interaction with their environment, and therefore must be described and specified in terms of their on-going behaviours...interacting with its own environment which consists of other modules.

responding promptly to changes in a dynamic and decentralised heterogeneous environment.

1.3 Overview of the Thesis

Many researchers agree that one of the major bottlenecks in ontology-based applications on the Web is ontology mapping. The reason is because there are simply too many ontologies available and they are too large to have manual mapping. Rather than manually defining correspondences which is currently the primary approach in mapping, it is time to have a technique to provide potential ways to map structures (e.g. representations and modelling paradigms) known by participants to new structures encountered on the fly. Hence the task of finding mappings automatically takes the centre stage in the ontology community.

Naturally, mapping is not a goal in itself. The resulting mappings are used for various integration tasks, given that ontologies are expressed in a language that can be used for inference and semantics of ontology specified with reasoning. In addition, ontology evolves over time. An applicable mechanism needs to take this into account to reflect any changes.

In this thesis, our main focus will be on developing an agent-based framework to tackle ontology management issues flexibly and effectively. In other words, agents which consume ontologies also engage in ontology management towards ontology interoperability. We intend to build a MAS to support the proposed framework in order to handle *ontology mapping*, *integration* and *refinement* on demand. In this system, available heterogeneous ontologies are converted to an internal representation and imported to the agent-based system. The system then performs *mapping* followed by *integration* or ontology *refinement* separately, or in parallel. Integrated ontology is then exported in a RDF(s) format. The results of *refinement* are either used through a *mapping* process to impact the current *mapping* and *integration* processes, or exported in RDF(s) formats. The exported ontology or ontologies in RDF(s) can then be reused with other existing ontologies in the agent-based system (Figure 1.1).

To summarise, we discuss three dimensions of ontology management, under an

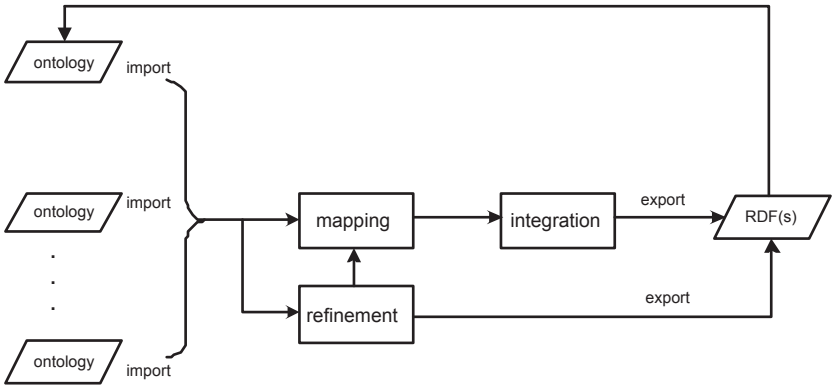

Figure 1.1: Major tasks of the thesis

agent-based framework, in the thesis:

- **Ontology mapping**: Finding mappings between pairs of ontologies. This consists of importing ontologies to the system, deciding which concepts representing similar notions based on defined similarity measurement, exporting mapping results for the next stage (integration), and so on.

- **Ontology integration**: Integrating ontology is based on the previous mapping results. It describes how the system integrates given ontologies and exports in a specific format to allow information exchange between the system it is hosted in and the other knowledge-base frameworks (e.g. Protégé). Ontology reuse is at the core of avoiding error-prone and time-consuming reinvention. This is discussed in detail to enable ontology reuse on demand.

- **Ontology refinement**: As long as an ontology exists, it is subject to change in its life cycle. The negotiation mechanism is applied to cope with changes whenever needed. Exports of ontology refinement may further feed back to the processes (e.g. *mapping* and *integration*), or are the final result of the management system.

In this thesis, we explore these dimensions. Specifically, our goals are as follows:

- to justify why the multi-agent perspective is appropriate for distributed ontology management;

- to develop a general framework;

- to investigate *ontology mapping*, *integration* and *refinement* mechanisms under the proposed framework, in which agent interactions are highlighted for the purposes of dynamic semantic mapping, abstract conformance viewing and unpredicted ontology evolution;
- to implement an agent-based ontology management system, in which *ontology mapping* and *integration* are realised as well as evaluation prototyping;
- to use process algebra to channel agent interactions for ontology evolution (e.g. ontology refinement); and
- to use description logics to identify problems like inconsistency in ontologies.

In saying so, we attempt to make contributions in the the following two aspects throughout this thesis:

- to provide a general framework to support heterogeneous and distributed ontology management;
- to provide a solution to the problem of how to dynamically perform ontology management.

1.4 Thesis Outline

The reminder of the thesis consists of 7 chapters. Chapter 2 is the literature review and requirements analysis. Major chapters in the thesis can be grouped into three levels. The upper level is the work toward the generic framework and mechanisms, which is contained in Chapters 3, 4 and 5. Chapter 3 addresses an agent-based framework for ontology management. Based on the framework, proposed in Chapter 3, Chapter 4 discusses the mapping mechanisms which are treated to pave the way to ontology interoperability in ontology-based applications. Chapter 5 addresses ontology integration based on the results of the mapping process to obtain a global view at an abstract level under certain circumstances. The middle level is the prototyping work of applying the framework and mechanisms proposed in the upper level to detailed applications. Issues such as ontology building, importing

from and exporting to the system, and how to generate agents to take different roles in a MAS are thoroughly investigated in Chapter 6. The bottom level extends the above work to tackle ontology evolution in its life cycle. Chapter 7 presents the ontology refinement mechanisms for tackling ontology evolution from a MAS's perspective. Finally, Chapter 8 consists of conclusions and future work. An overview of each chapter is as follows:

- Chapter 2, *literature review and requirements analysis*, briefly presents the motivation for ontology management. A brief survey of existing systems/tools and approaches is provided. Their advantages and disadvantages are summarised. Based on these, the requirements analysis is thoroughly discussed.

- Chapter 3, *agent-based framework for ontology management*, argues that an agent-based approach is well suited to tackling ontology management flexibly. The analysis and design of a general agent-based framework are discussed. Some material in this chapter is also reported in [73].

- Chapter 4, *agent-based ontology mapping*, discusses mechanisms of ontology mapping under the framework proposed in Chapter 3. Mapping is based on two kinds of semantic relations. A similarity thesaurus dictionary also plays an important role in mapping by providing synonym information. This dictionary is increasing in size over time. Based on mapping results, querying (`query`) in a distributed ontology environment is detailed. We argue that a query can always be made, given that there exists a `weakly connected graph` whenever mappings between pairs of ontologies have been performed. The work is also reported in [71, 75].

- Chapter 5, *agent-based ontology integration*, discusses mechanisms of ontology integration under the framework proposed in Chapter 3 and the mapping results achieved in Chapter 4. It is believed that a global view at an abstract level is very important in ontology-based applications where achieving an objective goal may require knowledge and capabilities beyond individual. Ontology reuse is taken into account in practice. `Consistency checking` of integrated ontology is discussed. The work is also reported in [76].

- Chapter 6, *prototyping and evaluation*, elaborates how to implement the proposed framework and mechanisms using the JADE agent platform for *ontology mapping* and *integration*. The prototype is called JOMI (Jade-based Ontology Mapping and Integration). JADE ACL communicative acts, especially semantics of ACL are introduced. Then a three-layer ontology model based on our ontology building experience and a relatively profound analysis of the current development of ontologies is presented. Detailed agent designs and implementations are then discussed. A case study is given to show the major functionalities of JOMI. Based on the implementation, a discussion is presented followed by the evaluation of the framework. The material in this chapter is also reported in [77].

- Chapter 7, *agent-based ontology refinement*, discusses mechanisms of ontology refinement under the framework proposed in Chapter 3. Agent negotiation mechanisms are applied to reflect ontology evolution over time. A strategic-negotiation model is used to describe and catch ontology changes. Process algebra provides strong theoretical background to abstract information exchange regardless of whatever semantic relations exist. The work is reported in [74, 151].

- Chapter 8, *conclusions and future work*, summarises the new ideas discussed in this thesis. Principle contributions are presented, followed by some interesting directions for future research.

Chapter 2

Literature Review and Requirements Analysis

Ontologies are becoming increasingly important as Web-based applications increase. They are useful (and arguably necessary) in supporting at least navigation, browsing, user-expectation setting, and parametric search. They facilitate interoperability between heterogeneous systems involved in commonly interested domain applications by providing a shared understanding of domain problems and a formalisation that makes ontologies machine-processable. This is why ontology research is attracting increasing attention from both academia and industry. Although much work in the field has been done, this mainly treat ontologies hosting environments as static by deploying traditional software engineering paradigms. Actually, ontology management, as an evolving process, is more complex than previously expected. As briefly discussed in Section 1.2, there exist some key issues regarding ontology management in the context of the Web. To reveal why a flexible approach is required in this field, classic approaches associated with ontology interoperability in database[1], AI and knowledge management communities are reviewed in Section 2.1.

While heterogeneous ontologies (including different representations and different sources) are increasing on the Web, in particular, whenever ontology semantics

[1]Much work [32, 83, 123] has been done in database community to facilitate retrieving information, matching of schema, and identifying similarity among schema towards interoperability in heterogeneous databases.

is involved, traditional distance-like similarity measurement is unsatisfactory for describing or illustrating ontology semantics at run-time, which contributes heavily towards ontology interoperability. By saying this, we do not mean that syntactic clues have nothing to do with estimation of semantic similarities. On the contrary, we utilise the knowledge that semantic similarity is strongly and positively correlated with syntactic similarity and relies on it whenever semantic similarity is required.

Intelligent agents and multi-agent systems represent a new paradigm of analysing, designing, and implementing complex systems such as ontology systems towards ontology interoperability. The agent-based perspective opens up prospects to considerably improve the way in which people conceptualise and implement many types of software. The current practice of agent-based software engineering is discussed in Section 2.2.

The trend of heterogeneous ontologies to increase significantly requires that any ontology management towards ontology interoperability must take their heterogeneity and dynamic features into consideration. Intelligent agent technology holds great promise for solving such problems. The corresponding requirements analysis is presented in Section 2.3. Finally, Section 2.4 summarises this chapter.

2.1 Typical Approaches to Ontology Interoperability

Attention has been paid and will continue to be paid to ontology and ontology related research fields from different perspectives. The first one is similarity measurement by deploying techniques from other research studies. For example, fuzzy logics from AI is utilised in some cases to cope with fuzzy information. The second one is systems or tools which utilise similarity definitions to provide some helpful assistance to human experts. We review major existing systems, tools, or approaches for the purposes of either mapping or integration relevant to our work. For previous works which did not claim their purposes, or can be assigned to either the mapping group or the integration group, we put them where they belong (with emphasis given to the primary criterion) according to our criteria. Please note our

emphasis varies from chapter to chapter even on the same work, given that we have different foci in each chapter. In other words, some work can appear in either mapping or integration related work, but not at the same time. Detailed criteria are given in corresponding Chapters 4 and 5, respectively.

Ontology mapping is not the goal in itself. It is the foundation of further ontology management (e.g. ontology integration). Similarity measurement is becoming a core issue whenever mapping is concerned. In the following, we first present related work in solving similarity of two elements (an element might be instantiated as an attribute, a concept, or an instance in practice) followed by the most referenced existing mapping and integration systems and tools.

2.1.1 Similarity Measurement

Specific techniques or knowledge in measuring similarities are utilised in the vast majority of existing solutions. For example, several studies have suggested the use of graph theory techniques to identify similarity among schemata, represented in the form of either a tree or a graph [16, 104, 129]. The distance-based (e.g. edit distance [70]) approach from information retrieval techniques is adopted in the work of Maedche et al. [86] to overcome the inadequacy of exact "keyword-based" matching. This approach is based on the presumption that semantic similarity is strongly correlated with syntactic similarity (e.g. similar attribute names representing semantic similarity). The DIKE [116] system computes the similarity between two representation elements based on the similarity of characteristics of the elements and the similarity of related elements. Protégé[2] (*http://protege.stanford.edu/*) relies on the assumption of the PROMPT [106] algorithm. PROMPT is a semi-automatic matching algorithm that provides suggestions in guiding experts through ontology matching and alignment.

Some approaches, for instance, in Doan's GLUE [21], machine learning algorithms are deployed to create a mapping between two attributes based on the

[2]Protégé is a free, open source ontology editor and knowledge-base framework. Protégé is based on Java. It is extensible, and provides a foundation for customised knowledge-based applications. Hence, it is supported by a strong community of developers and academic, government and corporate users, who are using Protégé for knowledge solutions in areas as diverse as biomedicine, intelligence gathering, and corporate modelling.

similarity among their associated values. Some form of Bayesian classifier [24, 68] is utilised, where mappings are based on classifications with the greatest posterior probability, given data samples.

A hybrid of matching techniques is proposed to provide a more reasonable measurement to leverage participants' strengths. Under this approach, a weighted sum of similarity measurement formula has been given to specify the similarity of any two objects (or elements, concepts; these terms are used interchangeably). Cupid [83] and OntoBuilder [101] are two models that support the hybrid approaches.

Those techniques are the basis of building and supporting application systems, especially in which similarity measures are essential. Popular systems and tools in this field are presented next.

2.1.2 Systems and Tools

2.1.2.1 Ontology Mapping Systems and Tools

Ontology mapping or sharing of ontology, is not a solved problem. It has been acknowledged and researched by the knowledge engineering community for years. There are many such kind of systems and tools, more or less related to that of database communities (as long as database schemas can be considered a particular kind of lightweight ontology) which claim to have provided some sort of solutions in resolving semantic heterogeneity. A survey by Rahm et al. [123] provides more details about schema matching, a basic problem in database application domains. Some commonly referenced ontology mapping systems and tools are listed as follows.

- CAIMAN [66]: This is a system which uses machine-learning for ontology mapping. It is used for text classification which measures the probability that two concepts are corresponding.

- Chimaera (*http://www.ksl.stanford.edu/software/chimaera/*) [90, 91]: This is a software system that supports users in creating and maintaining distributed ontologies on the Web. Two major functions it supports are merging multiple ontologies together and diagnosing individual or multiple ontologies. It supports users in such tasks as loading knowledge bases in differing formats,

reorganising taxonomies, resolving name conflicts, browsing ontologies, editing terms, etc. Chimaera is built on a platform that handles any OKBC compliant [19] representation system. It contains a simple editing environment in the tools. In addition, it also allows the user to choose the level of vigor with which it suggests merging candidates. It is the analysis capability that allows users to run a diagnostic suite of tests selectively or as a whole.

- Clio [94, 95, 122]: This is a system for managing data transformation and integration under development at IBM Almaden. Clio's engines, consisting of *Schema, Correspondence* and *Mapping Engine*, work together with an internal knowledge base, as well as the user to produce the desired mapping.

- ConcepTool [20]: This is a system which adopts a description logic approach to formalise a class-centred, enhanced entity relationship model. ConcepTool aims to facilitate knowledge sharing. It is an interactive analysis tool that guides the analyst in aligning two ontologies. Linguistic and heuristic inferences are also used to compare attributes of concepts in two models to resolve conflicts between overlapping concepts.

- FCA-Merge [133]: This is a method of ontology merging. It follows a bottom-up approach offering a global structural description of the merging process. For the source ontologies, it processes by means of three steps to generate the new ontology. First, the 'Linguistic Analysis and Context Generation' is used to extract instances from a given set of domain-specific text documents by applying natural language processing techniques. Based on the extracted instances, in the next step, 'Generating the Pruned Concept Lattice', mathematical techniques from Formal Concept Analysis [34, 145] are used to derive a lattice of concepts as a structural result of FCA-Merge. Finally, in the step of 'Generating the New Ontology from the Concept Lattice', the produced result is explored and transformed to the merged ontology by the ontology engineer.

- GLUE [23]: This is a system which employs machine learning technologies to semi-automatically create mappings between heterogeneous ontologies, where

an ontology is treated as a taxonomy of concepts. GLUE focuses on finding one-to-one mappings between concepts by using probabilistic definitions of several practical similarity measures in a taxonomy.

- IF-MAP [127, 57]: This is an automatic method for ontology mapping based on the Barwise-Seligman theory of information flow [5]. In IF-MAP, mappings have a strong relationship with ontology merging. IF-MAP is partially inspired by FCA-merge. Taxonomy is the main structure of an ontology.

- ITTalks [121]: This presents a mapping mechanism which uses text classification techniques as part of its web-based system for automatic notification of information technology talks (ITTalks). The text classification technique is used to generate scores between concepts in two ontologies based on their associated exemplar documents. Bayesian subsumption is then employed for subsumption checking.

- MAFRA (MApping FRAmework for distributed ontologies) [85, 131]: This is an ontology management tool developed by the University of Karlsruhe. MAFRA's framework defined for mapping distributed ontologies on the Semantic Web is based on the idea that complex mappings and reasoning about those mappings are the best approach in a decentralised environment like the Web. Semantic bridges are defined in MAFRA's framework.

- OntoMap [60]: This provides a mapping model for upper-level ontologies, and a few of the most popular ones are encoded in it. It includes a number of alternative viewers: HTML, DHTML, and a standalone GUI application. However OntoMap focuses on the evaluation and comparison of the ontologies, which are encoded into OntoMapO, rather than on providing ontology management services.

- PROMPT [106, 108]: This takes into account different features in the source ontologies to make suggestions and to look for conflicts. It has been implemented in a plug-in for Protégé-2000 (*http://protege.stanford.edu*). User's intervention is desired from time to time.

Naturally, defining the mappings between pairs of ontologies, either a general approach or a specific one, is not a goal in itself. The mapping results are used for various integration tasks in applications. Below are some known ontology integration systems and tools.

2.1.2.2 Ontology Integrated Systems and Tools

Like ontology mapping, previous solutions to integration are also related to integration of heterogeneous databases conducted in the 1980s and early 1990s [6, 128]. Some of the most popular ontology integration systems and tools are listed below.

- Ariadne (*http://www.isi.edu/info-agents/ariadne/index.html*): The Ariadne project aims at the development of technology and tools for rapidly constructing intelligent agents to extract, query, and integrate data from Web sources. It is being developed in the Information Sciences Institute (ISI) at the University of Southern California, with support from DARPA (*http://www.daml.org/*). This project tries to improve SIMS (Service and Information Management for decision Systems) [1], allowing the management of semi-structured sources such as Web pages. The application can access different sources via an *Ariadne information mediator*, which uses ontologies codified in LOOM[3] [82]. This mediator has mappings between the ontologies and the information sources.

- InfoSleuth [29, 105]: This is a system which integrates agent technology, domain ontologies, brokerage, and Internet computing, in support of mediated interoperation of data and services in a dynamic and open environment. Different types of agents have been developed in the system. The agent communication language is KQML.

- MOMIS (*http://sparc20.dsi.unimo.it/Momis/*) (Mediator envirOnment for Multiple Information Sources) [9, 10, 8]: This is an approach to the integration of heterogeneous data sources using a global ontology, which is the

[3]Loom is a representative for description logic based language, which can define concepts in terms of descriptions that specify the properties that objects must satisfy in order to belong to that concept.

result of a merge of the local data schemas. The ontology merging task is provided by the ARTEMIS tool [17]. The ARTEMIS tool provides the support in the matching task by determining the affinity between terms in the ontologies.

- OBSERVER (*http://siul02.si.ehu.es/ jirgbdat/OBSERVER/*) (Ontology Based System Enhanced with Relationships for Vocabulary hEterogeneity Resolution) [92]: This is a system which aims to overcome problems with heterogeneity between distributed data repositories by using component ontologies and the explicit relationships between these components. In OBSERVER, When a user poses a query, the first answer is given using just one ontology. If the user is not satisfied, he or she can choose the another to extend the result. The system provides the estimated loss of information, given that the correspondence between the concepts of an ontology with the concepts of the other one is not exact. That is, the system estimates how many items are not proper answers, and how many items should appear, but do not yet. A query can be expanded over multiple ontologies by rewriting the query from the local ontology to the component ontologies. However, the system has no intention of creating the mapping between ontologies. Automatically integrating ontologies during the query process is based on the inter-ontology relationships from the Inter-ontology Relationship Manager (IRM).

- ONION (ONtology compositION) [33, 97, 98]: It is an architecture based on a sound formalism to support a scalable framework for ontology integration that uses a graph-oriented model for the representation of the ontologies. Articulation rules are established to enable knowledge interoperability.

- PICSEL (*http://www.lri.fr/p̃icsel/*) [30, 31]: It is an information integration system over sources that are distributed and possibly heterogeneous. It has been developed by LRI (France). The approach which has been chosen is to define an information server as a knowledge-based mediator between users and several existing information sources relative to the same application domain. The mediator gives its users the illusion of a centralised and homogeneous information system. It allows them to ask domain-level queries and

takes charge of accessing the relevant sources in order to obtain the answers to the queries. Description Logic with function-free Horn rules are utilised in PICSEL.

- RDFT (RDF Transformation): Omelayenko and Fensel [113, 115] assume product catalogs from different organisations specified in XML documents. The approach to the integration of product catalogs is called two-layered because the product information itself is represented in XML, whereas the transformation between different representations is done in RDF. They propose a mapping meta-ontology for describing the transformation between RDF documents. The mapping meta-ontology, called RDFT, is specified using RDF Schema and is used to describe the mapping between two RDFS ontologies. Moreover, Omelayenko [114] describes a technique for discovering semantic correspondence between different product classification schemes based on a Naive-Bayes classifier.

2.1.2.3 Comparison of Systems and Tools

The comparison of the above systems and tools is based on the criteria presented below, which is set up in such a way that it enables us to catch the state of the art (e.g. advantages and disadvantages in semantic mapping and integration) of existing systems according to flexibility and applicability towards ontology interoperability in the Web context. We investigate two major characteristics as outlined below.

- **Semantic Similarity**: A vast majority of work in estimating semantic similarity relies heavily on similarity of syntactic clues. Similarity measurement plays a vital role in any mapping. And there is no exception in ontology mapping, which leads to semantic mapping intuitively.

- **Interaction** between ontologies: Predefined or manual creation of Semantic mappings has long been known to be unable to meet dynamic features of technologies such as the Internet and the Web. In particular, manual mapping is regarded as being extremely laborious and error-prone. As discussed in Section 1.2, the development of solutions to be able to respond promptly to

changes in its environment becomes truly crucial to building a broad range of ontology-based applications. In response to the requirement, solutions should be robust and flexible across domains.

Table 2.1 summarises the above two characteristics with respect to the listed systems and tools in this section. Notations used in Table 2.1[4] are presented below.
(i) "Yes" if a corresponding item in the table is supported;
(ii) "No" if it is not; and
(iii) "N/A" if no information is available with respect to a specific feature.

To summarise, the systems and tools for ontology mapping use the following features in ontology definitions (to a varying extent) to measure similarities between pairs of ontologies:

- concept denotation names;
- class hierarchical structures; and
- property definitions (attributes)

Note that the syntactic similarity is positively and strongly correlated with the semantic similarity. The above syntactic features will be utilised as foundations of semantic similarity measurement in our thesis.

Previous works listed in Table 2.1 have run into the problem of semantic heterogeneity. Although some specific techniques (e.g. machine leaning, heuristics) have been utilised to speed up the process or leverage workloads across the systems, they have the following limitations:

- Such systems pay less attention to run-time operations (e.g. lack of *interaction*), but only focus on design-time manipulations. This is not applicable when scalability and extensibility are involved;
- They suffer from flexibility if ontology/data interoperability is concerned in the context of the Web.

[4]The contents of the table represent the present situations and may change because of the evolution of listed systems/tools. This applies to the following tables.

Table 2.1: Comparison of systems and tools

System/Tool	**Semantic Similarity**	**Interaction**	**Remarks**
Mapping systems			
CAIMAN	Yes	N/A	measuring the probability of two corresponding concepts
Chimaera	N/A	N/A	resolving name conflict and aligning the taxonomy
Clio	Yes (to some extent) with additional information from users	mapping engine may update the mapping	query fine-grained mapping (e.g. SQL or XQuery); user intervention only when absolutely necessary
ConcepTool	Yes	N/A	six step methodologies are provided
FCA-Merge	Yes	N/A	bottom-up merging of ontology
GLUE	Yes	N/A	using machine learning approach
ITTalks	Yes	N/A	text classification techniques
IF-MAP	No	N/A	drawing from work of mathematical theory of information flow
MAFRA	Yes	N/A	implemented with KAON by providing bridging ontology
OntoMap	Yes	N/A	meta-ontology (e.g. OntoMapO); mapping between EuroWordnet Top (ontology) and other ontologies
PROMPT	Yes	N/A	semi-automatic guided ontology merging

Integration systems			
Ariadne	Yes	N/A	a natural extension of SIMS; users are assumed to be familiar with the application domain, and to use standard terminology to compose the LOOM query
InfoSleuth	Yes	Yes	query language is a variant of SQL
MOMIS	Yes	N/A	identify all possible relationships between a set of ontologies; integrating them in a global ontology
OBSERVER	Yes (provided by IRM)	N/A	component-based approach
ONION	Yes	N/A	ontology mapping based on the graph mapping
PICSEL	Yes	N/A	a few algorithms for reformulating queries expressed from domain relations into sources relations
RDFT	Yes	N/A	product catalogs specified in XML documents

Table 2.2: Comparison of the ontology integration systems and tools

System/Tool	Automatic or Semi-automatic	Defining Ontology Mappings	Extensible Ontology Integration Architecture
Ariadne	"+"	"+"	"++"
InfoSleuth	"++"	"+"	"+++"
MOMIS	"++"	"+"	"+"
OBSERVER	"++"	"++"	"+++"
ONION	"++"	"+"	"++"
PICSEL	"+"	"+"	"+"
RDFT	"+"	"+"	"+"

In addition, the following criteria are used to evaluate introduced integration systems and tools. Some of the specific challenges in ontology integration that must be addressed in the near future are listed below. Refer to [107] for more details of challenges in this field.

- Finding similarities and differences between ontologies in an automatic or semi-automatic way;
- Defining mappings between ontologies;
- Developing an ontology integration architecture which is possible to be capable of extension.

In Table 2.2, notation "+" is for weak support of a particular feature, "++" for reasonable support, while "+++" for strong support. Table 2.2 summarises the above characteristics with respect to the introduced integration systems and tools. It is clear that agent- or component-based frameworks provide stronger support for extensibility, while the foundation on which many scalable applications are built is the possibility of finding similarity in an automatic or semi-automatic way.

Nevertheless, applying any techniques to the domain of ontology management towards ontology interoperability need to consider any challenges (discussed in Section 1.2) raised by the emerging Web or Semantic Web. In the following, we provide a detailed discussion of agent-based software engineering, which is deemed

the most suitable paradigm for exploiting ontology interoperability in the context of the Web.

2.2 Agent-based Software Engineering

Designing and building high quality industrial-strength software is difficult. Indeed, it has been claimed that such development systems are among the most complex construction tasks undertaken by humans. Nowadays, agents are being advocated as a next generation model for engineering complex and distributed systems [147, 54]. Some researchers in this field have given a qualitative analysis to provide the justification for precisely why agent-based approaches are well suited to engineering complex software systems [50, 51, 55, 54, 52]. Significant advantages are shown clearly as follows:

- Agent-based approaches can significantly enhance software ability to model, design and build complex and distributed software systems;
- Agent-based approaches will succeed as a mainstream software engineering paradigm in so far as long as they are suited for designing and building complex systems.

Agent technology ushers in a new trend in software engineering. Agent-oriented software engineering represents a promising point of departure for software engineering through being a new programming paradigm[5]. The development of agent-based applications implies new programming abstractions and methodologies. There are some methodologies [47, 137, 148, 149, 153, 154] which provide clear guidelines for developing multi-agent systems.

Because of the nature of agent technology, it is likely that the successful adoption of agent technology in the environment, in which the ability to respond dynamically to changing circumstances is required, will have a profound, long-term impact. It is expected that agent technology will affect the competitiveness and

[5]A wide range of software engineering paradigms have been devised. In addition to agent-oriented software engineering, there are procedural programming, structured programming, declarative programming, object-oriented programming, design patterns, application frameworks and component-ware.

viability of IT industries as well as the way in which future computer systems are conceptualised and implemented [81]. Agent-based software engineering has proven very popular in coping with complex system design and construction. It plays a key role in constructing scalable systems. Key observations based on past experience [12, 53, 79, 80, 149, 152] clearly suggest the use of agent technologies for the full benefits of realising flexibility and scalability of the systems to be achieved. And this is especially true in the context of environments that the emerging Web has been envisaged recently.

2.3 Requirements Analysis

It is not uncommon for ontologies to be developed by several persons and continue to evolve over time, because of changes in the real world, adaptations to different tasks, or alignments to other ontologies. To prevent such changes from invalidating existing usage, a change management methodology is needed. It is known that ontologies either need to be integrated [117], which means that they are composed into one new ontology, or they can be kept separate. In both cases, the ontologies have to be matched, which means that they have to be brought into mutual agreement. The problems that underlie the difficulties in ontology mapping and integration are the mismatches that may exist between separate ontologies across organisations. Ontologies can differ at the language level, which can mean that they are represented in a different syntax, or that the expressiveness of the ontology language is dissimilar. Ontologies also can have mismatches at the model level, for example, in the paradigm, or modelling style [61]. Moreover, reusing existing ontologies is one of many complex tasks which requires considerable effort [140]. Besides, when the number of different ontologies is increasing, the task of maintaining and reorganising ontologies to secure the successful ontology interoperability is challenging.

Before reaching a real understanding of semantics on the Web (e.g. allowing machines to infer implicit knowledge), ontology mapping towards ontology interoperability, regardless of different models and languages used in conceptualising and

representations, must be resolved. To develop appropriate architectures and mechanisms to automate and improve existing systems or tools is of key importance. Furthermore, dynamic capability is essential our work.

Based on the literature review of the state of the art in ontology management, it is clear that existing methodologies and tools, which mainly treat ontology hosting environments statically rather than dynamically in a distributed fashion, lack of flexibility and extensibility in dealing with ontology management leads to these systems operate under limited operational conditions. Some causes of this problem are listed as follows:

(1) Most architectures used to develop these systems have failed to anticipate future requirements of ontology management. In fact, they are '*functional*' systems instead of '*reactive*' ones to exhibit intelligent behaviours. Unfortunately, the desired architectures for handling ontology management are not '*functional*' in the sense of the complex interactions required, and communications are exchanged in terms of high-level protocols and languages.

(2) Considerable prior knowledge is required to enable these systems to operate correctly in ontology management. However we know that knowledge provided in advance does not always hold in a changing environment. This means that these systems are brittle in the sense that human intervention is required to compensate for even slight shifts of the environment. Obviously, they are unable to cope with the requirements of ontology management in an open environment.

(3) Ontologies in these systems or tools are the only objects. In other words, these systems or tools manage ontologies from the knowledge management perspective rather than from the knowledge deployment perspective. However, it is known that ontologies are used to provide interoperability among ontology-based applications. And this is eventually used for reasoning purposes.

In short, existing systems or tools are observed to lack the capabilities to deal with highly changing environments with their underlying developing motivations.

They have either no theoretical backgrounds, or no clear architecture, or no flexibility. Agents in a MAS are autonomous and flexible enough to interact with other complex interactions in order to achieve defined objectives. the advantages of deploying an agent-based approach in ontology management are three fold:

(1) ability to support changes in the environment substantially, given that they are rational and have the capabilities to perceive any change and adapt to the change accordingly.

(2) ability to exhibit goal-directed behaviour to satisfy their design objectives; and

(3) ability to exhibit intelligent behaviours by interacting with other agents to achieve designed goals in the environment. For this reason agents are being advocated as a next generation model for engineering complex and distributed systems [147, 54].

With the above objectives achieved, the problems discussed above are expected to be addressed. Moreover, with the best known FIPA (the standards organisation for agents and multi-agent systems) specifications (*http://www.fipa.org/*) being officially accepted by the IEEE as its eleventh standards committee on 8 June 2005, it is expected that research into agent technology will become more attractive with increasing attention around the world.

2.4 Summary

In this chapter, the literature in relation to ontology interoperability has been widely reviewed. The major problems in current approaches, such as lack of flexibility and extensibility, have been analysed comprehensively. Based on the literature review and requirements analysis, Agent technology, which is able to support responding dynamically to changing circumstances, to exhibit goal-directed behaviour, as well as to exhibit intelligent behaviours is suggested for dealing with existing problems.

In response to the impediments mentioned in this chapter, a general agent-based framework to facilitate ontology management in a dynamic environment is needed. The framework is discussed in Chapter 3. With the support of the framework,

major ontology managements and relevant issues are addressed in Chapters 4, 5, 6 and 7, respectively.

Chapter 3

Agent-based Framework for Ontology Management

The limitations of current approaches, such as lack of flexibility and extensibility, have seriously blocked ontology development on the Web since existing frameworks failed to cope with ontology management dynamically. As stated in Chapter 2, agent-oriented software engineering is a promising new paradigm suitable for complex system development where responding dynamically to changing environment is required. In this chapter, an innovative agent-based framework is presented to facilitate heterogeneous ontology management with a focus on the run-time rather than only design-time phase. It aims to provide a flexible general framework to cope with distributed ontologies on the Web.

It is not surprising that there are some terms used inconsistently when people talk about ontology management and even ontology itself. To this end, ontology definition and terminology used throughout this thesis will be introduced in Section 3.1. Some distinguishing characteristics of agent-based approaches are addressed in Section 3.2. Bearing this in mind, in Section 3.3, a general framework is given with a brief introduction to different kinds of agents. Section 3.4 is a comparison, in which our agent-based framework is compared with well-known frameworks. Section 3.5 summarises this chapter.

3.1 Ontology Terminology

In this thesis, we follow a well known ontology definition by Gruber [38]. *An ontology is an explicit specification of a conceptualisation.* One of the reasons for representing ontologies is the ability to reason about them. Within the ontology definition, an ontology $\mathcal{O}$ with a specific domain model $\mathcal{T}$ is defined as: $\mathcal{O} ::= < \Sigma, \Psi >$. Thus a conceptualisation Σ is a pair of $< \mathcal{C}, \mathcal{R} >$, where $\mathcal{C}$ represents a set of concepts, and $\mathcal{R}$ stands for a set of relations over these concepts. A specification is a pair of $< \Sigma, \Psi >$ to show that Σ satisfies axioms Ψ. There are many representations and languages[1] available for encoding an ontology, however, to establish the notation of the ontology used in a MAS internally for the task of ontology mapping, an `Entity-Relation` (E-R) data model is considered for encoding an ontology, where concepts are regarded as classes. A typical concept class will have an identifier that distinguishes it from others, and a set of attributes that describes the properties of the concept class. Hence it is feasible to compare two concepts by looking at the identifier as well as the attributes. We assume that $\mathcal{O}_i$ and $\mathcal{O}_j$ are in the same domain ($i, j \in \mathsf{N}$, N natural numbers). c_i (c_i is a concept with inherent attributes, $c_i \in \mathcal{C}(\mathcal{O}_i)$) and c_j ($c_j \in \mathcal{C}(\mathcal{O}_j)$) are two different concepts.
Definition (*Equivalent*): Two concepts are semantically equivalent, i.e. if $\exists c_i, c_j$, such that $c_i \sim c_j$. Namely, these two concepts:

- have the same denotation names (e.g. labels);
- are synonyms; or
- their attributes are the same.

Definition (*Inclusive*): Two concepts are semantically inclusive, if $\exists c_i, c_j$, such that $c_i \leq c_j$ (e.g. c_i is a kind of c_j) or $c_i \geq c_j$ (e.g. c_j is a kind of c_i). Namely, the attributes of one concept are also the attributes of the other.
Definition (*Ontology Mapping*): A mapping $\Re$ between two ontologies $\mathcal{O}_i$ and $\mathcal{O}_j$ exists, if $\exists c_i, c_j$, such that $\Re(c_i, c_j) \in \{\sim, \leq, \geq\}$. In some cases, we

[1]There are many formal languages, such as RDF(s) (*http://www.w3.org/RDF/*), OIL (*http://www.ontoknowledge.org/oil/*), DAML+OIL (*http://www.w3.org/TR/daml+oil-reference*), and OWL (*http://www.w3.org/TR/owl-guide/*) for ontology representation. Intuitively, these languages differ in their terminologies and expressiveness. The ontologies that they model essentially share the same features.

can only achieve partial mapping (mappings between some pairs of concepts in two ontologies) instead of full mapping (mappings between pairs of concepts and relations in two ontologies).

Definition (*Ontology Integration*): Reusing available source ontologies within a range to build a new ontology which serves at a higher level in the application than that of various ontologies in ontology libraries. It is associated with semantic integration. Different levels of integration can be distinguished.

For the purpose of ontology integration, we need to consider the consistency problem of an integrated ontology. It is obvious that the E-R date model of an ontology and the defined semantic relations between concepts allow us to check the consistency of a newly derived ontology. The ontology disjoint and ontology consistency are defined as follows.

Definition (*Disjoint*): Two concepts are disjoint, if $\exists c_i, c_j$, such that $c_i \cap c_j = \Phi$. Namely, there is no common attribute between them.

Definition (*Consistent*): An ontology is consistent, if $\forall c_i^k, c_i^n, c_i^m$ $(c_i^k,\ c_i^n, c_i^m \in \mathcal{C}(\mathcal{O}_i)$, and $c_i^k \neq c_i^n \neq c_i^m$ $(k, n, m \in \mathsf{N})$, $c_i^k \leq c_i^n$ and $c_i^n \cap c_i^m = \Phi$, such that $c_i^k \not\leq c_i^m$. Namely no sub-concept (a sub-concept c_j has something with a particular concept c_i such that $c_i \geq c_j$) of a particular concept is also a sub-concept of another concept where these two concepts are disjoint.

Definition (*Ontology Refinement*): An ontology evolves throughout its life cycle. A successful ontology management is not possible without reflecting ontology change. Ontology refinement is such a mechanism in exploiting ontology evolution in order to seamlessly incorporate evolving ontology in ontology management.

Definition (*Ontology Interoperability*): This is a full realisation of a vision in which at least two properties need to be present: (1) any two ontologies $\mathcal{O}_i$ and $\mathcal{O}_j$ are interoperable, i.e. $\mathcal{O}_i \longleftrightarrow \mathcal{O}_j$, if $\exists \Re$ (direct mapping) between these two ontologies, or $\exists \Re', \Re'', \ldots,$ (indirect mappings) which lead to the mapping of these two ontologies; and (2) the ontologies comply with the above defined ontology mapping, integration and refinement.

3.2 Agent-based Approach for Ontology Management towards Interoperability

Agent technologies represent a trend in software development. They are fundamental to the realisation of next generation computing [81]. They have the potential to significantly improve current practice in software engineering and to extend the range of applications that can feasibly be tackled [54]. Because of the nature of agent technology, it is likely that the successful adoption of agent technology in the areas of distributed, dynamic and heterogeneous environments will have a profound, long-term impact.

According to Russell et al. [126]: "an agent is any entity that can be viewed as perceiving its environment through sensors and acting upon its environment through effectors". Also Jennings [52] and Wooldridge [147] give the following definition: "an agent is an encapsulated computer system that is situated in some environment and that is capable of flexible, autonomous action in that environment in order to meet its design objectives." In more details [55], agents are: (i) clearly identifiable problem solving entities with well-defined boundaries and interfaces; (ii) situated (embedded) in a particular environment in that they receive inputs related to the state of their environment through sensors and they act on the environment through effectors; (iii) designed to fulfill a specific role in that they have particular objectives to achieve and have particular problem solving capabilities (services) that they can bring to bear to this end; (iv) autonomous in that they have control both over their internal state and over their own behaviour; and (v) capable of exhibiting flexible problem solving behaviour in pursuit of their design objectives in that they need to be both reactive (able to respond in a timely fashion to changes that occur in their environment) and proactive (able to opportunistically adopt goals and take the initiative). By the above description, the agent is essentially similar to the traditional AI systems where it is human experts who observe changes in the environments and report to the systems. Autonomous agents, on the other hand, observe the changing environment and respond promptly to it by performing appropriate actions. Agents in a MAS are assumed to be autonomous and flexible enough to take actions corresponding to changes in the environments. Wooldridge

and Jennings [146] suggest the following characteristics in terms of flexibility:

- **Reactivity**: Agents are able to perceive their environment, and respond in a timely fashion to changes that occur in it in order to satisfy their design objectives.
- **Proactiveness**: Agents are able to exhibit goal-directed behaviour by *taking the initiative* in order to satisfy their design objectives.
- **Social ability**: Agents are capable of interacting with other agents (and possibly humans) in order to satisfy their design objectives.

Whatever the nature of the social ability, there are two points that qualitatively differentiate agent interactions from those that occur in other computational models [28].

- Agent-oriented interactions tend to be more sophisticated than in other contexts, dealing, for example, with notions of cooperation, coordination, and negotiation;
- Agents are flexible problem solvers, operating in an environment over which they have only partial control and observability.

Thus, interactions need to be handled in a similarly flexible manner, and agents need the computational apparatus (e.g. a range of techniques, such as reinforcement learning and mechanism design) to make context-dependent decisions about the nature and scope of their interactions and to initiate (and respond to) interactions that were not foreseen at design time.

Based on the concept of agents, a MAS can be defined as "a loosely coupled network of problem solvers (agents) that work together to make decisions or solve problems that are beyond the individual capabilities or knowledge of each problem solver (agent)" [26]. In other words, a MAS is formed to fulfill a particular task that no single individual can cope with. The characteristics of MAS are [49]:

- each agent has incomplete capabilities for solving the problem (in a whole system), thus each agent has a limited viewpoint;

- there is no global system control;
- data is decentralised; and
- computation is asynchronous.

A MAS is ideally suited to represent problems that have multiple problem solvers (agents). That a MAS's ability to provide robustness and efficiency, allow interoperation of existing legacy systems, and solve problems in which data or control is distributed, makes a MAS perfectly suitable for scalable, autonomous environments. Although MASs provide many potential advantages, they are neither omnipotent nor magic. They also face many difficult challenges [49]. However, they can operate flexibly in an environment, provided that agents commit to their capabilities. It is these capabilities which distinguish a MAS from other forms of software.

In this thesis, terms like *agent*, *intelligent agent*, or *soft agent* are used interchangeably except where otherwise specified. An *agent-based system* is used to refer to a *multi-agent system* where agents of the system are designed and implemented to fulfill principle characteristics of a MAS. Additionally, a vision of MAS is quite similar to the definition of a business process, in which one of the principle characteristics is interaction between participants (e.g. agents) or sub-processes (e.g. MASs). We will not distinguish a *MAS* from a business *process* unless stated otherwise.

3.3 Agent-based Framework for Ontology Management

So far, it is clear that an agent-based approach is ideal because it is exactly what ontology management needed. In this section, a general agent-based framework is proposed, followed by brief introductions to the behaviours of each kind of agent and agent communication.

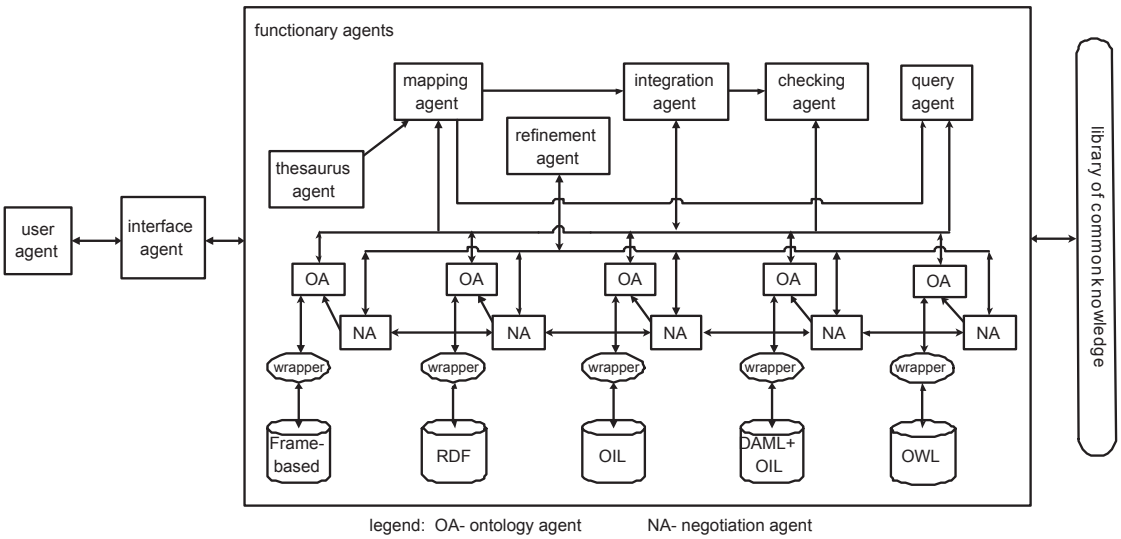

Figure 3.1: Agent-based framework

3.3.1 General Framework

A framework for ontology management must be flexible to allow agents to go on-line or off-line freely. The flexibility of the system becomes evident when applications become more and more complex. As stated in Section 3.2, a MAS is ideally suited to model ontology management on the Web. In this section, an agent-based framework is developed. The overall framework of the agent-based ontology management is shown in Figure 3.1. The behaviours of each kind of agent in this environment are briefly described below. These will be discussed in more detail in the following chapters.

- **User Agent** (*UA*): This is different from other agents developed in this thesis. A **user agent** is designed to only know the `interface agent` (*IA*). This agent interacts with a particular GUI. This includes getting the business scenario from the GUI and passing it on to *IA* to display proper results on the user interface (e.g. GUI) when it receives a return message from *IA*.

- **Interface Agent** (*IA*): This interacts with *UA*, which includes getting the business scenario from a particular GUI and passing it on to the virtual community[2], and then presenting *UA* with expected results. It acts as a broker

[2]a group of partners commonly interested in a certain business scenario

agent in an attempt to help various agents find each other in a distributed environment.

- **Ontology Agent** (*OA*): This acts on behalf of a certain ontology. It behaves properly in a specified agent platform. It is equipped with the functionalities of a certain ontology (e.g. it operates over the ontology structure and a particular intermediate result structure). The main purpose of an *OA* is to perform ontology related tasks which are isolated from external *functionary agents*. The presence of an *OA* allows flexible system organisation.

- **Negotiation Agent** (*NA*): This takes part in negotiation setting in an attempt to obtain ontology evolving information for the corresponding *OA*.

- **Functionary Agent** (*FA*): This is an agent which provides functionalities of thesaurus similarity measure, ontology mapping, ontology integration, ontology refinement, consistency checking and query. These are described below.

  - **Thesaurus Similarity Agent** (*SA*): This is in charge of maintaining thesaurus similarity within a suitable structure. The major tasks of a *SA* are to: (1) append new concepts to the structure; and (2) search a particular concept in the structure.

  - **Mapping Agent** (*MA*): This is shaped to provide linkages to pave the way for the interoperability of various ontologies on the Web. It is the foundation of further ontology management towards interoperability. The major task of a *MA* is to estimate whether two given concepts map each other according to its knowledge.

  - **Integration Agent** (*InA*): This is responsible for ontology integration based on a certain business scenario. The major tasks of an *InA* are to: (1) count the appearance of each specified concept; and (2) filter the unexpected items before sending them to the *OA*.

  - **Refinement Agent** (*RA*): This exploits how to maintain ontology coherence and integrity in an environment of dynamic changes of ontologies that may take place frequently. The major tasks of a *RA* are to: (1)

obtain up-to-date information for a specified concept; and (2) locate any differences between a previous description and the current one.

- **Consistency Checking Agent** (*CA*): This checks the consistency of an ontology. The major tasks of a *CA* are to: (1) check if a particular concept is the subclass of two disjoint concepts; and (2) warn the *UA* if inconsistency exists.
- **Query Agent** (*QA*): This shows how to utilise previous mapping results with distributed ontologies. The main tasks of a *QA* are to: (1) respond to any concept-like query; and (2) dispatch the query to other recongnised agents in the environment if the *RA* is unable to do it by itself; and (3) keep the query path.

- **Library of Common Knowledge**: This library includes atomic roles and primitive concepts which comprise the baseline of knowledge in a particular domain. It may initially be created during run time through agent communications.

- **Ontology Representations (Wrapper)**: Due to the diversity and heterogeneity of ontologies, it is not unusual that different kinds of representation languages coexist. Each ontology representation has a corresponding wrapper to translate a particular ontology representation into a common representation.

In this framework, the fact of different agents work collaboratively is highlighted. For example, *FA*s can effectively access ontologies via *OAs*, whilst a *MA* (one of the *FA*s) may contact a *SA* (another one of the *FA*s) or any other agents which can provide required functionalities. A scenario can be as follows: a user queries the system through the *UA*. Then the *UA* converts the user's query to a suitable representation. After that the *IA* collects relevant information from the previous process and passes it on to the corresponding *FA*s. After a series of operations, a final result is provided graphically to the user via the *UA*. Without doubt, ontologies are excellent foundations for agent communications.

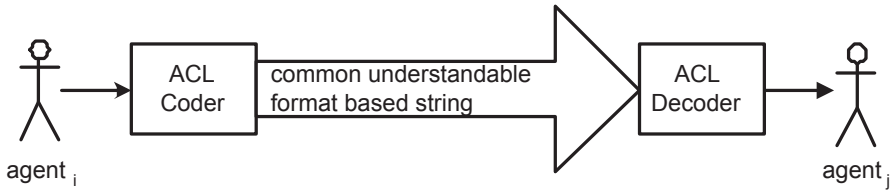

Figure 3.2: Agent communication channel

3.3.2 Agent Communication

A critical issue in the communication among agents is that of *ontological commitments*, i.e. agreement among various agents on the terms for specifying agent context and the context of the information handled by the agents.

In our proposed agent-based framework, there are five major kinds of agents. They are: (1) *OA*s; (2) *NA*s; (3) *FA*s consisting of a *SA*,a*MA*,an *InA*, a *RA*, a *CA*, and a *QA*; (4) an *IA*; and (5) a *UA*. As stated in [52, 147], agents in MASs are autonomous and can engage in flexible and highly interactive actions. In fact, agents are able to perceive their environment and respond in a timely fashion to changes that occur in their environment in order to satisfy their design objectives. They are capable of interacting with other agents (and possibly humans) for the purpose of achieving their design objectives. In other words, agents are highly interactive in the direction of satisfying their objectives. In detail, their abilities of reactivity, proactiveness, and sociability ensure that they can work together with other agents correctly and rationally to achieve the goal beyond their individual capabilities and knowledge.

An agent in this research is defined as: an agent has amongst its mental attitudes the `goal` G and `intention` I. Suppose that in some cases, the agent itself cannot carry out the `intention`. The question then becomes how to work with others by estimating whom to send messages to and which messages should be sent. According to its intention to satisfy the goal, it will send out messages to those agents to bring about an expected outcome or a rational effect on its goal (see Figure 3.2).

In Figure 3.2, $agent_i$ sends out a message to $agent_j$ in order to achieve its designed objective, for example, in a scenario where the *UA* sends a message to the *IA* when it has received a user action. Assuming that if a user is in the

integration interface and then chooses some agents to execute integration, the *UA* passes these agents to the *start-integration* method in the *UA*. Because the *UA* has no idea about the integration module, it will create a message and add specified *OA*s as parameters in the message and send this off to the *IA*. The *IA* then passes the message on to a particular *FA* which claims that it has such capabilities for tackling the corresponding task.

Agent communication is based on Agent Communication Languages (ACLs). ACLs provide agents with a means of exchanging information and knowledge, which is really the essence of all forms of interaction in multi-agent systems. Due to message overheads and the possibility of network congestion, normally the message is parsed into a common understandable format based string. The presence of an ACL Coder and an ACL Decoder at each end of the communication is to convert the message to *String* and vice versa.

At an abstract level, agents work together based on interactions (see Figure 3.2). Let us assume that individual agents cannot carry out the intention by themselves. If agents are behaving rationally, they will contact other agents which will satisfy the intention and thus achieve the end goal. An interaction process (e.g. diagram) is needed as interactions between agents may take place concurrently. After investigating the existing tools, we have chosen AUML (*http://www.auml.org/*) to describe the artifacts of agent interactions. The key idea underlying interactions is to investigate corresponding actions of relevant processes. Agent interactions in chapters 4, 5 and 6 will be depicted by AUML.

3.4 Comparison

One of the principle methods in developing a flexible framework to manage distributed ontologies is to comply to a highly modular standard. There is some research work involved in this topic. InfoSleuth, OBSERVER and MOMIS are three typical architectures.

The InfoSleuth architecture [105] in Figure 3.3 consists of a number of different types of agents. User agents and resource agents are the main agents in the system. User agents request information to fulfill the user's information needs and resource

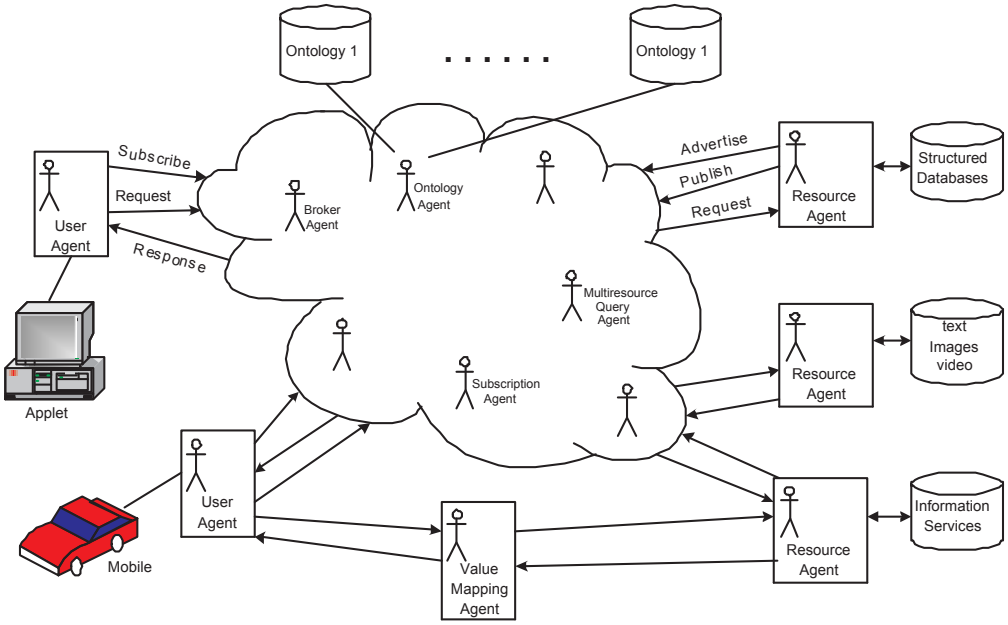

Figure 3.3: InfoSleuth architecture

agents provide that information. Other agents, for example service agents, query and analysis agents, and multi-resource query agents, provide mediation in the system.

The OBSERVER architecture [92], depicted in Figure 3.4, consists of a number of component nodes and the IRM node. A component node contains an Ontology Server which provides for the interaction with the ontologies and the data sources. It uses a repository of mappings to relate the ontologies and the data sources and to be able to translate queries on the ontology to queries on the underlying data sources. The architecture contains one Inter-Ontology Relationship Manager (IRM), which contains a one-to-one mapping between any two component ontologies.

A number of components are used to enable the MOMIS architecture [10]. The architecture in Figure 3.5 shows the main tools used to support the overall architecture:

- A *wrapper* performs the translation of the individual data source into the language (and translates the queries back);
- The *mediator* consists of the *query manager* (QM) and the *global schema*

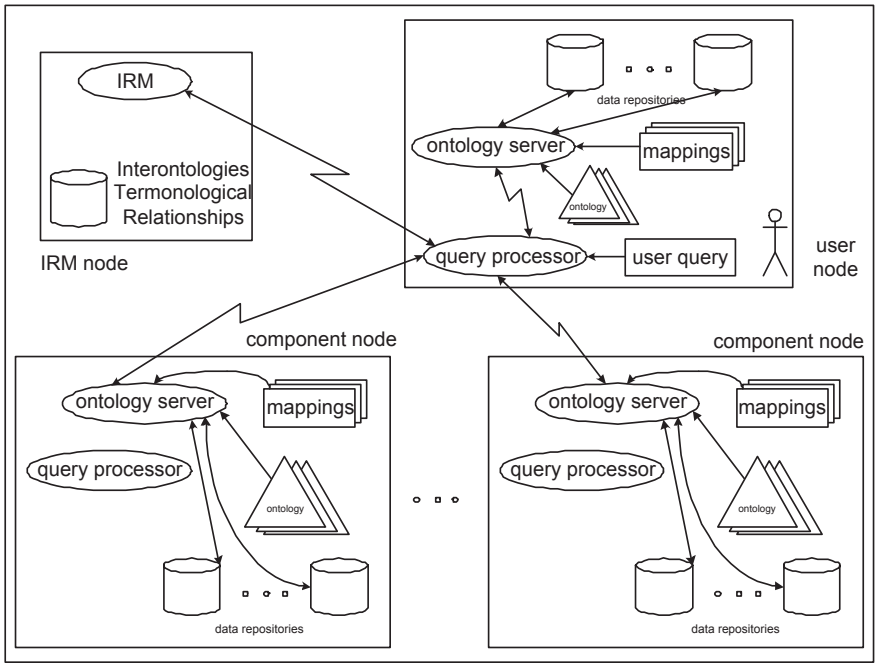

Figure 3.4: OBSERVER architecture

builder (GSB). The QM component breaks up global ODL_{I^3} (an object-oriented language with an underlying description logic language OLCD) queries into sub-queries for the different data sources. Therefore, the GSB is an off-line component, which aids in ontology merging and the QM is a run-time component, which performs the queries.

- The *ARTEMIS* [17] tool environment performs the classification (affinity and synthesis) of classes for the synthesis of the global classes.
- The *ODB-tools engine* (a description logic reasoner) performs schema validation and inferences for the generation of the Common Thesaurus, as well as query optimization for the Query Manager.

Among the above three systems, the InfoSleuth architecture takes a MAS perspective to support construction of complex ontologies from smaller component ontologies. However, a flexible framework, from our perspective (for ontology mapping and integration), should allow creation of ontology mapping as well as integration at run-time. Although InfoSleuth architecture provides service agents

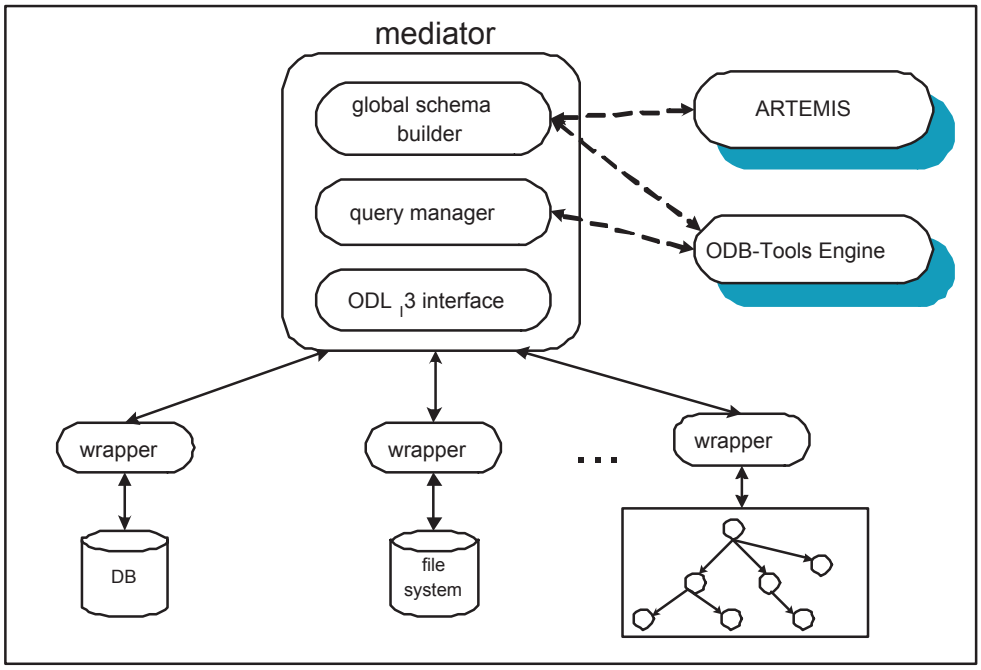

Figure 3.5: MOMIS architecture

such as broker agents to maintain advertised information, automatic ontology mapping is very limited since mapping is explicitly specified among those ontologies. In the OBSERVER architecture, the way to achieve flexible framework design is through a component-based approach. However, its mapping is also limited due to the use of IRM. The MOMIS architecture also provides a way to integrate heterogeneous data sources but by using a global ontology rather than creating ontology mapping dynamically.

In summary, the above architectures have the following limitations:

- They are not designed to automatically create ontology mapping. Instead, mappings are specified for those involved ontologies;
- It is unlikely to add/delete ontologies to/from the systems dynamically (in InfoSleuth, a set of information about the ontology mapping rules are presented). For example, in OBSERVER, an IRM is used to provide translations between the terms among the different component ontologies.

Table 3.1: Comparison of frameworks

Framework	**Broker Agent**	**Ontology**	**Mediation**
InfoSleuth	+++	-	+++ (*service agents*)
MOMIS	-	-	++ (*mediator*)
OBSERVER	-	+++	+ (*query processor*)
Our Framework	+++	+++	+++ (*service agents*)

- Communication languages between components or agents are totally different. Therefore, it is hard to interrelate each one in an integrated tool environment.

Table 3.1 shows comparison of our framework to the above three well-known frameworks. Notations of the table are defined below:

(i) " +" if a corresponding item in the table is supported, with " +++" representing strong support, while " +" indicates weak support;

(ii) " -" otherwise.

Three major characteristics in Table 3.1 are outlined below:

- **Broker Agent**: This enables agents in the system to advertise their abilities or the services they can provide.

- **Ontology**: Participating agents, especially *OA*s, are strongly recommended to have their own ontologies regardless of representation languages and hosting platforms. In this sense, the underlying ontology provides definitions for the terms used in a particular domain and also allows corresponding *OA*s to be able to ask for help from other *OA*s if the same agent is unable to fulfill a certain task (e.g. query).

- **Mediation**: This is the mechanism by which the overall operations can be monitored through a certain process.

Without doubt, all these three types of existing architectures have influenced our work. For example, designing service agents to act on broker agents and mediation is motivated by InfoSleuth. Nevertheless, our work goes beyond these three, as

it provides an integrated view of the overall ontology management, from mapping, integration to refinement. It is significantly distinguished from other work in that characteristics such as flexibility and reusability are highlighted. We emphasize our innovations in applying MAS to ontology management to achieve interoperability.

In terms of framework flexibility, high modularity makes it possible to easily adapt it to specific domains. As for reusability, in practice, this is not possible without incorporating specific techniques. In order to avoid taking on the heavy burden of reinventing the wheel, approaches or mechanisms that promote ontology reuse are highly recommended. In our framework, the problem of realising ontology reuse for the purpose of achieving ontology interoperability is ultimately reduced to providing feasible mechanisms to create special agents which are responsible for taking any further actions. The proposed framework enables us to take these into account to significantly enhance the effectiveness of applications created under the framework.

In short, in our framework, agent technology, a technique deemed most suitable for exploiting complex systems in dynamic and heterogeneous environments, is employed. Moreover, actions of different ontologies are based on knowledge obtained at run-time by means of interactions. The following characteristics become evident to enable the proposed framework to cope with dynamic and heterogeneous features of an environment:

- flexibility
- interactivity
- interoperability
- scalability
- reusablity
- reliability

In the following chapters, we highlight these important aspects of our approach and illustrate our innovations in applying MAS to ontology management. We will come back to these characteristics after prototyping in Chapter 6.

3.5 Summary

This chapter presents the ontology definition and relevant terminology to be used throughout this thesis. An agent-based approach for ontology management is examined thoroughly. Based on the analysis, a novel framework has been presented with a brief introduction to the different kinds of agents involved. Agent communication by means of message passing between agents has also been discussed.

Our framework has the following characteristics that differentiate this work from others:

- The ability to perform functionalities based on exchanging messages;
- `Functionary agents` can easily access relevant information or knowledge of existing ontologies via `ontology agents` (*OA*s);
- The presence of the *OA*s allows adaptive system organisation; and
- The ability to generate an *OA* on the fly which looks after the newly integrated ontology when needed;

Further work is needed to detail the proposed framework. Mechanisms of defined agents in our framework will be discussed in Chapters 4, 5, 6 and 7, respectively.

Chapter 4

Agent-based Ontology Mapping

It is hard to have a conformable view as long as different organisations have various aims and purposes where ontologies are concerned. In this sense, an ontology is not a solution for all unless everyone adheres to the same one. It is difficult and requires great effort to construct an ontology that is sufficient for the environment due to large-scale, individual privacy needs, dynamics and heterogeneity. On the other hand, it is a trend for different organisitions to work together to solve a problem beyond individual capabilities and knowledge. All these factors may take place at unpredictable times between unpredictable organisations. Agent technology [150, 81], as stated in Chapters 2 and 3, embodied with autonomy, adaptation, and other features, fits well in this situation by providing a MAS environment with agents working together to solve problems. A MAS perspective is thus suitable for tackling ontology mapping within and across the boundaries of organisations. The aim of this chapter is to develop novel agent-based ontology mapping mechanisms to operate ontology mapping dynamically to achieve ontology interoperability.

Ontology mapping is treated as a reconciliation approach and is seen as a key technology because it does not intend to unify ontologies and their data, but accepts this fact and endeavours to identify how different ontologies are mapped and related. It is a feasible and effective mechanism of communication among multiple ontologies or ontology-based applications. Of the approaches to ontology mapping, there are two major methods so far. They are manual ontology mapping and automatic ontology mapping. Obviously, distributed and dynamic environments desire

automatic ontology mapping. In practice, most work in this area has been classified to the former because of the prior knowledge required or user intervention desired from time to time. However, the prior knowledge may not take effect then; or the workload of a user may increase dramatically since ontology evolution is inevitable. We refer the reader to an excellent and thorough review [58] for a detailed discussion in this field. It is a great challenge to perform ontology mapping as flexibly as possible by considering heterogeneous ontology sources (including heterogenous platforms and different ontology representations) in a dynamic environment. To our best understanding, to address ontology mapping on the fly through agent communications or interactions is a novel and feasible approach. It is better than current systems and tools which treat the environment of ontology mapping mainly statically.

To this end, agent-based ontology mapping mechanisms based on previous work [77] are developed. A crucial feature of our work is its flexibility and scalability; its ability to perform ontology mapping dynamically. It is natural that in business, the achievement of some goals are beyond individual capabilities and knowledge. Hence, working together to achieve goals is inevitable. When adopting an agent-based vision in ontology mapping, it soon becomes clear that multiple agents are needed to represent multiple perspectives of competing but coordinating organisations. As actions of individual agents are based on interactions, any changes of ontologies may reflect on their interactions. Thus the proposed ontology mapping mechanisms can be thought of as also taking ontology changes or evolution into consideration. We anticipate that applications of knowledge management, e-commerce and the Semantic Web could benefit from the flexible ontology mapping proposed in this thesis.

This chapter starts by addressing problems in current ontology mapping in Section 4.1. Then some specific related work of ontology mapping is discussed in Section 4.2. The scope of agent-based ontology mapping is detailed in Section 4.3. Based on these analyses and descriptions, mapping process and mapping mechanisms are examined in Section 4.4 and Section 4.5 respectively, followed by examples in Section 4.6. A summary is given at the end of this chapter.

4.1 Problems in Ontology Mapping

Ontology mapping is seen as a solution in today's landscape of ontology research. A single ontology (if applicable) is no longer enough to support the tasks envisaged by a distributed environment such as the Web. A variety of ontologies are made publicly available and accessible. And the tendency is increasing steadily given the next generation of the Web, the so-called Semantic Web, is emerging. Mapping could provide a common layer from which several ontologies could be accessed and hence could exchange information in semantically sound manners [58]. Developing such mappings has been the focus of a variety of works over a number of years. However, some problems listed below are still critical issues for reaching mapping objectives.

- Mapping is conducted according to previous information, little response to changing environment;
- Mapping process does not allow agents to reason using part of the other agents' ontology;
- Mapping results, if they really represent the semantic relations between the concepts in the ontologies of the agents, are no longer being prepared for use any more.

The first limitation hampers ontology mapping in adapting to a dynamic environment which is seen as pervasive nowadays. To overcome this problem, the agent-oriented paradigm is applied to describe changes of other agents (the environment) and hence provides sufficient information for this agent to make a decision on further action. The second impediment has a serious impact on the original goal of ontology, in which defined ontology is intended for sharing among different applications with the ability to reason over available ontologies [107]. Since ontologies are developed for use with reasoning engines and semantics of ontology are specified with reasoning in mind, inference and reasoning are at the heart of ontology research. In a MAS, an agent is devised to respond properly, based on available information (including knowledge[1] and information derived from agent interactions).

[1]The term *knowledge* implies that the facts have been analysed, condensed, or combined with other facts to produce useful information.

The reasoning ability of an agent enables knowledge sharing between applications. In addition, existing ontology mapping works have paid little attention to applying mapping results. As ontologies are envisioned to assure interoperability, ontology mapping can be treated as a potential way to make progress in this area. Therefore, rather than just setting up for mapping as mentioned, mapping results should allow further reasoning to be performed for the purpose of knowledge sharing.

4.2 Specific Related Work

Similarity measurement is one of the most important issues where ontology mapping is concerned. Rodrguez et al. [124] give a general overview of similarity. Li et al. [71] define ontology mapping based on similarity computation.

Maedche et al. [87] present an ontology engineering process and elementary changes in ontology as well in forms of Add_Concept, Add_SubConceptOf, Delete_Concept, and Delete_Property, etc. This is the foundation of ontology mapping and related research work.

Of the related work in this area, some progress has been made already. For instance, PROMPT [108] uses the structure of ontology definitions and the structure of a graph representing an ontology to suggest to ontology designers which concepts may be related. It is a semi-automatic approach to ontology merging and alignment. Users' intervention is required. The work of McGuinness on Chimaera [89] presents potential matches (including hierarchical structures) to make an easier decision for users. GLUE [22] applies machine-learning techniques to find ontology mapping by defining probability to several practical similarity measures. These approaches are focused on providing tools to assist in the ontology mapping process. IF-MAP [127, 57] is an automatic method of ontology mapping based on the Barwise-Seligman theory of information flow [5]. Lacher et al. [66] present CAIMAN, a system which uses machine learning for ontology mapping. Text classification is utilised to measure the probability that two concepts are corresponding. Prasad et al. [121] develop a mapping mechanism (in ITTalks) which uses text classification techniques for similarity information collection, and Bayesian reasoning for resolving uncertainty in similarity comparisons. A certain threshold (posterior

probability) is applied to obtain the best mapping. Compatangelo et al. [20] present a system (ConcepTool), which adopts a description logic approach to formalise a class-centred, enhanced entity-relationship model. They also use linguistic and heuristic inferences to compare attributes of concepts. Their approach is similar to MAFRA's framework [85, 131] in that they both define semantic bridges. Bouquet et al. [13] present an approach for semantic mappings based on SAT, which is in the direction of complex matches.

Of the other work in this field, Maedche [84] mentions an essential part of ontology engineering - ontology learning. The development of the taxonomic backbone of the ontology is also involved. The clustering analysis mainly discussed in natural language processing previously, has been highlighted from learning the taxonomic relation perspective. The work of Obitko et al. [111] assumes that the sending agent knows the structure of the target ontology. A mapping approach is established based on a mapping matrix which takes "yes", "no", and "unknown", but no further detail about the "unknown" value. Wiesman et al. [144] indicate three types of conflicts in terms of ontology mapping. Their work addresses "structure conflict" and "name conflict" conflicts by introducing flattened representations of instances. Rules are defined to solve conflicts if there is more than one instance of the target ontologies competing with each other to match the source instance in a certain circumstance. However, their flattened representations are pertinent to structures of ontologies rather than playing no role as stated in the paper. We believe ontology semantic mapping should take structures into consideration. Maedche et al. [85] propose that mapping is one way to mediate between ontologies. They argue that existing approaches/tools are mostly centralised and fail to properly adapt to distributed systems. A "semantic bridge" is undoubtedly a bridge to fill the gap between two ontologies. However, domain experts' intervention is required from time to time during the mapping process. The work of Silva et al. [131] also addresses the semantic bridge and demonstrates it with an example.

Still other related work is more concerned with instance-based ontology mapping to accommodate requirements such as cooperation in an open MAS. In an open agent environment, adapting rapidly to new categories, conceptualisations, and specification of domains is required because none of these can be given a finite

definition. Wang et al. [141] present a learning approach to deal with collectively designing common ontologies actively. However, the number of clusters needs to be fixed in advance.

Rather than treating the environment of ontology mapping in a static fashion as existing systems and tools have mainly done, our proposed agent-based ontology mapping mechanisms aim to tackle ontology mapping in a flexible way (e.g. dynamic instead of static). In addition, agents are devised with the ability to reason (e.g. to take appropriate actions) against perceived information (including mapping results). Certainly, mapping results are employed in one way or another. In this thesis, they are applied to ontology query (Section 4.7) and ontology integration (Chapter 5), respectively.

Some characteristics of our proposed framework [77] can be distinguished from others. For example, heterogeneous ontology representations (e.g. a wrapper-like converting module) and heterogeneous ontology hosting platforms (e.g. platform-independent framework enabling end-to-end interoperability between agents of different agent platforms) are possible in our framework. Moreover, the fact that agents act properly, given that they have the capability to perceive changes of an environment and respond promptly, also enables ontology mapping to take ontology change into consideration whenever needed.

4.3 Scope of Ontology Mapping

Some ontologies exist in the same area with different aspects and overlapping information. Organisations with independently created ontologies may need to contact each other in a certain scenario by mapping a source ontology to a target ontology at an unexpected time and for an unexpected reason. The strengths of dynamic mapping are prominent. Nevertheless, it is not easily achieved. We attempt to enable dynamic semantic mapping by taking heterogeneous platforms and different ontology representations into consideration (Figure 4.1).

Figure 4.1 points out the scope of ontology mapping with respect to the other parts of ontology management. In can be roughly grouped into three parts from left to right:

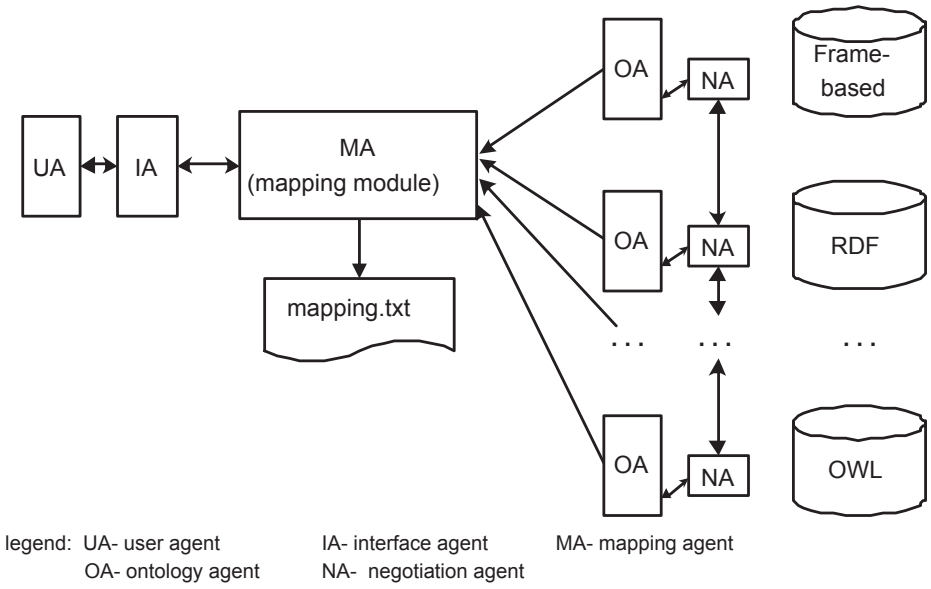

Figure 4.1: Ontology mapping

(1) On the left, there are the *UA* and the *IA*, in which the *UA* gathers information and translates it into a more precise form, and also presents the results back to the user in an acceptable and understandable form. Certainly it is the *IA* which assists the *UA* to realise its full functions by dealing with the dynamic nature of the ontology hosting environment, where ontologies become available, and then go off-line;

(2) In the middle, the mapping module works out semantic mappings between pairs of ontologies if possible. `mapping.txt` represents mapping results; and

(3) On the right, *OA*s (acting on behalf of existing ontologies) as well as *NA*s (taking part in negotiation to anticipate ontology change and report it to *OA*s) provide operable ontologies.

4.4 Mapping Process

In our proposed framework [77], there are different kinds of agents to represent multiple perspectives of competing but coordinating organisations, and the distributed nature of the problems. Suppose an *OA*, which acts on behalf of the corresponding

ontology, is responsible for ontology related tasks. It provides as much information of the ontology it acts on as possible. The *OA* operates over two structures: (1) the ontology structure (e.g. the ontology it acts on behalf of); and (2) the mapping result (e.g. `mapping.txt`).

Unlike the *OA* or other agents, the *MA* does not operate directly over existing structures. It has the effect via the *IA* in the module (Figure 4.2) of deciding whether existing ontologies come from the same domain or not; or in the module of ontology mapping (Figure 4.3) through the *OA* and *SA* to acquire relevant information. The former paves the way for deployment of predefined rules in the latter. Actually, these rules are the little required prior knowledge (see **Definition *Equivalent*** & **Definition *Inclusive*** in Section 3.1). The rules are:

R1 : If two concepts have the same labels, they are of the same meanings, namely they are semantically equivalent.

R2 : If labels of two concepts are synonyms (from a dynamic generated synonyms list), they are semantically equivalent.

R3 : If two concepts have the same attributes as each attribute of the same *datatype*, namely, if pairs of <*attribute, datatype*> are the same; they are semantically equivalent.

R4 : If all pairs of <*attribute, datatype*> of one concept are also pairs of another concept, there is a class-subclass relationship between them.

R5 : If super-concepts are the same (according to **R1**, **R2**, **R3**), sub-concepts of one node (e.g. super-concept here) are also sub-concepts of another node.

In Figure 4.3, a scenario starts with the *MA* sending a particular request to the existing *OA*s to obtain the required concepts. The *MA* then performs the mapping module to check whether the given concepts are semantically equivalent or inclusive. Now, let us take a close look at the mapping module presented in Figure 4.3.

An *MA* is responsible for ontology mapping related tasks by working with *OA*s. The mapping module starts with the *MA* sending *OA*s requests (for the current concept) to two given ontologies (for simplicity, interactions between the *UA* and

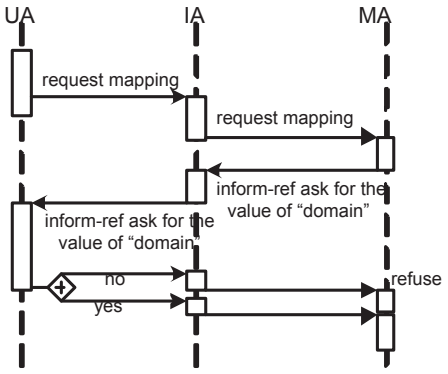

Figure 4.2: Module for deciding the domain

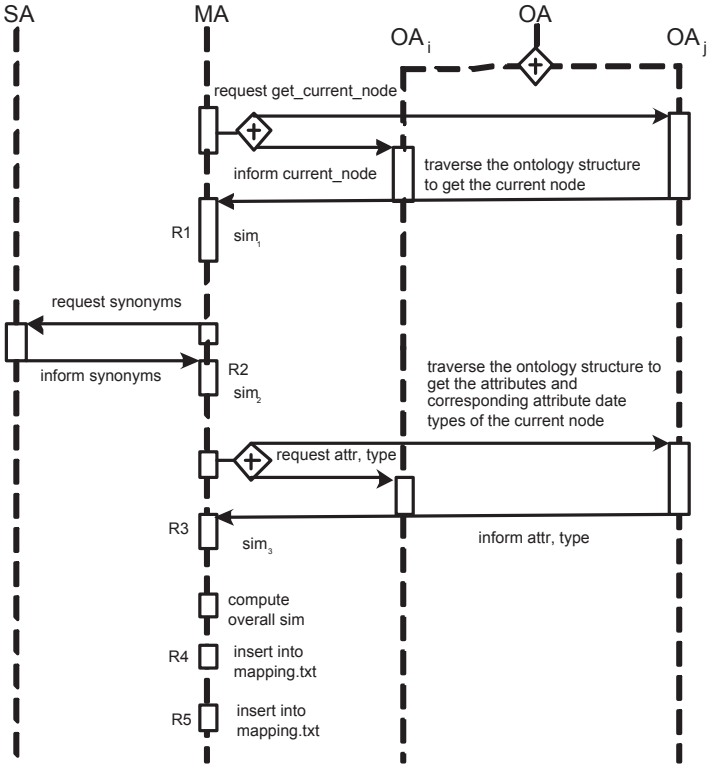

Figure 4.3: Interactions from *MA*'s view

IA, *IA* and *MA* are omitted). It then checks if two acquired concepts satisfy the rules in its knowledge base. The similarity measurement is defined as:

$$sim(c_1, c_2) = \sum_{j}^{k} w_j sim_j(c_1, c_2)/k'$$

where $j, k, k' \in \mathsf{N}$, $k' \leq k$, k' is the number of non-zero similarity of a specific method, w_j is the weight for a specific method, $w_j \in$[0,1]. $sim_j(c_1, c_2)$ is the similarity measurement of a specific method. When the summarising similarity of two concepts from similar ontologies is greater than a given threshold δ, the *MA* sends a specified *OA* a request for inserting the mapping result into a `mapping.txt`. The process may loop and wait to be enacted for further acquired concepts until it runs out of sub-concepts of a specified ontology. Mapping is conducted from a particular *OA*'s perspective. In other words, mapping is with directions. This is outlined as follows:

(1) obtain ontology related information via *OA*s;

(2) estimate obtained information according to the knowledge in the *MA*'s knowledge base, namely *MA* computes if semantic relations between pairs of ontologies exist, and which kind of semantic relations they are;

(3) write the mapping results to a file (e.g. `mapping.txt`);

Following the above approach, mapping results are available for further ontology operations such as ontology integration discussed in Chapter 5.

4.5 Mapping Mechanisms

The mapping module operates over available ontologies in order to achieve ontology interoperability. The *MA* is in charge of mapping in the environment. The module uses the following functions or data structures to check whether two given concepts meet the predefined rules.

- `initilise`: initilise the process.
- `next-c`: request a particular *OA* for the next concept of a specified ontology. It returns the concept if it exists;

- `next-a`: request a particular *OA* for the attribute of a specified concept. It returns the attribute if it exists;
- `next-t`: request a particular *OA* for the corresponding data type of a certain attribute of a specified concept. It returns the data type if it exists;
- `comp`: compare two items. It returns *true* if two compared items are equal. Otherwise, it returns *false*;
- `syn`: request the *SA* for the synonyms of a given concept. It returns a synonym list (e.g.$syn - list$) if it exists;
- `measure-sim`: compute the weighted similarity over the given methods (see Section 4.4 for the calculation formula);
- `add`: add items to the mapping results (e.g. `mapping.txt`).
- `next-i`: request a equivalent relation from the `mapping.txt`;
- `add-sub-concept`: add inclusive relations to the `mapping.txt`.

The pseudocode for the mapping algorithm is shown below.

Pseudocode of mapping algorithm

/*assume the mapping module starts from a given start point;
O_1, O_2: the source ontology and target ontology, respectively;
c_1, c_2: concepts from the source ontology and target ontology, respectively;
att_1, att_2: the attributes of c_1, c_2, respectively;
$type_1, type_2$: data types of att_1, att_2, respectively;
$flag_1, flag_2$: two flags with Boolean values;
$syn - list$: a list of synonyms of a specified concept label returned from function `syn`;
w_1, w_2, w_3: a weight for a specific method;
sim_1, sim_2, sim_3, sim: similarities;
δ: a given threshold to filter out inappropriate mappings;
`mapping.txt`: mapping results.

```
*/
Function mapping  {
1. initialise;
2. do  {
3.   c1=next-c(O1);
4.   c2=next-c(O2);
5.   if !(comp(c1.label, c2.label))  {
6.        syn(c1.label);
7.   } else {
8.   sim1=measure-sim(R1);
9.   } //end if
10.   flag1=flag2=true;
11.   while (syn-list(c1.label)!=Null)  {
12.        if !(comp(syn-list(c1.label), c2.label))  {
13.        flag1=false;
14.        } else {
15.        sim2=measure-sim(R2);
16.        } //end if
17.        if !(flag1)  {
18.          while (next-a(c1)!=Null && next-a(c2)!=Null)  {
19.            att1=next-a(c1);
20.            att2=next-a(c2);
21.            type1=next-t(c1);
22.            type2=next-t(c2);
23.            if !(comp(att1, att2) && comp(type1, type2))  {
24.            flag2=false;
25.            } else {
26.           sim3=measure-sim(R3);
27.          }// end if
28.         }//end while
29.        }//end if (flag1)
30.        if (flag2)  {
```

31. **if** ((`next-a`(c_1)!=Null) && (`next-a`(c_2)==Null)) {
32. `add`(`mapping.txt`,($c_1, c_2, \sqsupseteq$));
33. }//end if
34. **if** ((`next-a`(c_1)==Null) && (`next-a`(c_2)!=Null)) {
35. `add`(`mapping.txt`,($c_1, c_2, \sqsubseteq$));
36. }//end if
37. }//end if ($flag_2$)
38. sim=$\sum_{j=1}^{3}$`measure-sim`(R_j);
39. **if** $sim \geq \delta$ {
40. `add`(`mapping.txt`,($c_1, c_2, =$));
41. }//end if
42. }//end while
43. } **while** (`next-c`(O_1)!=Null && `next-c`(O_2)!=Null);//end do
44. **while** (`next-i`!=Null) {
45. `add-sub-concept`;
46. } //end while
47. }// end function

4.6 Examples

The running example ontologies come from the domain of beer and concern the types of beer. The first one is from the DAML ontology library (*http://www.daml.org /ontologies/66*). A fragment of this is shown in Figure 4.4. The second one is built on the definition of the term beer from the WordNet (*http://wordnet.princeton.edu/*). A fragment of this is shown in Figure 4.5. The third one is based on basic types of beer provided by the website *http://www.dma.be/p/bier/1_2_uk.htm#*, which states: "every beer in Belgium, the Netherlands and Luxemburg is classified in one of the given basic styles. These basic styles are the building blocks of every thematical classification of beer styles". A fragment of this classification is shown in Figure 4.6. The fourth one is an Australian beer types ontology based on information from websites: (1)*http://www.australianbeers.com/beers/beer_types/beer_types.htm*;

and (2) *http://www.fosters.com.au/beer/ about/beertypes/beer_types.asp.* A fragment of this classification is shown in Figure 4.7. Our main interest is in the term beer and the corresponding hyponym relationship. These four ontologies are about types of beer, but from different points of view.

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns="http://www.daml.org/2001/03/daml+oil#"
xmlns:daml="http://www.daml.org/2001/03/daml+oil#"
xmlns:gen="http://www.cs.umd.edu/projects/plus/DAML/onts/general1.0.daml#">

<Ontology about="">
  <versionInfo>beer-ont, v.1.0</versionInfo>
  <comment>An ontology that models brewers and types of beer.</comment>
  <imports resource="http://www.cs.umd.edu/projects/plus/DAML/onts/general1.0.daml"/>
</Ontology>

<Class ID="ScotchAle">
  <subClassOf resource="#Ale"/>
</Class>

.....
```

Figure 4.4: Fragment of beer ontology from DAML ontology library

In the running example ontologies, we assume that $\mathcal{R}$ consists of relations $\{$*hyponym*(is-a), *part-of*$\}$. Moreover, we assume that participating ontologies agree upon a small set of common sense knowledge regardless of syntactical, structural and language heterogeneity to conform to a normalised uniform representation. Generally speaking, the taxonomy is at the heart of ontologies and ontology applications. So we also assume that ontologies have the same structures (e.g. hierarchical). Moreover, we suppose that there are no polysemous words in an ontology, and there is only one possible relation between two concepts of the ontology. Finally, we assume that different ontologies are available and there is no need to consider conceptual modelling issues ourselves.

These four examples are used to demonstrate the mapping mechanisms in a MAS environment. Different kinds of agents (e.g. a *UA*, an *IA*, a *MA*, a*QA* and an *OA*) have been created to take different roles in ontology mapping. see Chapter 6 for detailed implementations.

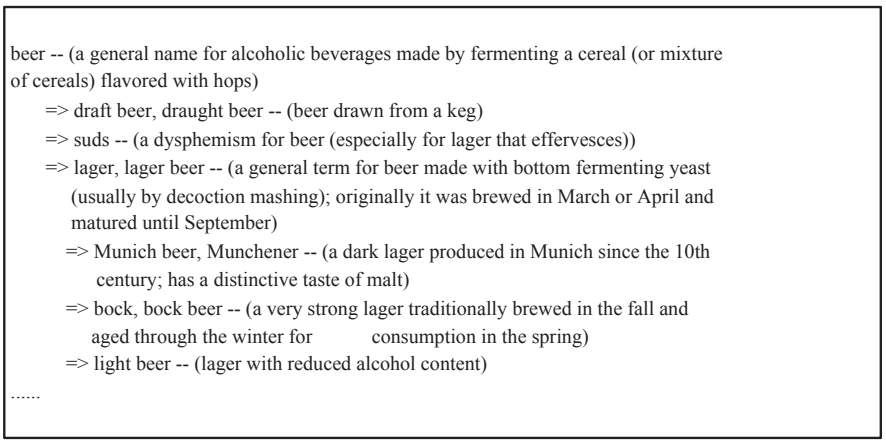

Figure 4.5: Fragment of term beer from WordNet

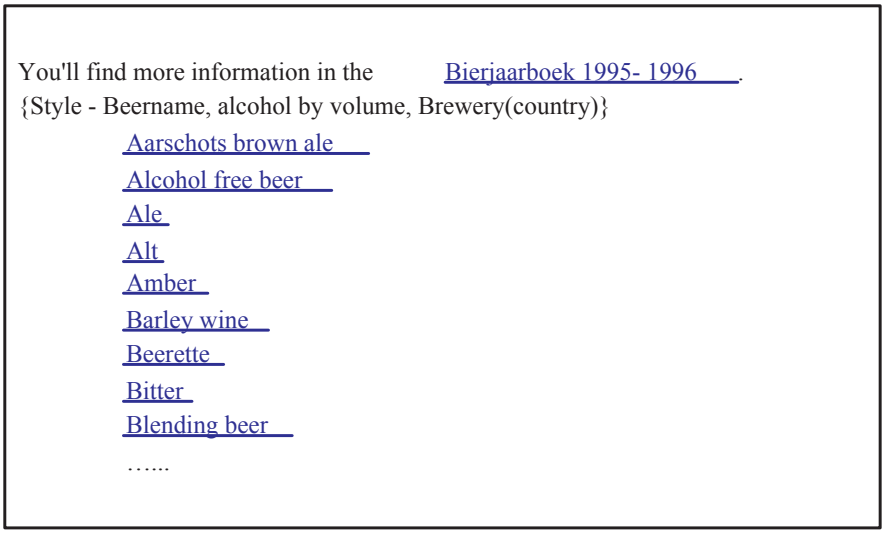

Figure 4.6: Fragment of classification of basic beer types

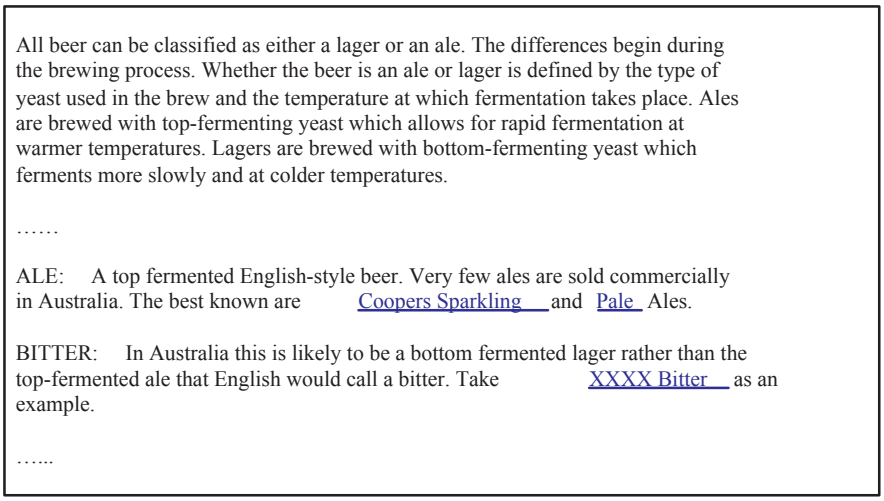

Figure 4.7: Fragment of Australian beer types

4.7 Ontology Mapping Evaluation

We evaluate the mapping mechanisms in the beer type domain with the above four different beer ontologies. The evaluation is conducted to verify the quality of the MAS mapping results against the expected ones in the same domain.

4.7.1 Query in Prototype

Based on the semantic equivalent relations built in the mapping process, further operations such as query over other than this ontology can be achieved.

The query is conducted to verify the mapping - equivalence effectiveness on the prototype. In the query module, we deliberately query a particular *OA* before/after ontology mapping. The experiments clearly show that a particular *OA* can instantly return corresponding concepts or *true/false* of given propositions after running the mapping module. As an *OA* can refer to the corresponding `mapping.txt` file(s) for each pair of ontologies, the *OA* is certain about the results and can obtain what is requested by dispatching the query to other *OA*s if it is unable to answer the query itself. We may regard existing ontologies as **vertices** (natural numbers are used to notate vertices), and mappings between ontologies (if they exist) as **edges**. As mapping is conducted between pairs of *OA*s, when there is a mapping from one ontology to another, the query module draws an edge to link these two vertices

together with a direction. In the end, a directed graph is obtained. Assuming that the number of available ontologies is n, clearly, it is a weakly connected graph[2] if $(n-1)$ pairs of ontologies have performed mapping (see Figure 4.8 for example).

The main task of the *QA* is to maintain an effective path from a specified agent (e.g. an *OA*) to a target ontology in a decentralised ontology network when the queried term is processable. The query module is defined as follows:

(1) *request* a query from a specified agent e.g. an *OA*;

(2) *inform* the *QA*, two cases may take place in this situation:

   (a) if the queried term is understandable by the specified agent *OA*, then *END*;

   (b) otherwise, this specified *OA* dispatches the query to other available *OA*s which are known to this *OA*, whilst it informs the *QA* which agent the query is passed to.

(3) *draw* a line from the query sending agent *OA*s to the query receiving agent *OA*s;

(4) *repeat* steps (2) and (3) until the queried term is understandable or the process runs out of candidate ontologies;

(5) *return* a path annotating by the vertex numbers.

In terms of which agents dispatch the query, the *OA* may choose either depth-first search (DFS) or broad-first search (BFS) to dispatch the query. We are concerned about a reachable path at the end, so DFS is used in our work.

We believe that a query can be made by the query module provided the graph is `weakly connected`.

Below are representative examples. For simplicity, $q(OA_i, x)$ is for query(OA_i, x); where $i \in [1..4]$; x is the content of query; OA_i is an agent which acts on behalf of the existing ontology (e.g. $\mathcal{O}_i$); and c_i^k ($k \in \mathsf{N}$) represent an concept in ontology O_i. In this example, dispatching a query from OA_1 to OA_2 is denoted as

[2] A `weakly connected graph` is where the direction of the graph is ignored and the connectedness is defined as if the graph was undirected.

$OA_1 \longrightarrow OA_2$, and the path is denoted by the vertex number in a form of $1 \longrightarrow 2$ if vertex 2 is reachable from vertex 1. Suppose the initial state is: q(OA_1, c_4). The query module is conducted as follows:

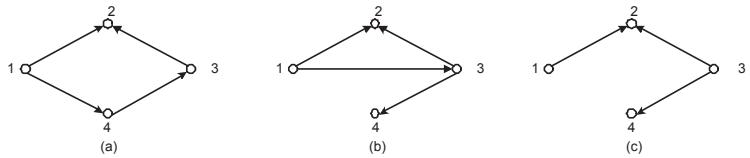

Figure 4.8: Query in a directed graph

Case 1 (Figure 4.8(a)): the query can be executed via OA_1 because of a direct edge from $\mathcal{O}_1$ to $\mathcal{O}_4$, i.e. no need to dispatch the query to any other agents;

Case 2 (Figure 4.8(b)): since no such kind of mapping exists, the query module may dispatch a query to any agents associated. In this case, the optional agent set is $\{OA_2, OA_3\}$. The process is as follows:
If $OA_1 \longrightarrow OA_3$, it is settled. The path is $1 \longrightarrow 3 \longrightarrow 4$;
If $OA_1 \longrightarrow OA_2$, then another dispatching (e.g. $OA_2 \longrightarrow OA_3$) is needed. Moreover, as the mapping direction is from $\mathcal{O}_3$ to $\mathcal{O}_2$, the corresponding agent (e.g. OA_2) needs to reverse the mapping results. After this is settled, the query is made. The path is: $1 \longrightarrow 2 \longrightarrow 3 \longrightarrow 4$. But the best path is: $1 \longrightarrow 3 \longrightarrow 4$ in this case.

Case 3 (Figure 4.8(c)): the module will follow the process: $OA_1 \longrightarrow OA_2 \longrightarrow OA_3$. The best path is: $1 \longrightarrow 2 \longrightarrow 3 \longrightarrow 4$.

In any case, a query can always be made given the graph (mapping) is `weakly connected`.

The mapping results shown in Table 4.1 also illustrates that equivalent-based query works very well. However, it is not alway the case. A further study in the inclusive results is needed. Details are described in the following section.

4.7.2 Mapping Results Analysis

Ontology mapping is executed between pairs of ontologies. Mapping mechanisms will decide equivalent and inclusive relations then. Let us have a look at the

mapping results in the beer type domain. We will illustrate the MAS mapping results against expected ones in the same domain in the following. For simplicity, we focus on O_2 in Table 4.1. But others can be done in the same way. Details are displayed in Table 4.1. Notations in the table are defined as follows:
O_s: an source ontology involved in mapping;
O_t: an target ontology involved in mapping;
M→: two ontologies involved in mapping;
MAS Eq: the number of MAS equivalent relations (MAS mapping mechanisms proposed in Section 4.5);
MAS Inl: the number of MAS inclusive relations (MAS mapping mechanisms);
Eq: assumed number of equivalent relations which would be picked up by human user;
Inl: assumed number of inclusive relations which would be picked up by human user.

Table 4.1: MAS mapping evaluation

O_s	O_t	**M**→	**MAS Eq**	**MAS Inl**	**Eq**	**Inl**
O_1	O_2	$O_1 \rightarrow O_2$	10	10	10	6
O_3		$O_3 \rightarrow O_2$	5	2	5	0
O_4		$O_4 \rightarrow O_2$	7	3	7	2

As shown in Table 4.1, the equivalent relations in mapping have been correctly picked up (columns 4 and 6 in the table). But there are differences between columns 5 and 7 in relation to inclusive relations. The difference varies from ontology to ontology. For example, the difference in $O_4 \rightarrow O_2$ is 1 compared with number 4 of the difference in $O_1 \rightarrow O_2$.

It seems that mapping - equivalence works very well in this domain (we have discussed in Section 4.7.1). The reason is that the defined syntactic-based similarity is sufficient to deal with similarity measurement for the time being.

On the other hand, mapping - inclusion works in a slightly different way from mapping - equivalence. It is mainly caused by various conceptualisations of involved ontologies. Let us take a close look at the cause of the differences in Table 4.1.

- $O_1 \rightarrow O_2$: in O_1, `bock`, `Pilsner`, `porter` and `stout` are sub-concepts of `beer`. However, in O_2, `bock` and `Pilsner` are sub-concepts of `lager`, while `porter` and `stout` are sub-concepts of `ale`. Also in O_2, `lager` and `ale` are sub-concepts of `suds` which is equivalent to `beer` according to proposed similarity methods. At the end, the difference or inconsistency comes to appear (more details will be discussed in Section 5.7, Chapter 5). The differences leads to that inclusive relation is 10 (row 2, column 5, Table 4.1) instead of assumed 6.

- $O_3 \rightarrow O_2$: for the same reason, `porter` and `bitter` with different definitions in O_3 and O_2 lead to that inclusive relation is 2 (row 3, column 5, Table 4.1) instead of assumed 0.

- $O_4 \rightarrow O_2$: the only difference is `bitter`. The reason is that in Australia, bitter is likely to be a bottom fermented lager rather than the top-fermented ale that English would call a bitter. This leads to that inclusive relation is 3 (row 4, column 5, Table 4.1) instead of assumed 2.

As discussed in the above, these inconsistency cannot be solved until performing the consistency checking. Consistency checking is necessary to eliminate inclusive relation differences as shown in columns 5 and 7, Table 4.1. We will investigate into consistency checking in Section 5.7, Chapter 5.

4.8 Discussion

Rather than mainly treating the environment of ontology mapping statically as in most existing systems and tools, the proposed agent-based ontology mapping aims to tackle ontology mapping in a more flexible way. On the one hand, under our proposed framework [77], as an ontology representation has a corresponding wrapper-like converting module to translate the ontology representation into a common representation, heterogeneity of ontology representations can be handled. On the other hand, the agent platform used in the work enables end-to-end interoperability between agents of different agent platforms. Thus heterogeneity of platforms can also be handled. Moreover, the fact that agents act correctly, given that they

are capable of perceiving changes in an environment and respond promptly, also enables ontology mapping to take ontology evolution into consideration whenever needed. A comparison of the proposed agent-based mapping approach with other existing mapping mechanisms is illustrated in Table 4.2. Seven criteria are at the core of the comparison. They are:

1. **Specific Technique**: This shows if specific techniques are used to facilitate ontology mapping.

2. **Mapping Type**: This indicates if mapping is 1-to-1 or other general cases. It will also detail which kind of mappings (e.g. SubClassOf, PartOf) they are if applicable.

3. **Logic**: This supports logics in the systems (e.g. Horn Logic).

4. **Ontology Organisation**: This specifies ontology structures over which mapping is operated. In most cases, it is a taxonomy or a hierarchical structure when ontology organisation is involved.

5. **Interaction**: This indicates that interactions between participants/agents play important roles for a certain goal.

6. **Language**: This means supported ontology representation languages, such as RDF, DL, etc.

7. **Integration/Composition**: This checks if the mapping results will be used in further ontology integration or composition. It is argued that ontology mapping is the foundation of ontology integration. So we expect systems will take it into consideration.

Notations are the same as those presented in Section 2.1.2. For simplicity, we use corresponding numbers to specify different columns in Table 4.2.

It is clear that existing systems/tools have ignored features such as interactions between actors and have shown no intention to deploy mapping results for further ontology management. Our proposed agent-based mapping approach has three unique features compared with some existing systems:

Table 4.2: Comparison of proposed agent-based mapping with some existing mechanisms

System /Tool	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Our Agent-based Mapping	agent technology	1-to-1 (Synonym, AttibuteOf, SubClassOf)	fragment of FOL; DL	taxonomy	Yes	RDF; with Protégé, it may support more semantic languages	Yes
Chimaera	KSL Ontolingua platform	1-to-1 (synonym, subsumed)	N/A	taxonomy	N/A	OKBD-compliant representation languages such as ANSI KIF, Ontolingua, Protégé, CLASSIX, and iXOL	N/A
CAIMAN	machine learning; text classification	1-to-1	N/A	taxonomy	N/A	N/A	N/A
ConcepTool	DL reasoner	1-to-1	DL; E-R model	taxonomy	N/A	N/A	N/A
GLUE	machine learning (joint probability distribution)	1-to-1	fragment of FOL	taxonomy	N/A	N/A	N/A

IF-MAP	information flow theory	generic	Horn logic	taxonomy	N/A	KIF [35]; Ontolingua; OCML [102]; RDF; Prolog	N/A
ITTalks	text classification; Bayesian approach	1-to-1	N/A	mostly taxonomy	N/A	DAML+OIL	N/A
PROMPT	knowledge-based approach	1-to-1	N/A	taxonomy	N/A	OKBD-compliant representation languages	N/A

- Agent technology is used. We expect that agent technology can significantly enhance the system's ability to cope with heterogeneous and varied ontologies in the context of the Web;
- Interaction is essential for agents to respond to changing environments;
- Ontology mapping aims to pave the way for ontology interoperability. So we consider using ontology mapping results for further ontology management.

Furthermore, our approach keeps pace with most existing mapping mechanisms in that the most popular relationships, for example, *SubClassOf, AttributeOf*, and *Synonym* are addressed throughout the thesis.

4.9 Summary

In this chapter, we have presented a novel agent-based ontology mapping in order to achieve ontology interoperability regardless of heterogeneous environments and different ontology representations. The proposed mechanism allows flexible system organisation by also taking ontology evolution into consideration. Moreover, it is

also in accordance with the little prior knowledge needed but acquiring relevant knowledge directly from agent interactions during run-time.

It is natural to take the further step to the use of mapping results because mapping is not a goal in itself. In the following chapters, Chapter 5 will discuss ontology integration, Chapter 6 will demonstrate ontology mapping and integration as well as evaluation. In fact, where ontology management is involved, ontology evolution must be addressed. Specifically, the succeeding chapter (Chapter 7) will exploit ontology refinement with process algebra as the theoretical background to providing a channel for timely information exchange.

Chapter 5

Agent-based Ontology Integration

As stated in the previous chapter, ontology mapping is not a goal in itself. It is the foundation for further actions, for instance, ontology integration. The aim of this chapter is to develop novel agent-based integration mechanisms to conduct ontology integration based on proposed framework in Chapter 3 and mapping results derived from Chapter 4.

This chapter starts by addressing problems in current ontology integration in Section 5.1. Then some specific related work of ontology integration is discussed in Section 5.2. Integration architecture incorporated with ontology reuse is described after that in Section 5.3. Based on these analyses and descriptions, an integration process and integration mechanisms are addressed in Section 5.4 and Section 5.5, respectively. Following on is a case study in Section 5.6. Integrated ontology is not always consistent, hence ontology consistency checking is examined in Section 5.7 followed by discussion in Section 5.8. Finally, comes a summary of the chapter.

Under the framework, agents' roles in interaction diagrams are highlighted. A crucial feature of our framework is its flexibility and extendibility; its ability to define new entrants in a dynamic environment. We anticipate that a spectrum of applications ranging from content management, e-commerce, to the Semantic Web could benefit from integration of ontologies.

5.1 Problems in Ontology Integration

Some of the specific challenges in ontology integration are as follows. More details of ontology integration issues can be found in [107].

- Finding similarities and differences between ontologies in an automatic and semi-automatic way;
- Defining mappings between ontologies;
- Representing uncertainty and imprecision in mappings;
- Composing mappings across different ontologies; and
- Developing an ontology integration architecture/framework.

We classify the above items into two categories. The first four items are about defining similarity measurements and mapping in certain and uncertain environments. The last item is about integration architecture. Although the problem of specifying the architecture is at the heart of integration for the Web, it has not been thoroughly investigated yet. It is a great challenge to automate ontology managing processes as much as possible to reduce the burden of manual operations for large scale resources. What we are going to address is closely associated with a novel architecture which fundamentally impacts on the performance of ontology integration. In this way, through ontology integration in the context of the Semantic Web it is possible to consider both flexibility and extensibility.

5.2 Specific Related Work

Some ontologies in the same domain exist with different aspects and overlapping information which are independently created by individual organisations. They may be merged later on if necessary, which would lead to ontology integration. Since integration itself is a process, it is natural to treat it from a process perspective. Pinto et al. [118] present activities that should be performed in the integration process. The work of them also describes a methodology to perform the integration. Their approach provides support and guidance for the activities

that compose the integration process. Even though each stage of the integration process still needs to be addressed in the future, we find some existing tools which claim to support it in some way. These systems are: PROMPT [108], Chimaera (*http://www.ksl.stanford.edu/software/chimaera/*), OntoEdit (*http://www.ontoknowledge.org/to*
FCA-Merge [133], OBSERVER [92], MOMIS [8, 9, 10], ONION [97, 98], and InfoSleuth [29, 105]. Moreover, some other tools listed in [25] support ontology integration to some extent.

Besides those tools, some researchers have investigated ontology integration from different technical aspects. Calvanese et al. [15] present a framework for ontology integration. Grüninger et al. [41], in their position statement, adopted an "Ontology Stance" to predict the set of sentences that the inference system must satisfy. In their work, mediating ontology, the so-called Interlingua architecture, is a neutral interchange ontology to fill the gap between applications. The work of Kalfoglou et al. on IF-MAP [59], draws on the proven theoretical grounds of Information Flow and channel theory which can provide a systematic and mechanised way for deploying the approach in a distributed environment to perform ontology mapping among a variety of different ontologies. Similar work [48] of Itoh et al. computes "uniqueness" (frame format information) from the information theory to provide an approach for automatic integration of different ontologies at the vocabulary level in the information retrieval area. Gómez-Pérez et al. [36] offer another way of combining ontology with problem-solving methods to configure new knowledge systems from existing, reusable components.

Another closely related work is Process Specification Language (PSL) (*http://www.mel.nist.gov/psl/ontology.html*). Its aim is to create a process interchange language that is common to all manufacturing applications, generic enough to be decoupled from any given applications and robust enough to be able to represent the necessary process information for any given applications. It is more appropriate to refer to it as an ontology or a data model than a language.

Our work is inspired by the approaches in information integration and PSL. We argue that different techniques and methodologies are complementary and thus must be used in combination and not exclusively. Bearing these in mind, we adopt a MAS perspective to model a variety of ontologies in ontology-based applications.

We believe that the agent technology is ideally suited in a dynamic and distributed environment such as ontology integration on the Web.

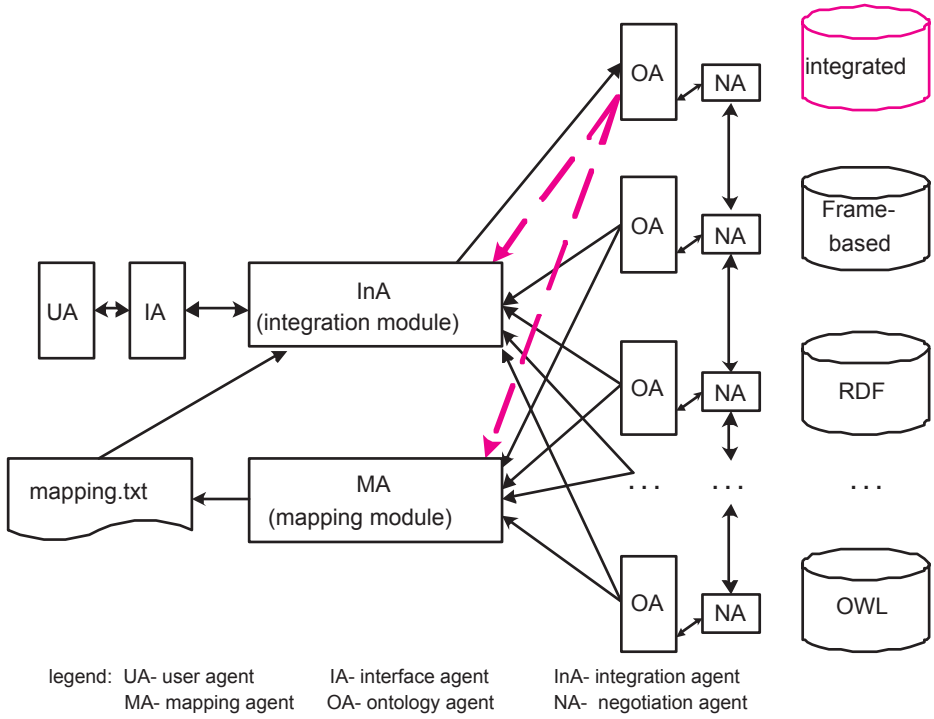

Figure 5.1: Ontology integration

5.3 Scope of Ontology Integration - Incorporating Ontology Reuse in Integration

Some ontologies in the same area exist with different aspects and overlapping information. Ontologies independently created by individual organisations may be integrated later on if needed. Moreover, integration serves ontology reuse in applications. Ontology reuse here has two meanings. On the one hand, existing ontologies can be used to generate new ontologies. On the other hand, newly generated ontologies are ready for reuse in the system whenever needed. Ontology integration embedded with ontology reuse is shown in Figure 5.1.

Figure 5.1 points out the scope of ontology integration with respect to the other parts of the ontology management. It clearly depicts the interdependency

of the ontology integration module and mapping module (refer to Section 4.3 for descriptions of the mapping module). Integration is based on mapping results, which are generated by the mapping module described in Chapter 4. A newly integrated ontology, displayed at the top right of Figure 5.1, can be reused in the system as shown by the dotted lines.

5.4 Integration Process

A vision of a MAS is not possible without agent interaction as discussed in Section 3.2. Figure 5.2 displays the interactions between agents involved in the integration module. In Figure 5.2, *InA* directly operates over the intermediate result, which records all concepts that meet the requirements, for instance, the items with the number of occurrences greater than a given threshold. A visualisation module of the *UA* can present a graphic view of the result upon request.

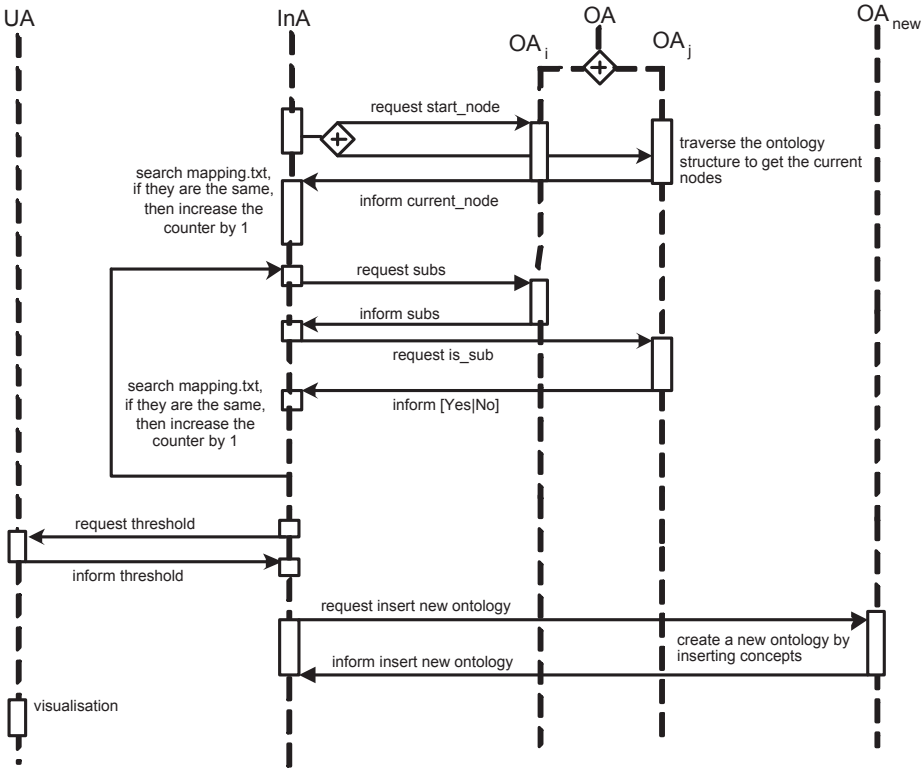

Figure 5.2: Interactions from *InA*'s view

In this thesis, the integration module starts from the root of the specified ontology and then traverses all sub-concepts of the ontology. Briefly speaking, *InA* counts the appearance of each concept of existing ontologies, and then filters unexpected concepts with a given threshold. By saying this, we do not mean that we attempt to change the conceptual modelling of the ontology. Rather, ontology integration is based on a specified ontology. The process is as follows:

(1) obtain ontology related information via *OA*s (e.g. by accessing the `mapping.txt`);

(2) keep the numbers of occurrences of each concept in the specified ontology;
Three cases may take place according to the mapping results. They are:

- Case 1: semantic equivalence for the current two concepts (for example, "beer" is the same as "suds"):
  In this case, increase the number of occurrences of the concept by 1 for each equivalence.
- Case 2: inclusive relation for the current two concepts (for example, "stout" is a kind of "ale"):
  In this case, insert sub-concepts of the counterpart into the specified ontology structure but keep the original relations.
- Case 3: no semantic equivalence for the current two concepts but their corresponding direct ancestors are semantically equivalent.
  In this case, insert the counterpart into the specified ontology but without conflict with existing sub-concepts of the same ancestor.

(3) filter unexpected concepts by a given threshold.

The user can request visualisation of the integrated ontology or export this new ontology to a RDF(s) format.

Following the above approach, a new *OA* is created instantly (to act on behalf of this ontology) corresponding to the newly derived ontology which is based on the mapping results of the mapping module. The newly created *OA* enables ontology reuse in practice since *OA*s in the proposed framework are devised to be responsible

for ontology related tasks. Hence, the integrated ontology can be reused with other existing ontologies in the repository. The presence of the *OA* makes the integration process self-contained in that it can directly and seamlessly reuse integrated results to continuously improve its effectiveness.

5.5 Integration Mechanisms

The integration module operates over available mapping results. The *InA* is responsible for ontology integration in the environment. The module uses the following functions or data structures to execute relevant operations. Pseudocode representation of the integration algorithm is shown below.

- `initialise`: initialise the process.
- `next-c` *concept*: request a particular *OA* for the next concept of a specified ontology. It returns the concept if it exists.
- `search`: search for relations in mapping results. It returns existing relations if it exists or NIL otherwise.
- `insert-sup`: insert a specified concept as a super-node of a given concept in a particular ontology structure.
- `insert-sub`: insert a specified concept as a sub-node of a given concept in a particular ontology structure.
- `get-threshold`: contact the *UA* via the *IA* to obtain a threshold. It returns the threshold.
- `filter`: filter unexpected items from a given ontology, and returns a filtered ontology.

Pseudocode of integration algorithm

/* assume the integration module starts from a given start point;
O_s, O_t: two different ontologies to be integrated, respectively;

O_d: the derived ontology;
c_s, c_t: the concepts from O_s and O_t, respectively;
l_{c_s}, l_{c_t}: the number of occurrences of concepts from O_s and O_t, respectively;
m: the number of available ontologies;
relation: relations between two given concepts from different ontologies;
threshold: the threshold given by the user to filter unexpected items from the generated ontology.
*/

Function `integration` {

1. `initilise`;
2. **for** $(i = 1; i < m; i++)$ {
3. **while** ((`next-c`(O_s)!=Null) && (`next-c`(O_t)!=Null)) {
4. c_s=`next-c`(O_s);
5. c_t=`next-c`(O_t);
6. *relation*=`search`(c_s, c_t);
7. **switch** (*relation*) {
8. **case** "=":
9. $l_{c_s}++$;
10. **break**;
11. **case** "$\sqsubseteq$":
12. `insert-sup`(c_t, c_s);
13. $l_{c_t} = 1$;
14. **break**;
15. **case** "$\sqsupseteq$":
16. `insert-sub`(c_t, c_s);
17. $l_{c_t} = 1$;
18. **break**;
19. **default**:
20. $l_{c_s} = 1$;
21. } //end switch
22. } //end while
23. } //end for

24. *threshold*=`get-threshold`;
25. `filter`(O_d, *threshold*);
26. } //end function

5.6 Examples

We assume that the mapping results of four examples are available from Chapter 4. Ontology integration is demonstrated by specifying "WordNet" beer (one of four available ontologies) as a source ontology. The newly generated ontology is shown in Figure 5.3. Newly generated ontology is organised in a hierarchical structure. Chapter 7 will provide more details.

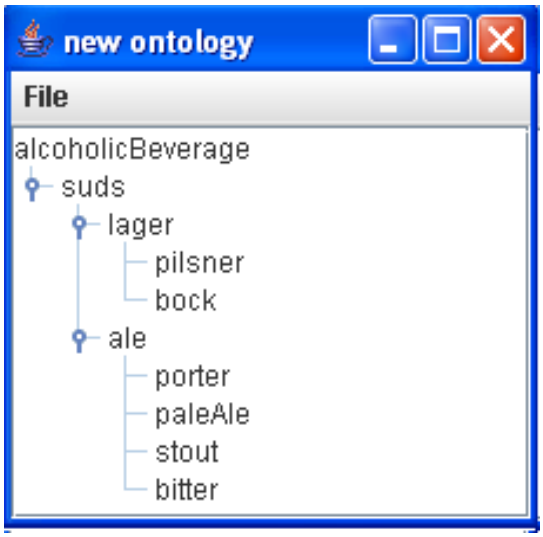

Figure 5.3: Integrated ontology

5.7 Consistency Checking in Prototype

To verify the prototype, *consistency checking* is conducted by applying a certain description logic (DL) based reasoning tool. In DL, consistency checking also means satisfiability checking. It is known that axioms like $C \sqsubseteq D$, (e.g. C is subsumed by D, "$\sqsubseteq$" denotes subsumption relations in DL) exists if and only if $C \sqcap \neg D \sqsubseteq \bot$. That is to say that $C \sqcap \neg D$ is not satisfiable (inconsistent). DL-based inference

Table 5.1: Comparison of some OWL inference engines

	RACER	**FaCT**	**Pellet**	**F-OWL**
Logic	DL	DL	DL	Horn Frame Higher Order
Reasoning support	OWL-DL	OWL-DL	OWL-DL	OWL-Full
Handling XML datatype	Yes	No	Yes	Yes
Decidable	Yes	Yes	Yes	No
Complete consistency checking	Yes (OWL-Lite)	Yes	Yes (OWL-Lite)	No
Interface	DIG Java GUI	DIG Command line	DIG Java	Java GUI Command line
Known limitation	N/A	No ABox support	N/A	Poor scaling

engines, which use tableau based algorithms [3, 2], are decidable and support complete consistency checking.

Some works on the OWL inference engine [155] are available from the Semantic Web research community and description logic community. Table 5.1 lists some of the popular engines (notations are the same as those presented in Table 4.2 Section 2.1.2). However, as we are interested in the graphic ontology editor Protégé, we have chosen RACER[1] (*http://www.racer-systems.com/*) instead of others. RACER can perform reasoning over RDF and DAML+OIL as a background reasoner. In our approach, RACER is used with Protégé as a user interface.

[1]RACER, the Renamed ABox and Concept Expression Reasoner [43]. RACER implements a TBox and ABox reasoner for the description logic $ALCQHI_{R+}(D)^-$ [42]. It can be regarded as: (1) a Semantic Web inference engine; (2) a description logic reasoning system capable of both TBox and ABox reasoning; and (3) a prover for modal logic Km. In the Semantic Web domain, RACER's functionalities include developing ontologies (creating, maintaining and deleting concepts, roles and individuals); querying, retrieving and evaluating the knowledge base, etc. It supports DAML+OIL and RDF.

RACER is typically started by opening a terminal/console window and starting the reasoner running with HTTP communication enabled (By default, RACER runs with the HTTP service enabled on port 8080.). If a class has been reclassified (i.e. if its super-classes have changed) then the class name will appear in blue in the *inferred hierarchy*[2]. If a class has been found to be inconsistent, its icon will be circled in red. See the tutorial at the website *http://www.co-ode.org/resources/tutorials/ProtegeOWLTutorial.pdf* for more details.

By using a DIG[3] compliant reasoner over the ontologies in Protégé, we have achieved the goal of checking the consistency of the integrated ontology by importing this ontology into the editor and running RACER.

5.8 Discussion

The integration mechanisms proposed here aim to provide flexibility and extensibility as much as possible in a dynamic environment where improvements on traditional computing models and paradigms are suggested. The characteristics of dynamic and open environments require built systems to operate effectively within rapidly changing circumstances. Interactions between heterogeneous systems are becoming more essential than ever before. In practice, agent technologies have become some of the primary weapons for addressing problems such as managing complexity. There is also an implicit and gradual realisation that agent technologies need to be exploited to achieve flexibility and extensibility. Table 5.2 (notations are the same as those presented in Table 4.2 Section 2.1.2) compares some of the most important characteristics of our agent-based integration with other systems and tools reviewed in Section 5.2 but with a focus on assisting integration. We extend the seven criteria outlined in Table 4.2 Section 4.8 to include **Reuse**. Additionally, we now distinguish interactions between systems and human users from those between participating agents. In this sense, the criteria are defined as follows:

[2]The class hierarchy that is automatically computed by the reasoner is called the *inferred hierarchy*.

[3]DIG (Description Logic Implementers Group), a DIG compliant reasoner provides the means to communicate via the DIG interface, which is a standard interface/protocol for talking to description logic reasoners.

Table 5.2: Comparison of proposed agent-based integration with some existing systems

System/Tool	Interaction	reuse
Our Agent-based Integration	Yes	Yes
Chimaera	between human user and the system	N/A
FCA-Merge	N/A	N/A
InfoSleuth	N/A	Yes
MOMIS	N/A	N/A
OBSERVER	N/A	Yes
ONION	N/A	N/A
OntoEdit	between human user and the system RDFS OIL	N/A
PROMPT	between human user and the system	N/A

- **Interaction**: Interactions between participants/agents play important roles for a certain goal. Moreover, rather than interacting with human users, we would like to distinguish interactions between different parts of systems and tools (agents).

- **Reuse**: This indicates that integrated ontology can be reused or not in the system it is derived from.

Table 5.2 clearly indicates that existing systems and tools have paid less attention to features such as interactions between participating actors/agents and ontology reuse. Compared with some available systems and tools, our agent-based integration has two unique features:

(1) Interaction between agents is essential and agent behaviour is conveyed through this agent interacting with other agents in the environment;

(2) The integrated ontology (deriving from existing varied sources) is ready to be reused in the system. Because of deployment of agent technology, an agent which is responsible for the integrated ontology (e.g. *OA*) will be generated

on the fly to take effect immediately. This allows easy modification whenever needed.

5.9 Summary

Ontologies are becoming more and more important in the context of the emerging Semantic Web. The need for integration and reuse of ontologies is increasing as well. In this chapter, we have presented ontology integration processes and mechanisms based on work in Chapters 3 and 4 to investigate ontology integration in the environment where similar ontologies exist. With the presence of `ontology agents`, newly generated ontologies can be reused immediately, which is in accordance with the intuition of reusing ontologies regardless of where the ontologies are and when they created/will to be created. Consistency checking has been considered and fulfilled by applying description logics and relevant tools.

In the next chapter, different agents under the proposed framework will be created to demonstrate ontology mapping and integration. Evaluation is also discussed.

Chapter 6

Prototyping and Evaluation

In Chapter 3, we have presented an agent-based framework. In Chapters 4 and 5, we have investigated ontology mapping and integration. In the discussion in the earlier chapters, it was argued that:

- Agent technology is one the most suitable paradigms for modelling complex systems;
- A MAS shows promise in solving dynamic, distributed and heterogeneous problems;

The proposed framework in Chapter 3 is general. We believe that the technical platform should be given much more attention for the reasons (1) to illustrate that agent technology is potentially adequate for ontology management (in an environment where ontologies reside in heterogeneous platforms with different ontology representations); and (2) to apply the proposed general-purpose framework in a broad variety of application scenarios without expending too much effort to consider platform compliant problems later on when needed.

In this chapter, the criteria for choosing an appropriate technology platform to meet the requirements is discussed in Section 6.1. Detailed ontology description is presented in Section 6.2. Different kinds of agents are thoroughly investigated in Section 6.3 . We highlight some important aspects of agents in our approaches. Prototype systems (ontology mapping and integration) are demonstrated in Section 6.4. Section 6.6 is about the lessons learned. Section 6.5 covers framework

evaluation. And finally, Section 6.7 summarises this chapter.

6.1 Implementation of the Framework

In order to realise the proposed agent-based framework and mechanisms of ontology mapping and ontology integration, several points should be kept in mind while choosing the technology platform. They are as follows:

(1) The platform should be independent, have open sources and comply with FIPA specifications[1].

(2) The architecture must be able to govern operations in a dynamic, distributed and heterogeneous environment.

(3) The platform must offer flexible and efficient messaging facilities between agents. By saying so, we focus on the semantics of the communication rather than the concrete syntax of the message.

(4) The platform should be easy to use.

With these criteria in mind, we next discuss why we chose JADE (*http://jade.tilab.com/*) [7] as the implementation platform. According to our survey, JADE is an appropriate technology platform that meets the above criteria.

6.1.1 Technology Platform

JADE is a software framework to facilitate the development of agent-based applications. JADE, complying with the FIPA standard model of an agent platform as shown in Figure 6.1 (*http://jade.tilab.com/doc/programmersguide.pdf*), runs all those mandatory agents (including Agent Management System (AMS), Directory

[1]FIPA, the standards organisation for agents and multi-agent systems, was officially accepted by the IEEE as its eleventh standards committee on 8 June 2005. Since its foundation in 1996, FIPA has played a crucial role in the development of agent standards and has promoted a number of initiatives and events that contributed to the development and uptake of agent technology. Furthermore, many of the ideas originated and developed in FIPA are now coming into sharp focus in new generations of Web/Internet technology and related specifications.

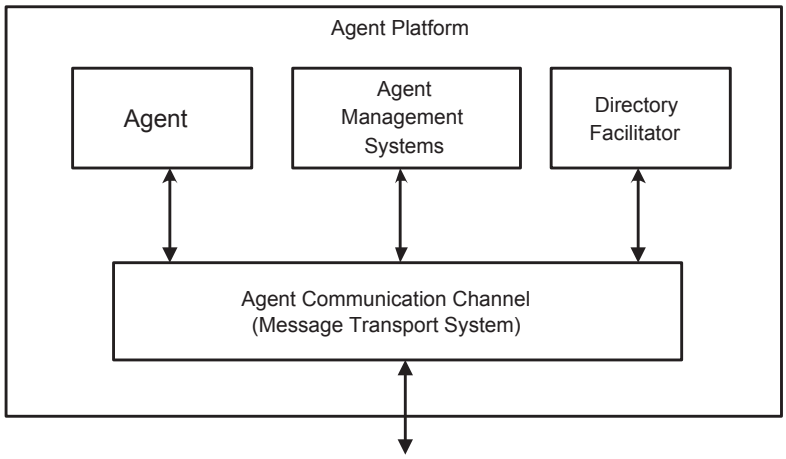

Figure 6.1: Reference architecture of a FIPA agent platform

Facilitator (DF) and Agent Communication Channel (ACC) to manage the platform. The major characteristics of JADE in response to the above criteria are listed below.

(1) JADE is claimed to comply to FIPA (*http://www.fipa.org*) specifications.

(2) JADE is a software framework fully implemented in Java language. It is an architecture-neutral language which is independent of the underlying hardware. Besides, it is an open-source project, complies with FIPA specifications, and includes all those mandatory components that manage the platform.

(3) In JADE, the agent platform can be distributed over several hosts. It also provides a complete run time environment for agent execution and allows several agents to concurrently execute on the same host. Moreover, JADE enables end-to-end interoperability between agents of different agent platforms.

(4) The communication architecture of JADE offers flexible and efficient messaging. All agent communication is performed through message passing, where FIPA ACL is the language to represent messages. It supports the full FIPA communication model (most pertinent to semantics of communicative acts).

(5) JADE provides a homogeneous set of APIs that are independent from the underlying network and Java version. Furthermore JADE API's are easy

to learn and use. JADE has been designed to simplify the management of communication and message transport.

Additionally, by using the well-known FIPA-compliant agent technical platform (e.g. JADE) to implement the prototype, commonly raised questions such as efficiency and management overheads will be solved with the presence of AMS, DF and other bundled tools to simplify platform administration and application development.

In short, although there some other agent tool kits/platforms (*http://www.agentlink.org/resources/agent-software.html*) are available, JADE[2] is one of the most suitable platforms so far according to our implementation criteria.

6.1.2 Agent Communication Languages

At an abstract level, agents achieve a goal by working with other agents to solve problems that are beyond individual capabilities and knowledge. Agents work together based on interactions. ACL is a FIPA agent communication language which is based on communicative acts of *speech act* theory (*http://www.fipa.org/specs/fipa00037/SC00037J.html*). It is treated as an agent communication language outweighing KQML [27] in which the semantics of ACL are rigorously defined. An ACL is a set of primitives (e.g. performatives) that allow an agent to state its intention to achieve by exchange of messages between agents.

ACL is an outer language that specifies message format and includes descriptions of the pragmatics, that is the communicative acts or intentions of the agents. ACLs are message formats as well as an "instruction" to describe an agent's intention in the communication. It allows flexibility in designing autonomous and heterogeneous agents in the implementation.

6.1.2.1 Agent Communication

Communicative acts are based on *speech act* theory, in which agents have, amongst their mental attitudes, a goal G and an intention I. Let us assume that individual

[2]Refer to the website *http://exp.telecomitalialab.com/* for more details of why JADE is a suitable technical platform for modelling distributed and heterogeneous environments.

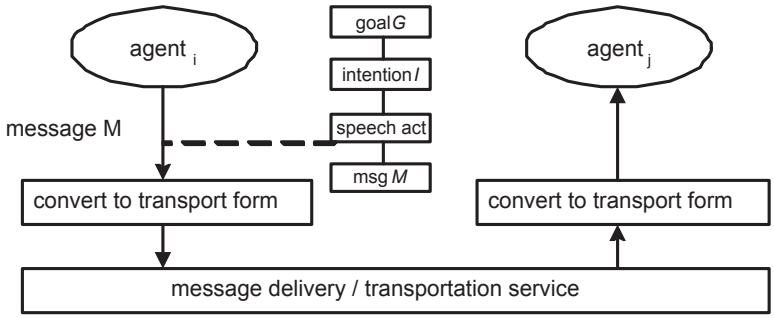

Figure 6.2: Message passing between agents from FIPA communicative act library specification

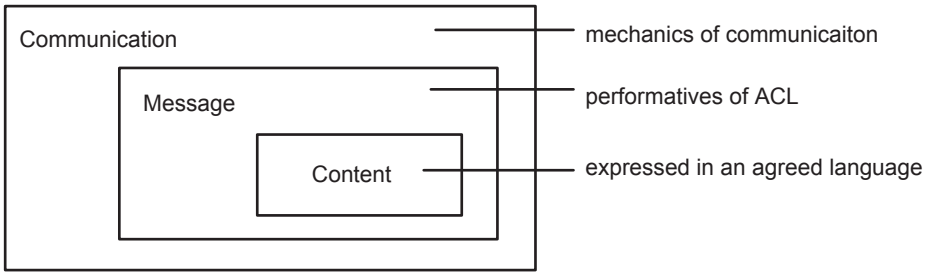

Figure 6.3: Three-layer agent communication model

agents cannot carry out the intention by themselves. These agents will contact other agents which will satisfy the intention and thus achieve the end goal if they behave rationally (Figure 6.2).

the agent communication language used in the thesis is FIPA ACL. In essence, an agent communication language provides a set of communication acts for agents in a MAS to perform. The purpose of agent communication is to convey information about an agent's own mental state with the objective of affecting the mental state of the communication partner. As some kind of shared ontology is the foundation in agent communications, we build an ontology with respect to accessing ontology structure (below we name it `meta-ontology`). Following the FIPA ACL (*http://www.fipa.org/repository/aclspecs.html*) communicative acts (*http://www.fipa.org/repository/cas.php3*), a three-layer model (Figure 6.3) of the FIPA ACL includes the *content* (layer) of the message; the *message* (layer) of particular attitude towards the content in the forms of performatives of ACL [150]; and the *communication* (layer) of the mechanics of communication. The concrete syntax

for FIPA ACL messages closely resembles that of Knowledge Query and Manipulation Language (KQML) (*http://www.cs.umbc.edu/kqml/*), but ACL language differs from KQML fundamentally, as FIPA-ACL has greatly enhanced the semantics of passing messages by eliminating any ambiguity and confusion from the usage of the language. Let us take the two most important performatives `inform` and `request` in the FIPA ACL as an example. They are shown in the following:
Example 1:

```
(request
   :sender i
   :receiver j
   :content "get_syn(beer)"
   :language fipa-sl
   :ontology meta-ontology)
```

In the case of this message, $agent_i$ asks $agent_j$ to perform the action of searching the term beer in the synonym file. The content of the message is in "fipa-sl" (e.g. FIPA Semantic Language), and the communication is based on the `meta-ontology`.
Example 2:

```
(inform
   :sender j
   :receiver i
   :content "syn(beer,suds)"
   :language fipa-sl
   :ontology meta-ontology)
```

In the case of this message, $agent_j$ wants $agent_i$ to believe that the statement that "beer" and "suds" are synonyms is true within the same ontology of Example 1.

6.1.2.2 ACL Semantics

The Semantic Language (content language of the FIPA ACL messages) is the formal language used to define the semantics of the FIPA ACL.

Practically, the semantics of `inform` are as follows:
$< i,\ inform(j, \varphi) >$
feasibility precondition (FP): $B_i\varphi \wedge \neg B_i(Bif_j\varphi \vee Uif_j\varphi)$

rational effect: $B_j\varphi$ where

1) $B_i\varphi$ means agent i believes φ or that it believes $\neg\varphi$;

2) $Bif_j\varphi \equiv B_j\varphi \vee B_j\neg\varphi$, namely that either agent i believes φ or that it believes $\neg\varphi$;

3) $Uif_j\varphi \equiv U_j\varphi \vee U_j\neg\varphi$, namely that either agent j is uncertain about φ or that it is uncertain about $\neg\varphi$;

Thus agent i sending an `inform` message with content φ that is either *true* or *false*, or that agent j believes whether φ is either *true* or *false*, or that agent j is uncertain of the truth or falsity of φ. If agent i is successful in performing the inform, then the recipient of the message (e.g. agent j) will believe φ.

Here is the semantics for `request`:
$< i, request(j, \alpha) >$
feasibility precondition: $FP(\alpha)[i\backslash j] \wedge B_i\ agent(j, \alpha)\neg B_i\ I_j\ Done(\alpha)$
rational effect: $Done(\alpha)$ where

1) $FP(\alpha)[i\backslash j]$ denotes the part of the FPs of α which are mental attitudes of agent i;

2) $agent(j, \alpha)$ means that agent j denotes the only agent that ever performs (in the past, present or future) the actions which appear in action expression α;

3) `intention` is defined as a persistent goal imposed on the agent to act. Thus I_jP means that "agent j has P as a persistent goal" and "I has the intention to bring about P". In the above formula, P is $Done(\alpha)$.

Thus agent i requests agent j to perform action α means that agent i believes that the agent of α is j, and agent i believes that agent j does not currently intend that α is done. The rational effect is that the action is done (e.g. which i wants to achieve by sending the message).

6.2 Ontologies

The purpose of this work is to develop a flexible system for agents, which commonly consume ontologies, engage in ontology management such as ontology mapping and

ontology integration. An agent, if it works with some other agents, clearly needs some kind of ontologies to communicate with others to know the meanings of the content expressions. Moreover, the agent is able to verify that messages it receives are meaningful pieces of information which comply with the specifications of the agent ontologies by means of which both the sender (agent) and the receiver (agent) agree upon the proper meaning of the messages. As distinct from user defined ontologies, herein, "agent ontology" denotes the internal ontologies, while "ontology" represents user defined ontology. There are various user defined ontologies with different representations. These ontologies come from different sources (different organisations in business). When cooperation among agents is called for, first of all, that these agents desire a variety of user defined ontologies is understandable. As this work is focused on using agent technology in handling ontology related management, we would prefer to emphasise operations over ontologies on a given content expression of an ACL message. The "agent ontology" is the ontology of operations over the user defined ontologies, it is called `meta-ontology`. Next, descriptions of user defined ontology and some relevant terminology are given.

6.2.1 User Defined Ontology

A three-layer model (Figure 6.4) has been developed to represent user defined ontologies in the system. This model allows different ontology languages and representations when instantiating various ontologies. The three layers are: E-R model, object, and ontology.

The top level E-R model is a general model used in modelling the reality at an abstract level. It is in accordance with Gruber's well known ontology definition [38] where *an ontology is an explicit specification of a conceptualisation*. As addressed in Section 3.1, an ontology is a specification, namely a pair of $< \Sigma, \Psi >$ to show that Σ satisfies the axioms Ψ derived from a domain model, where a conceptualisation Σ is a pair of $< \mathcal{C}, \mathcal{R} >$ with $\mathcal{C}$ representing a set of concepts, and $\mathcal{R}$ standing for a set of relations over these concepts. Under the *E-R model layer* (Figure 6.4) is the *object layer* where *id*, *label*, *attribute*, *datatype* and *relationship type* are defined. Objects in this layer are instances of E-R model layer classes. The bottom layer is the *ontology layer*. Various ontologies of the system are instantiations of objects

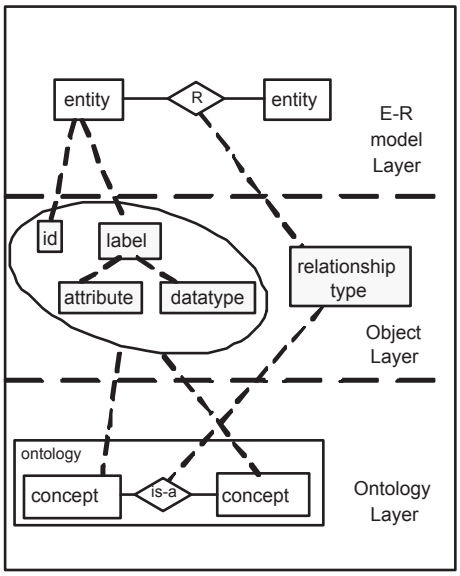

Figure 6.4: Three-layer ontology model

in the object layer.

A prominent feature of the ontology in this system is the *attribute* of an "entity" or a "slot". This is designed to take either simple data types or complex data types such as *class* type.

6.2.2 User Defined Ontology in System

When an ontology is imported to the system, an *OA* is automatically developed to act on behalf of this ontology and to perform ontology related operations for it such as accessing and updating the ontology. For example, once the four running examples (Section 4.6) are imported, four different *OA*s run to act on these four ontologies accordingly.

6.3 Agent Design and Implementation

In this section, the interrelated processes and functionality together with the implementation of each agent (Figure 3.1) in the JADE agent platform are presented. For simplicity, as indicated in Chapter 1, the implemented system is called JOMI (Jade Ontology Mapping and Integration).

6.3.1 User Agent

The *UA* assists the user in formulating his/her requests, posting queries (e.g. tasks) to the proposed system via the *IA*, and visualising the required results according to the user's requirement. The *UA* isolates complex internal system designs and implementations from the user. Upon initialisation, it waits for information of user actions via the `user interface`. The *UA* must engage in "communications" with the *IA* in fulfilling its role because the internal structures are also invisible to it. The *UA* only knows the *IA*. It does not know any other agents we have created in our MAS environment.

This *UA* is started up with the main container when we start up JADE. It waits for actions from the `user interface` then sends an ACLMessage to the *IA* when it has received a user action. For example, once the user selects two agents in the `mapping interface`, then it performs the mapping module. The `user interface` passes the agents to the "start mapping" method in the *UA*. The *UA* does not know how to map, therefore, it creates an ACLMessage and adds these two *OA*s as parameters in the ACLMessage. After that, it sends this ACLMessage off to the *IA*. When information is to be displayed to users, the *IA* passes a message back to the *UA* which shows the necessary results on the `user interface`.

6.3.2 Interface Agent

The *IA* acts as an interface between agents in our MAS and the *UA*. Upon initialisation, every agent knows about the *IA*. So if any other *FA*s, for example, the *MA*, want to show mapping results to the user, they can send a message to the *IA* which will pass it on to the *UA* to display. On the other hand, when new agents, for example *OA*s, are added in, the *IA* will let all other agents know. So existing agents may refer to newly added agents.

This *IA* is started up with the main container when we start up JADE. It receives messages from the *UA*, then based on the *conversation id* of the ACLMessage, it passes it on to the appropriate agent in our MAS to deal with. For example, the *IA* may receive a message from the *UA* with the *conversation id* of "start-mapping". The *IA* does not know anything about mapping but knows when it gets

this message to pass it on to the *MA* to handle. The *MA* then extracts the content of the message and performs mapping on these two *OA*s.

6.3.3 Ontology Agent

The *OA* provides ontology related information and operations to other agents. The *OA*s isolate details of external ontologies from *FA*s.

Upon initialisation, it waits for information from other agents. It engages in "communication" with these agents with respect to the required information. This process may loop until the module runs out of concept candidates of a specified ontology. The *OA* may return a result to the user via the *IA* and the *UA*.

The *OA* provides as much information about the ontology it acts on as possible. The *OA* operates over ontology structures on behalf of which it acts. Interactions among agents, specifically between *FAs* and *OAs* are shown in Figure 6.5. The major operations of the *OA* are described as follows.

- *Insert*, namely insert a new concept into the ontology structure as requested. For example, the *UA* may specify where to attach the concept (for ontology integration phase).
- *Traverse*, namely traverse the structure, return the requested concepts or *true/false* of a given proposition.
- *Delete*, namely delete out-of-date concepts and corresponding relations as required.
- *Update*, namely update the labels of concepts and relations as required.

Furthermore, an *OA* acts correctly when triggered by different sources of driving forces. For example, in the refinement module, it has capabilities to collabrate with the *NA* to modify (e.g. insert, delete, update) its acting ontology. For another example, in a query module, an *OA* will dispatch a particular query request to other known *OA*s if it is unable to answer the query. We focus on ontology mapping related interactions, by taking into account the interactions between the *OA* and the *MA*, and the *OA* and the *QA* as well.

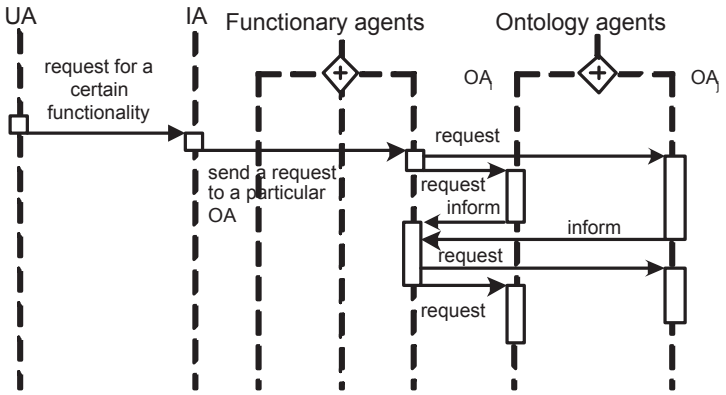

Figure 6.5: Interactions from *OA*'s view

An *OA*'s main responsibility is to look after an ontology, either the one imported from a RDF(s) file or the one created in an object oriented way. Ontology agents are all equipped to be able to perform some functions to reify agent behaviours responding to in coming ACLMessages.

The *OA*s are not added to JADE at start up. They are added to our current running container when the user asks for them via the `user interface`. For example, suppose that the user would like to add some *OA*s, for instance two of them, to our current running container of JADE. When these agents first enter our system, they register themselves with the *IA*. Then all agents will have a reference to these two added *OA*s via the *IA*.

6.3.4 Thesaurus Similarity Agent

The *SA* maintains a thesaurus for the purpose of similarity. Upon initialisation, it waits for information from either a *FA* or the *UA* with respect to the query or updates the contents of the thesaurus. It then returns the corresponding results. The *SA* may work in two cases by interacting with other agents (Figure 6.6). One is in the mapping module when the *MA* is looking for synonyms for a given term from ontologies if the *MA* has found no such concept in other ontologies. Another case is when the user asks to update the thesaurus list.

We assume that the *SA* holds a list of common words and synonyms of words. This agent is started up with the main container when we start up JADE. For

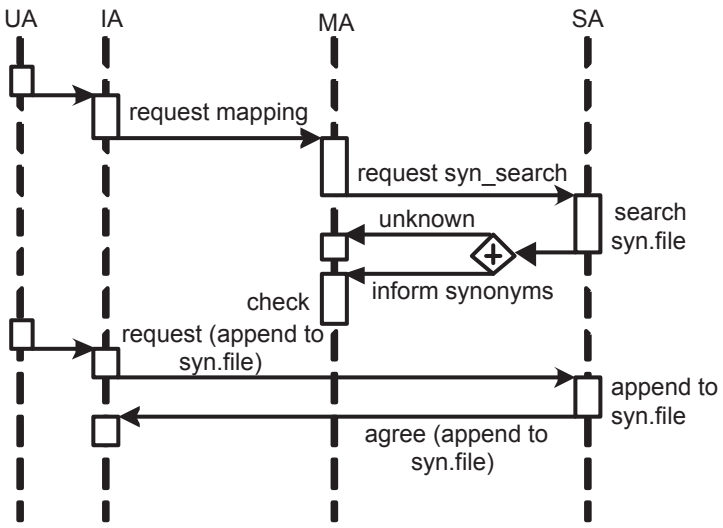

Figure 6.6: Interactions from *SA*'s view

example, during the mapping process, if the *MA* is trying to perform mapping on two ontologies, and gets the concept *suds* from one ontology, and wants to map it to another ontology, but that ontology has no concept called *suds*, then the *MA* sends an ACLMessage to the *IA*. The *IA* reads the *conversation id* of this message and passes it on to the *SA*. The *SA* then looks in its list and gathers a list of synonyms for the particular term. The *SA* then sends this list back in an ACLMessage to the *IA*, which passes it back to the *MA*. The *MA* will then use this list for mapping.

6.3.5 Mapping Agent

Dynamic mapping is thought to be on the right track to pave the way for further ontology operations. The *MA* takes effect on receiving a mapping request from the *UA* via the *IA*. It engages in "communication" with *OA*s and the *SA* to execute mapping until the process is completed. *OA*s will have a reference to the mapping results.

Unlike the *OA* or other agents, the *MA* does not operate directly over existing structures. It takes effect via *IA* in the module (see Figure 4.2) in deciding whether existing ontologies come from the same domain or not; or in the module of ontology mapping (see Figure 4.3) through the *OA* and the *SA* to acquire relevant information. The former paves the way for deployment of predefined rules (actually, these

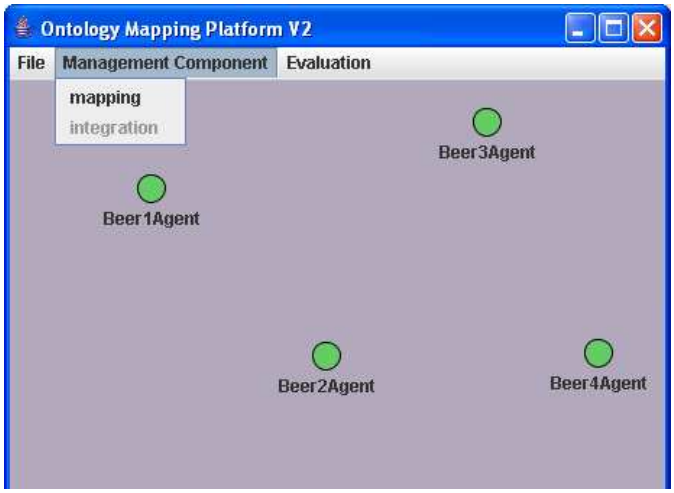

Figure 6.7: Mapping main window

rules are the required little prior knowledge). The latter depicts agent interactions in the ontology mapping process.

In reality, the process of deployment of rules may loop until it runs out of sub-concepts of a specified ontology. Mapping is conducted from a particular *OA*'s perspective. In other words, mapping has directions.

This agent is started up with the main container when we start up JADE. It performs mapping on pairs of ontologies and writes the mapping results out to a file (e.g. `mapping.txt`). We take the example below to illustrate the entire process.

When the user wants to perform mapping between two ontologies, the user inputs two ontologies. The `user interface` then sends these two ontology names to the *UA* which wraps them up in an ACLMessage and passes it on to the *IA*. The *IA* will pass the message on (based on *conversation id*) to the appropriate agent to handle. Here the *MA* will then extract the content of the message and perform mapping on these two specified *OAs* (acting on these two ontologies).

Figures 6.7, 6.8 and 6.9 are screenshots. Figure 6.7 is the `mapping main window` when "mapping" is selected. Figure 6.8 is the `mapping interface` for the user to key in some information. Figure 6.9 is the `mapping result` based on information from the `mapping interface` when "OK" is clicked. The mapping results indicate which concept from one ontology is the semantic equivalent to a concept from another ontology after the mapping module has been executed, for instance,

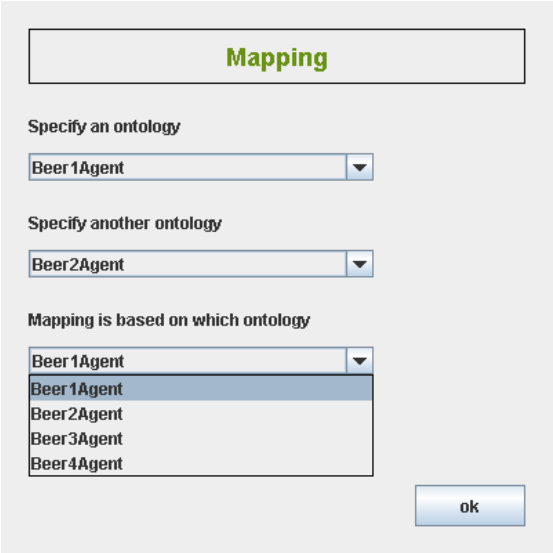

Figure 6.8: Mapping interface

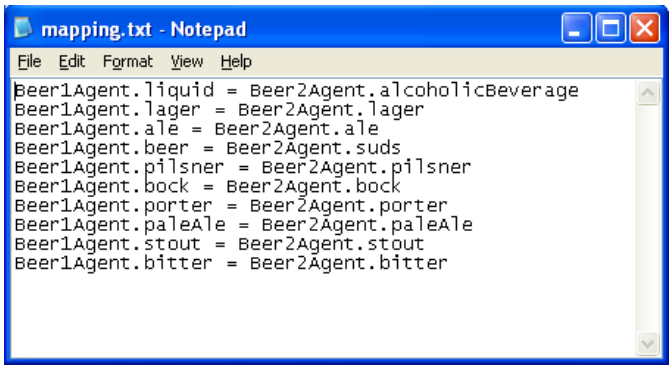

Figure 6.9: Mapping result

`Beer1Agent.beer=Beer2Agent.suds` from Figure 6.9. Later on when performing ontology integration, the integration module knows that `Beer1Agent` beer equals `Beer2agent` suds.

6.3.6 Integration Agent

Ontology integration is based on the results of ontology mapping. It is from a particular agent's perspective. In other words, the system has no intention of being engaged in conceptual modelling issues (e.g. how to build an ontology). Instead, it uses the existing conceptual models by choosing one of them. The purpose of

integration is to abstract the existing ontology from a global view (according to a certain scenario).

Upon initialisation, the *InA* waits for the action of performing integration over specified ontologies. The *InA* then engages in "communications" with proper *OA*s until the integration task has been solved. The integration module executes integration from a given start point of an ontology. It then requests corresponding *OA*s to traverse sub-concepts of the ontology. Briefly speaking, the *InA* counts the appearance of each concept in existing ontologies, and then filters unexpected concepts with a given threshold. By saying this, we do not mean that we are attempting to change the conceptual modelling of the ontology. Instead, ontology integration is based on a specified ontology.

This *InA* is started up with the main container when we start up JADE. We take the example below to illustrate the entire process.

When the user wants to integrate the ontologies they provide the dominating ontology and a threshold. The `user interface` passes this information on to the *UA* which wraps it up in an ACLMessage and sends it to the *IA*. The *IA* looks at the *conversation id* of this message then passes it on to the *InA*. The *InA* will request the ontologies from available *OA*s (this is done again via ACLMessages). When the integration module has all the ontologies (by contacting the *OA*s), it performs the integration module based on the dominant ontology, the mapping results (e.g `mapping.txt`), and of course the given threshold for a better outcome. When the integration has been done, firstly an *OA* is created on the fly which will look after the newly integrated ontology. Then the *InA* sends the new ontology to the *IA* in an ACLMessage. The *IA* then passes it on to the *UA* which will display the new ontology to the user in a tree view. The user can later export this new ontology to RDF(s) format.

Figures 6.10 to Figure 6.13 are screenshots. Figure 6.10 is the `integration main window` when "integration" is selected. Figure 6.11 is the `integration interface` for the user to key in some information. Figure 6.12 is the "integrated ontology" based on information from the `integration interface` when "OK" is clicked. It is in hierarchical structure. Figure 6.13 is the "integrated ontology" in RDF(s) format.

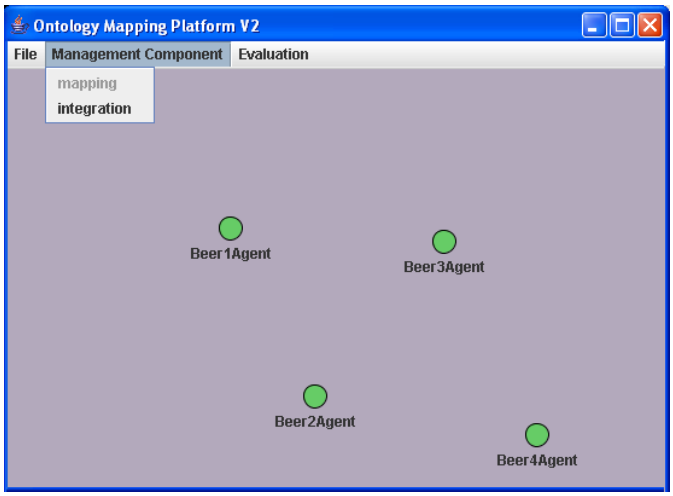

Figure 6.10: Integration main window

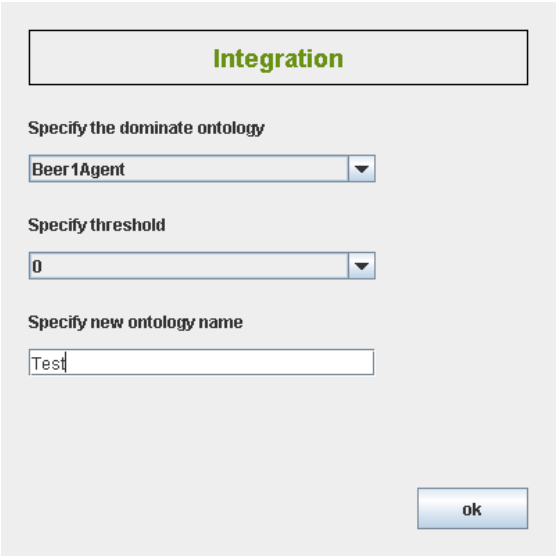

Figure 6.11: Integration interface

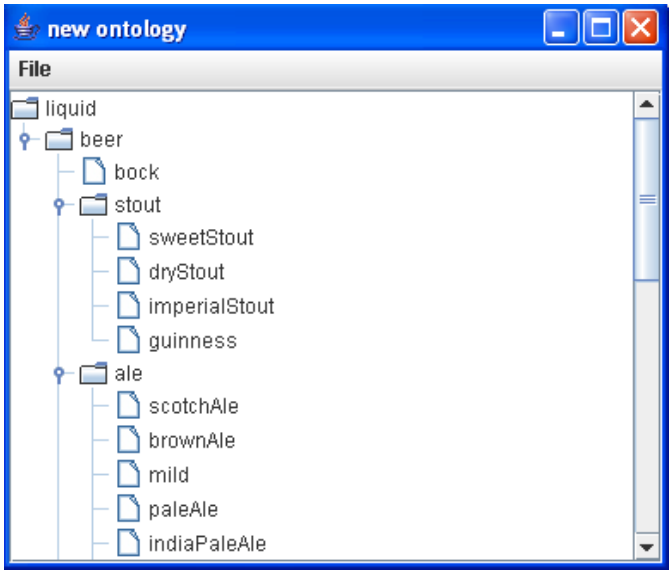

Figure 6.12: Integrated ontology

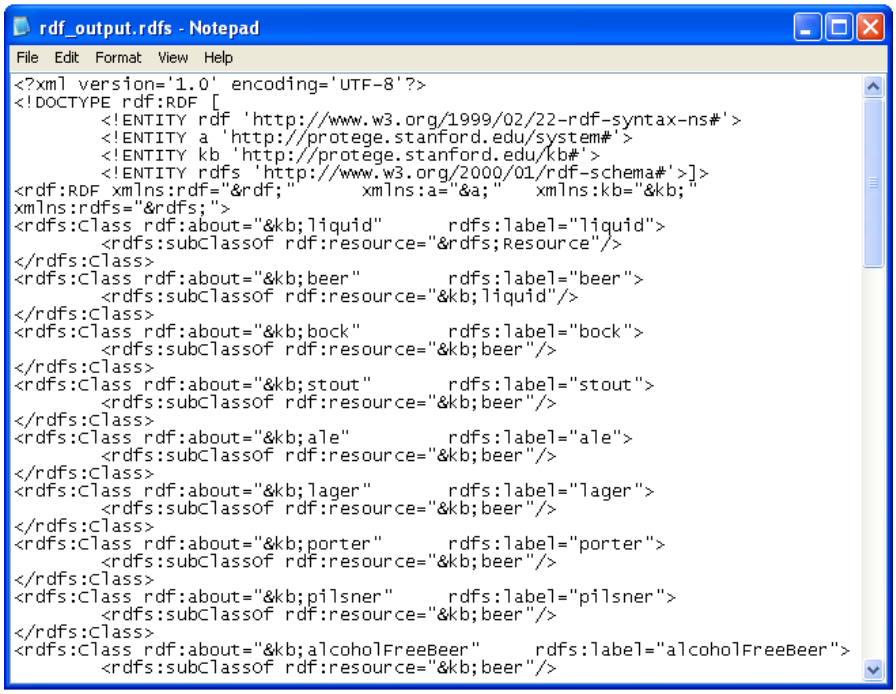

Figure 6.13: Integrated ontology in RDF(s) format

6.3.7 Checking Agent

The *CA* is developed to check the consistency of the integrated ontology (assuming all given ontologies are consistent at the beginning). Upon initialisation, the *CA* waits for actions from the `user interface` to perform the consistency checking task upon a generated ontology. The *CA* then executes the checking module by engaging in "communications" with a particular *OA*. This process may loop until no candidate concept from the ontology is left. When the *CA* (Figure 6.14) initialises, the *CA* waits for a request to perform checking task over a specified ontology. After the process is completed, the module returns the user either (a given ontology) consistent or not. If the ontology is inconsistent, the problem part of the ontology is highlighted in the `user interface`.

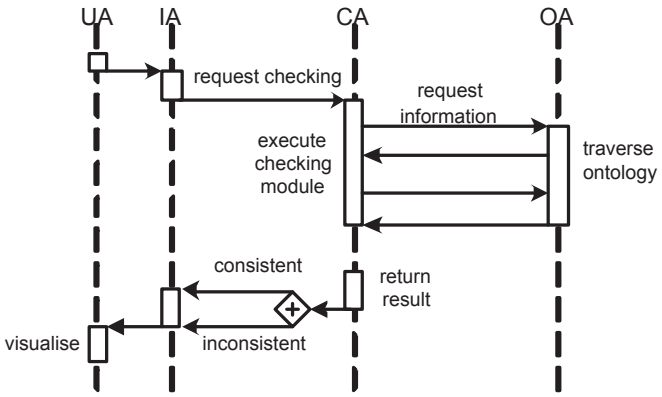

Figure 6.14: Interactions from *CA*'s view

This agent is started up with the main container when we start up JADE. The *CA* gets an ACLMessage from the *IA* to check the consistency of an ontology. It sends a message back stating that inconsistencies or no inconsistencies were found, which will eventually be passed back to the `user interface`. Figures 6.15 and 6.16 are screenshots. Figure 6.15 is the `checking main window` when "consistency check" is selected. Figure 6.16 is the status of consistency of the integrated ontology.

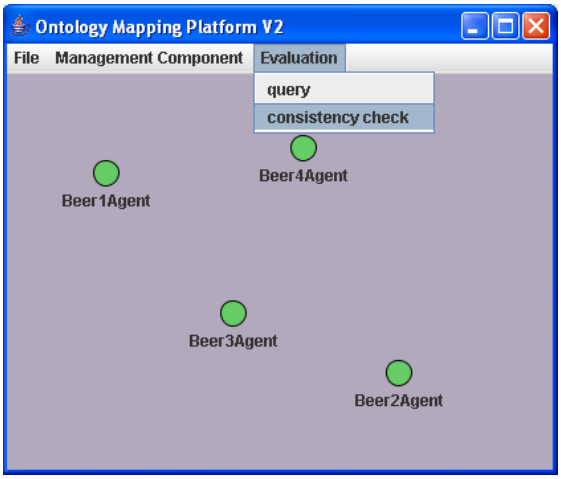

Figure 6.15: Checking main window

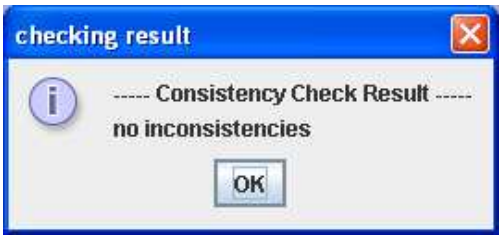

Figure 6.16: Consistency check result

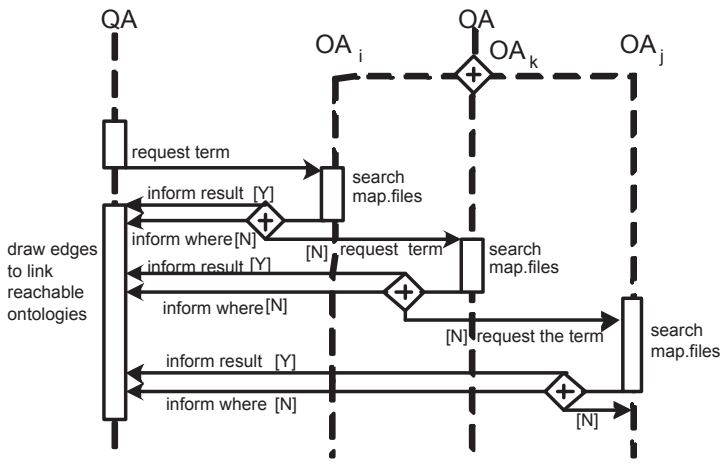

Figure 6.17: Interactions from *QA*'s view

6.3.8 Query Agent

The *QA* is designed to govern query execution around available *OA*s over the mapping results. The query aims to give an equivalent description of the queried term if the agent knows it. If not, the agent then passes the query to other available *OA*s. The query dispatching module is continually executed until either the query is answered or the process has queried all existing *OA*s with respect to the term. The result (e.g. a query routing) is displayed on the `user interface`.

Upon initialisation, the *QA* waits for actions from the `user interface` to perform the query. The *QA* then executes the query module by engaging in "communications" with *OA*s with reference to mapping results (e.g `mapping.txt`). This process is completed when the queried terms is interpreted.

The query agent acts as a planning agent not only for an optimal route but for a reachable path in a decentralised ontology network. It is responsible for tasks such as keeping track of *OA*s in case backtracking is required (Figure 6.17).

The *QA* is started up with the main container when we start up JADE. This agent gets an ACLMessage from the *IA* to query an ontology of a specified term. For example, when a user performs a query, the user inputs the query term and the ontology to query. The `user interface` will pass the query term and the ontology to the *UA* which will wrap up this information in an ACLMessage and send it to the *IA*. The *IA* will look at the *conversation id* of the message and realise that it

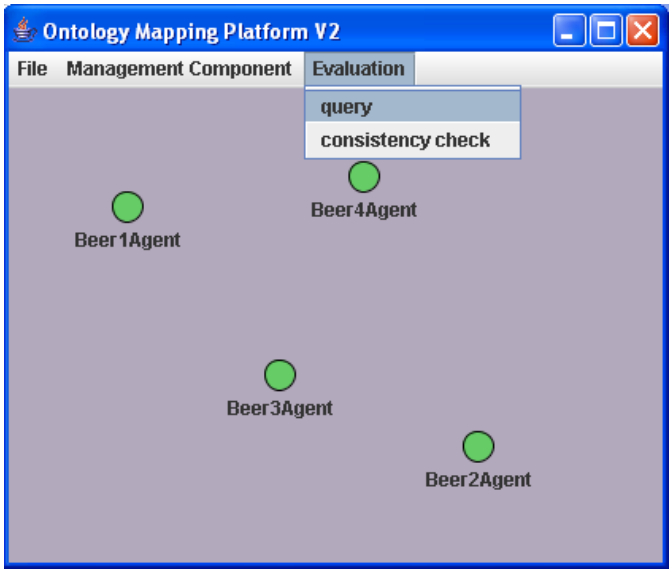

Figure 6.18: Query main window

needs to send it to an *OA* itself. So the *IA* will unwrap the message to determine which agent it needs to send it to. The *IA* then sends the ACLMessage to that *OA*. Each *OA* has been programmed to handle this kind of message. So when an *OA* receives this message it searches through its own ontology for the query term. If it is unsuccessful, it then searches through appropriate mapping files. It will then pass a result message back to the *IA*, which will pass it on to the *UA*. Then the user will see a message posted on screen either something like `suds = BeerAgent1.beer` or `no semantic equivalence`.

Figures 6.18 to 6.20 are screenshots. Figure 6.18 is the `query main window` when "query" is selected. Figure 6.19 is the `query interface` for the user to key in some information. Figure 6.20 is the "query result" based on the user's input information in the `query interface`.

6.3.9 Ontology Import and Export

6.3.9.1 Import RDF(s) from Protégé

Importing an RDF file (ontology) from Protégé into JOMI RDF format file works like a glue to join JOMI to Protégé. It is possible to create an ontology in Protégé without spending too much effort in considering ontology integrity and consistency

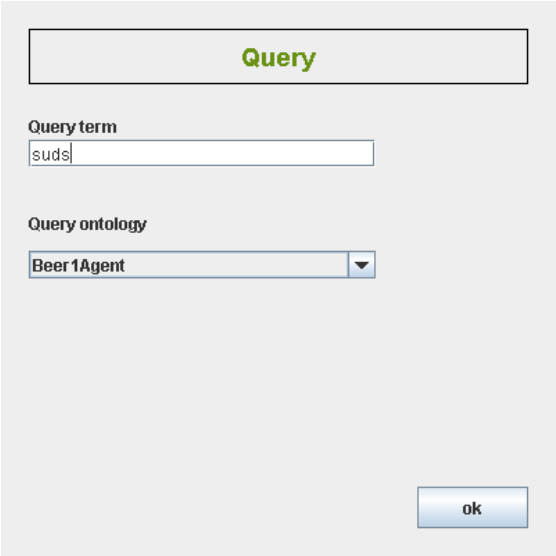

Figure 6.19: Query interface

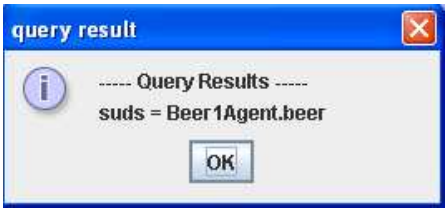

Figure 6.20: Query result

during the ontology creation phase. In other words, the prototype JOMI allows both object-oriented ontology and Protégé ontology in RDF(S) format (importing a RDF file from Protégé).

Once a RDF file has been exported from Protégé and saved in a specified folder, JOMI can access it. When a RDF file is specified as an imported ontology, the JOMI imports this RDF(s) file instead of the usual Java file. At the same time on the screen, this imported RDF(s) file is represented in a different colour in contrast to other Java ontology files. What happens in the background is that a Java file is created for the RDF(s) file and stored in a particular folder. For example, if you had a RDF schema file called *test.rdfs* which was stored in a specified folder when you import it, there will now be a file called *test.java* in a particular folder. Then the *test.java* file is added to the current running JADE container.

6.3.9.2 Export Ontology in RDF(s)

When JOMI has integrated a new ontology, it may be exported in RDF(s) format. The export interface is shown in Figure 6.21. From the dialog, there will be an RDF(s) file which is called *test.rdfs* created in a specified folder. An RDF(s) file will be created able to be imported into Protégé for consistency checking.

Figure 6.21: Export interface

6.3.10 Important Notes

All our agents make use of JADE Message Templates to perform certain tasks based on certain attributes of the message. For example, an *OA* can send concepts. When the *OA* receives a message, it needs to determine which behaviours to execute. Message Templates allow us to filter messages based on attributes like *performative type*, *conversation id*, and *sender*, etc.

6.4 Ontology Mapping and Integration

The JADE agent platform is used to build our MAS system. Following the analysis of the proposed architecture, we have worked out the details of the prototype. To meet the main tasks of the prototype, different classes (e.g. *concept class*, *ontology class* and *agent class*) have been defined to enable ontology and agent generation on the fly. The prototype consists of the following agents: one `user agent` (*UA*), one `interface agent` (*IA*), one `mapping agent` (*MA*), one `integration agent` (*InA*), one `consistency checking agent` (*CA*), one `query agent` (*QA*), four `ontology agents` (*OA*s). In the prototype, different ontologies, for example RDF(s) ontologies, are created for demonstration. In the following, two major ontology management - ontology mapping and integration will be demonstrated in detail.

6.4.1 Ontology Mapping

As working together in a business environment is becoming inevitable with business globalisation, it is necessary that agents should be able to collaborate. But the fact is that there is often more than one ontology designed by different orgaisations in a given domain. Since agents need to cooperate in a MAS, there has been an increasing interest in dynamic ontology mapping. It is seen as a feasible and effective approach to achieve ontology interoperability regardless of ontology representations and hosting platforms.

In our work, mapping attempts to provide mapping results for any further ontology operations, for example, ontology integration. Mapping is operated between

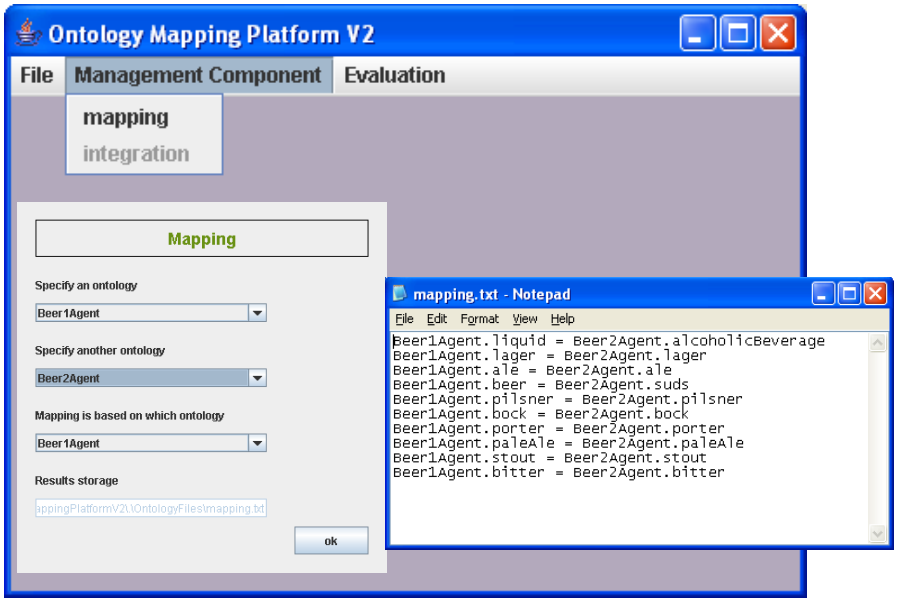

Figure 6.22: Screen shot of ontology mapping

pairs of ontologies based on basic mapping rules (Section 4.4).

The mapping module in the prototype runs as follows:

(1)*Import* existing ontologies;

(2)*Develop* corresponding *OA*s for each available ontology;

(3)*Develop* corresponding *FA*s (e.g *MA*);

(4)*Execute* the mapping algorithm as described in section 4.5.

The above mapping module may start from the top of the ontologies, it may also start from particular parts of ontologies. It may loop until it runs out of concepts of ontologies according to a scenario.

Figure 6.22 is a screenshot of the ontology mapping and the demonstration of the mapping process. The window at the back is an overall prototype when "mapping" is selected. The lower left part is a mapping screen where the mapping direction is specified in addition to two given ontologies. The mapping result in plain text is shown in the lower right of Figure 6.22.

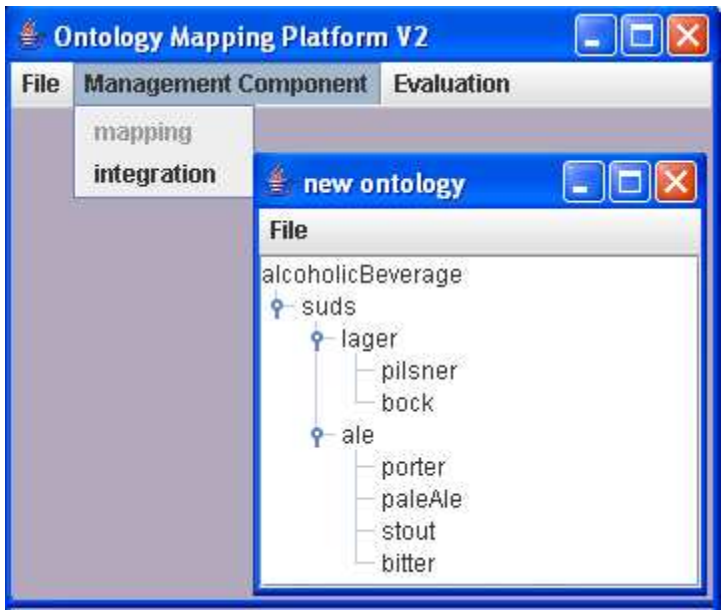

Figure 6.23: Screen shot of ontology integration

6.4.2 Ontology Integration

Ontology integration is based on ontology mapping results in order to provide a global view of existing ontologies in the environment. Since ontology conceptualisation is out of our scope in this work, we will not extend this topic any further but instead we will choose one available ontology to perform ontology integration.

The prototype runs as follows:

(1)*Import* existing ontologies;

(2)*Develop* corresponding *OA*s for each available ontology;

(3)*Execute* mapping module;

(4)*Develop* corresponding *FA*s (e.g *MA* and *InA*);

(5)*Execute* integration algorithm as described in Section 5.5;

(6) *Visualise* the integrated ontology;

(7)*Export* the integrated ontology in a specified format (e.g. RDF(s)).

Figure 6.23 is a screenshot of the ontology integration and integration results. The upper part is the overall prototype when "integration" is selected, while the lower right part is the integrated ontology in the hierarchical structure. See Figure 6.13 for RDF(s) format export.

6.5 Evaluation

Ontologies and ontology-based applications perform in the environment of dynamics, distribution and heterogeneity. The agent-based framework proposed in this thesis is suitable for tasks such as ontology mapping and integration in a certain business scenario. By adopting a MAS perspective, interactions among multiple agents, which work together to achieve the goal beyond individual capabilities and knowledge, are highlighted. In other words, MASs are able to take a variety of environmental circumstances into consideration rather than mainly treating the environment as being static. The evaluation work takes the following characteristics into account:

- **Flexibility**: In the framework, the already set up agent communication channel facilitates message delivery. Moreover, the presence of the *OA*s allows for flexible system organisation. The system allows freely adding/deleting *OA*s and *FA*s (including all defined agents for a particular task) to/from the system;

- **Interactivity**: Agents are highly interactive in the framework. Interactions take place not only between *OA*s and *FA*s, but also between *FA*s if a particular task needs to deploy the functionalities of other *FA*s;

- **Interoperability**: The framework enables interoperability between agents of different agent platforms. In terms of syntactic and semantic heterogeneity of ontologies, a `meta-ontology` is developed to resolve semantic heterogeneities;

- **Scalability**: In the framework, different classes are developed. They include a *concept class*, an *ontology class* and an *agent class*. Moreover, all ontology related operations are encapsulated and isolated from the *FA*s' view. By extending corresponding classes, the *OA*s and *FA*s can be created easily;

- **Reusability**: In the framework, whenever a new ontology is generated based on existing ontologies, an *OA* is developed accordingly. This enables the general view of a particular application domain to be reused in the system;

- **Reliability**: this depends on agents performing rationally in the framework. As every agent of the system is required to register and advertise its capabilities to the *IA*, any other agents are able to reach all available capabilities in the system whenever needed. Moreover, the upper bound on the number of iterations (the developed agent algorithms) required to reach a fixed point is the number of concepts in an ontology. It is known that the number of concepts in an ontology is finite.

To sum up, the proposed framework provides a flexible and effective modelling approach to tackle ontology mapping and integration over a variety of ontologies.

6.6 Lessons Learned

(1) *Problems we encountered in developing the framework using JADE*

JADE provides a good platform for developing multi-agent systems but a major problem we encountered was with JADE's *Ontology Class*. In our prototype we were mainly looking at performing mapping and integration on two or more ontology's whereas JADE's *Ontology Class* looks at ontologies only at an instance level which is used for communication purposes between agents.

(2) *Our solutions*

We came up with a solution but it took quite a lengthy time to implement. Basically we had to create our own *ontology class* and provide methods that would allow us to change the dynamics of the ontology. For instance, we included methods to allow the user to add concepts, sub concepts, add slots to concepts, return concepts, and return slots from a concept. We found the easiest way to do this was to extend Java's JTree Class because this class already provided us with the most functionality in accessing things in a hierarchal structure.

(3) *Enhancement of JADE*

From our point of view, maybe JADE could provide two representations of their *ontology class*. One for the instance level and agent communication,

and another for an abstract view to be used to perform certain tasks on the ontology. Something that needs to be addressed in JADE is that sometimes it runs at the back of other programs and therefore needs to provide a method for terminating itself from another program. For example, when we have been running JADE in our program and close our program, JADE still has a process running. JADE might provide for this but we were unable to find any documentation about it.

6.7 Summary

A variety of ontologies from different sources exists in a dynamic and heterogeneous environment. Existing systems or tools have mainly treated the environment statically. Agents in a multi-agent system are autonomous and flexible, given the agents are capable of perceiving changes in the environment and responding promptly. Agent technology is thus suitable for ontology mapping and integration. In this chapter, an agent-based prototype has been implemented to demonstrate our work discussed in previous chapters (e.g. Chapters 3, 4 and 5). In order do this, the JADE agent platform was deployed. We have detailed each kind of agent and its implementation in JADE. Two major ontology managements - ontology mapping and integration are demonstrated in the prototype.

Additionally, Protégé (*http://protege.stanford.edu/*) is an ontology editor with incremental plug-ins. Although ontology building is out of the scope of this work, we expect it will be possible for various ontologies with different representations and languages to be imported into the agent-based system. RDF(s) is such a channel to link our agent-based system with Protégé. Actually, the agent-based system is designed to be able to import and export ontologies in RDF(s) format. Some ontology related work such as ontology verification may thus be moved to Protégé by taking advantage of a number of plug-ins to support this.

Chapter 7

Agent-based Ontology Refinement

In Chapters 4 and 5, we have investigated agent-based ontology mapping and integration. And in Chapter 6, we have described the prototype to realise the framework proposed in Chapter 3, in which ontology mapping and integration related work (e.g. ontology mapping and query, ontology integration and consistency checking) discussed in Chapter 4 and 5 have been implemented. Based on the work we have done so far, it is clear that ontology mapping and integration benefit from deploying agent techniques. Certainly, ontology is the conceptual backbone that provides meaning to data on the Semantic Web. The vision of the Semantic Web can only be realised through proliferation of well-known ontologies describing different domains. However, ontology is not a static resource and may evolve over time, especially where different domains are concerned. Ontologies are rarely static, but are constantly being adapted to changing requirements. In fact, they are subject to continuous change. Changes in the domain, conceptualisation and explicit specification can cause changes in an ontology [110]. In detail, changes can take place as follows:

- New concepts need to be added to the ontology.
- Outdated concepts need to be removed from the ontology.
- The common ontology (in integration) is required to change (add/delete /update) with the changes brought by new information resources.
- Better ways of organising information are available.

In a nutshell, ontology evolves with changes in the environment. Hence, ontology evolution can be defined as the timely adaptation of an ontology and consistent propagation of changes to the dependent artefacts. The complexity of ontology evolution increases as the number and the size of ontologies grows, so an insight into potential solutions in ontology refinement is required (herein, ontology refinement is used to provide mechanisms along with *NA*s, which are responsible for agent negotiations in the environment). We will not distinguish between agents and process when agents commit to processes in this chapter.

To this end, this chapter is designed to tackle ontology change from a MAS perspective. It is an extensive chapter with a focus on individual ontology in contrast to previous chapters (e.g. Chapters 4, 5 and 6). In this chapter, we first identify problems in ontology refinement in Section 7.1 followed by specific related work in Section 7.2. Based on the introductions to the negotiation model in Section 7.3, the process algebra is outlined in Section 7.4 followed by the utility function in Section 7.5. Then the scope of ontology refinement is presented in Section 7.6. After that, the refinement process and refinement mechanisms are thoroughly exploited in Section 7.7 and Section 7.8, respectively. A case (based on a VO) is presented in Section 7.9 to demonstrate how agents reach an agreement by means of negotiation in terms of process algebra to describe information exchange between agents. Discussion is then presented in Section7.10. Finally, we summarise this chapter.

7.1 Problems in Ontology Refinement

A changeable environment enforces underlying ontologies to evolve over time. For this reason, ontology refinement needs to look in depth of the requirements of dynamic changes in the environment and rapid development of the Web. The research in ontology evolution is in its very early stages [110]. Undoubtedly, there are some critical issues that ontology evolution research needs to address before the proliferation of ontologies is made possible in the field such as supporting semantics-based search, interoperability support and Semantic Web applications. These issues are as follows:

- Addressing requirements of environmental (to host ontologies) change;
- Reflecting environmental change and guiding corresponding ontology refinement accordingly; and
- Developing an ontology refinement architecture.

The very viable and rapid growth of the Web has made ontology change even more prominent than ever before. The timely addressing of ontology refinement and the provision of a conformance view for agents involved in a certain scenario are thus seen as the foundation for further operations (as shown in Figure 1.1, ontology change may affect ontology mapping and integration). Therefore, a much more detailed analysis of ontology evolution and supporting architecture becomes a higher priority on the agenda of research into ontology evolution. In this chapter, we attempt to provide potential solutions to some of the above mentioned challenges.

Although procedure of reflecting ontology changes and taking appropriate actions by agents in a MAS is an essential requirement for successful application of ontologies (taking ontology evolution into account), methodologies and tools to support this complex task are largely missing. Among available techniques, as discussed in previous chapters (Chapters 2, 3, 4, 5 and 6), agent technique is identified as a suitable way of coping with ontology change in a dynamic and heterogeneous environment. Agent interaction, a potential solution for reflecting dynamic changes in an environment, is seen to be promising in targetting problems such as what information is known by agents in a MAS and how interactions can affect agents' succeeding actions. More importantly, interactions facilitate understanding between agents (semantically) through provided ontologies.

Given multiple ontologies, finding a conformance view for all participating agents seems impossible in advance as far as heterogeneity, distribution, autonomy and evolution are concerned. The run-time interaction is needed in this regard. In other words, by agents taking part in the ontology management at run-time, we can expect that the agents will be able to make decisions on how ontologies evolve by themselves rather than by specified rules in advance, because few a priori defined rules will apply given that changes are unpredictable.

Our contribution lies in that we propose that dynamic semantic understanding in a business scenario be supported by means of interactions between self-interested but rational agents. To comply with the requirement of run-time setting up association for understanding a particular issue, our approach is summarised as follows: (1) to describe and analyse interactions between agents but with a focus on ontology refinement on the basis of process algebra; (2) to discuss and develop a decision making strategy on the basis of negotiation between agents in the event of collaboration and competition both existing.

It is worth noting that we attempt to provide a flexible solution by modelling agents' interactive and negotiated activities in a collaborative environment. In saying so, our assumption is that ontologies of different organisations are available. In addition, this is different from "ontological commitment" which focuses on the situation of an agent committed to ontology when its actions are consistent with the definitions of the ontology. This is when agents agree to a shared understanding for a set time by putting aside the meanings for a specific time. To our understanding, the ontological commitment is entirely different from the attempts of managing individual ontologies by considering reflections from others. The reason for this is that "ontological commitment" is mainly focused on the agreements for using the shared vocabulary in a coherent and consistent manner, whereas our work investigates ontology management on demand, which is a further step. Our approach is not only considering ontological commitment but also attempting to provide a set of strategies and corresponding mechanisms to cope with the evolution of ontology over time.

7.2 Specific Related Work

Ontologies are becoming an integral part of many industrial and academic applications in the field such as supporting semantics-based search, and Semantic Web applications. As ontology development becomes a more ubiquitous and collaborative process, ontology evolution is becoming an important area of ontology research.

Although ontology evolution is essential in ontology management and even ontology engineering, there are no commonly accepted methodologies or guidelines for ontology evolution so far. Thus there are very few approaches to its investigating.

In [89] the author presents the guiding principles for building consistent and principled ontologies in order to facilitate their creation, their usage and maintenance in widely distributed environments.

The approach to ontology evolution given in [132] is very interesting. In this work an ontology is used to specify the semantics of possible changes of a knowledge base. It presents a six-phase evolution model to check the ontology after changes have been made with the possibility of reversing these changes. Noy and Klein [110] discuss an ontology evolution system by extracting the operations from two versions of one ontology. They developed a framework [64] for managing ontology evolution by extracting the operations from one ontology version and transferring them to another. Similar to [132], an ontology for specifying change operations is presented. A recent project, the KAON (*http://kaon.semanticweb.org/*) [88], is more flexible, by enabling the user to control and customise the manner of resolving change. However, this requires the possible means of resolving change being specified in advance. One possible way is to enable the system to calculate on its own, all ways that satisfy the users' needs.

Ontology evolution can be treated as a part of the ontology versioning mechanism that is analysed in [63]. Klein and Fensel provide an overview of causes and consequences of the changes in ontology. However, the most prominent shortage is the lack of a detailed analysis of the effect of specific changes on the interpretation of data.

Tamma et al. [135] present an extended ontology knowledge model to describe what is known by agents in a MAS. However, the model is not used for supporting ontology evolution.

Other research communities have also influenced the research into ontology evolution which benefits from many years of research of database and knowledge-based system evolution [93].

In contrast to previous work that addresses ontology evolution in one form or another, we go one step further by allowing agents to take part in the evolution

process and reflect ontology changes in the environment. The application of an agent-based approach in ontology refinement is two-fold. On the one hand, it fills the various gaps in understandings of a specific issue by providing an appropriate approach to catch differences whenever they arise. On the other hand, reflections from agents have a great impact on the agent which has identified a discrepancy in understanding on the basis of its ontology. This in turn activates the agent to perform some managing tasks to its ontology repository. With widespread use of ontologies, we believe this approach together with ontology mapping and integration discussed in Chapters 4 and 5, is applicable in the areas such as e-marketplaces, virtual organisations and ontology-based applications where interaction plays a vital role in overcoming the problems that evolution has been facing for a long time.

7.3 Negotiation Model

Negotiation in MAS includes a negotiation set, a protocol, a collection of strategies, and rules [150]. Here, negotiation aims to eliminate conflicts/inconsistency in order to reach agreement about specific negotiation issues. Figure 7.1 is a graphic demonstration. It also shows that agents approach the goal gradually after several rounds of negotiation. Obviously, efficient individual negotiation strategies in the correct direction of achieving the goal are at the heart of this stage. Normally a strategy is private, i.e., each agent has its own strategy profile which is invisible to other agents. In our term negotiation, an *open cry* setting is applied to reach the agreement if conflicts in understanding each other occur. The negotiation strategy specifies what to do next. The results of negotiation are applied to refine ontologies.

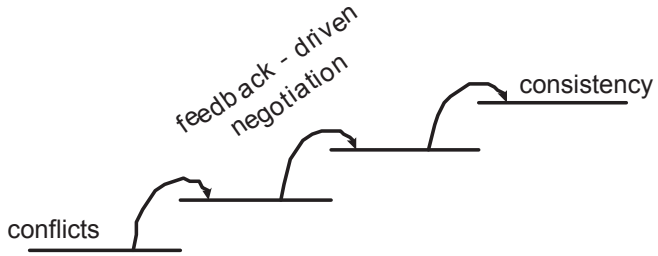

Figure 7.1: Feedback-driven negotiation

The negotiation strategy used is the strategic-negotiation model based on Rubinstein's model of alternative offers [125]. Please refer to [65] for a general view of negotiation of self-interested agents in a MAS. In the strategic-negotiation model, a set of agents is defined as $\mathcal{AG} = \{Ag_1, Ag_2, \ldots\}$. It is assumed that the agents can take actions defined as $\mathcal{A}c = \{\alpha_1, \alpha_2, \ldots\}$, which are available from agents' action repertoire in a time sequence set $\mathcal{T} = \{0, 1, 2, \ldots\}$, that is known to the agents. A sequence set is defined as $\mathcal{S} = \{s_1, s_2, \ldots\}$, corresponding to the time sequence. $\forall t \in \mathcal{T}$ of a negotiation period, if negotiation is still going on without any agreement being reached, the agent, with its turn to make an offer at time t, will suggest a possible solution to other agents which may choose one of the following three answers. Each of them may either `accept` an offer by choosing `Yes`, `reject` an offer by choosing `No`, or `opt out` of a negotiation by choosing `Opt`. If an offer is accepted by all agents, negotiation is terminated and then followed by implementation. Let $fo = (G_{index}, t)$ be the released *offer* that an agent makes at time period t, where G_{index} annotates a subgraph of a particular taxonomy ontology. Generally speaking, in order to reach an agreement in ontology management, simply choosing either `Yes`, `No`, or `Opt` is insufficient. For the purpose of providing more information in each negotiation round, the spectrum of the feedback state is extended to include the following activities:

- state 1: `No`.
  This indicates that the agent knows nothing about the offering question.

- state 2: `Yes`.
  Implicitly, two things may happen. The agent may choose either one from: (1) I know something about it (a certain degree); and (2) I know. If it chooses the second one, it is one hundred percent in agreement with a specific negotiation issue suggested by the agent who is currently making an offer.

- state 3: `Yess` (Yes to some extend).
  This provides the feedback with an approximate percentage to describe the agent's opinion.

- state 4: `Opt` (Opt out).
  If at least one of the agents opts out of the negotiation, then the negotiation

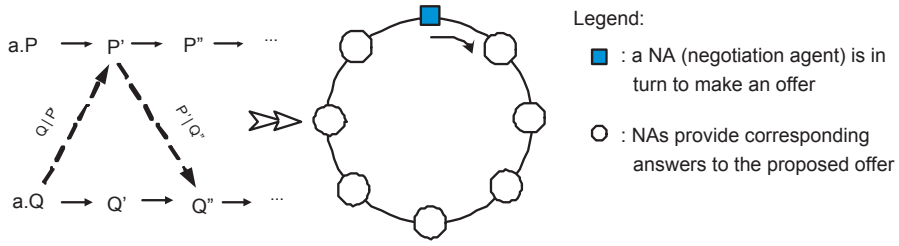

Figure 7.2: Negotiation process - strategy determined by interactions and agent's status

terminates.

By adopting process algebra for the description of interaction between agents, we intend to reflect agents' responses by receiving messages sent by an agent in the previous or current round of negotiation. This will decide which kind of strategy the agent will have when its turn comes. In other words, the strategy indicates to the agent what to offer at $t + 1$. The process is shown in Figure 7.2, where each agent has its own strategy profile decided by its individual global view at a certain period. On the left side of Figure 7.2, the interaction between agents is represented by process algebra as deduction rules. For simplicity, only two agents/processes (in a MAS we assume that the agent conducts the process, so we use both of them interchangeably) are shown. Generally speaking, interactions between agents can be depicted by parallel composition process $P|Q$ with $P, Q \in \mathcal{P}$, $\mathcal{P}$ is a process set. Section 7.4 will provide more details regarding the process description. Any reflection from the previous or current round will affect an agent's next action. The negotiation process is shown on the right side of Figure 7.2.

The process is thus described as a restricted token passing cycle with each agent making offers in turn, but other rational agents provide feedback (positive/negative) as much as possible. According to the above discussion, the potential solution to a corresponding *offer* is defined as: $fs = (\lambda, fo)$, where λ means the current agent's agreement upon a specific concept in the form of a percentage. We are interested in reflections between agents at a certain time point t. We will discuss agent interaction and its effects by means of process algebra in much more detail in the following sections.

7.4 Process Algebra

It is understood that a MAS is a collection of autonomous agents that work together to solve problems that are beyond the capabilities of individual agents. That agents in a MAS may not share common goals requires coordination of their activities and cooperation with others at run-time rather than hard-coded strategies at design-time. Bearing this in mind, we will investigate interaction protocols using process algebra in the following sections.

7.4.1 Process Algebra for Information Exchange

An agent can perceive any change of environment [150], that is, it is not deaf-and-dumb during the operation. For formally representing the interaction process, we consider a mathematical formalism based on π-calculus [96], a type of process algebra with the notation of names and processes specified for distributed and communication computations, that describes the communication behaviours of a distributed system. The interaction is likely to be described by means of information exchange with certain restrictions between interested agents. Two main types of actions are *sending* and *receiving* for a given *communication point*:

$\overline{a}(x).P$: meaning the action of sending the value along the point

$\underline{a}(x).P$: meaning the action of receiving the value along the point

Let $a, b, \ldots$, be set interaction points that are considered as the channel between the processes, while $x, y, z, \ldots$, is set of name variables and information values that can be instantiated as a specific object for transmission in process algebra. P, Q, A, B, ... are elements of process set $\mathcal{P}$. Despite the fact that there is more than one agent involved in a process, for the purposes of notation, we assume process P represents agents involved. Agents are thus ranged over by P, Q, ... The syntax of basic process algebra is defined as:

$$\begin{aligned} P ::= \alpha.P & \qquad prefix \\ P + Q & \qquad Sum \\ P|Q & \qquad Parallel \\ \mathbf{0} & \qquad Nil \end{aligned}$$

$$(\nu x)P \qquad \textit{Restricting}$$

where α is the prefix of sending and receiving ranging over $\overline{a}(x)$ and $\underline{a}(x)$. We have five constructors which are detailed in the following.

- Prefix $\alpha.P$ defines a process with the information flow prefix and continuation.
- *Sum* $P + Q$ represents an agent that can enact either P or Q.
- *Parallel composition* $P|Q$ represents two independent components which can act independently, and may have dependency by P acting to send information flow, and another Q performing complementary action receiving information flow along the same point.
- The empty agent $\mathbf{0}$ which cannot have any interaction with the environment.
- $(\nu x)P$ behaves as P, but restricting its performance on information flow, name y cannot be used for communication.

The purpose of describing reflections of individual agents in a negotiation setting is achieved by utilising interesting properties of the process algebra, specifically, deduction rules, in governing negotiation processes at run-time. Just as in traditional process algebra [4], the deduction operator "$\rightarrow$" over agents (processes) like $P \rightarrow Q$ is used to define information exchange among the agents for a computation step. For example, the following representation

$$(\ldots + \overline{a}(\overrightarrow{x})).P|(\ldots + \underline{a}(\overrightarrow{y})).Q \rightarrow P|Q\{\overrightarrow{x}/\overrightarrow{y}\}$$

means that process $(\ldots + \overline{a}(\overrightarrow{x})).P|(\ldots + \underline{a}(\overrightarrow{y})).Q$ is transformed into $P|Q\{\overrightarrow{x}/\overrightarrow{y}\}$ in which $\overrightarrow{y}$ is substituted for $\overrightarrow{x}$ after the communication occurs ($\overrightarrow{x}$ and $\overrightarrow{y}$ are of equal length to the name vector). The reduction rules can be derived from process commutative and associative properties based on the above deduction operator definition (e.g. deduction relation). In this respect, information exchanges between agents can be modeled by process algebra to highlight an agent's prompt reflection and the effects caused accordingly, and how it affects an agent's next action. By saying so, we are attempting to describe agents' interactions (process interaction) by using process algebra.

7.4.2 Agent Process Interaction

Describing and fulfilling interactions among agents involved in a dynamic environment is inspired by using process algebra. Process algebra gets its richness from mathematics. Here we are interested in the use of process algebra to depict interactions among agents in a particular negotiation setting.

Definition 1: An interaction process is an enactment process, denoted as P_I, $P_I \in \mathcal{P}_\mathcal{I}$, $\mathcal{P}_\mathcal{I}$ is a interaction process set. It defines at least two independent processes that interact via a channel for a name to be sent and received on the channel.

Using the π-calculus notation, we may express the interaction process $P_I = \overline{a}(x).P|\underline{a}(y).Q$, where $\{\overline{a}, \underline{a}\}$ or $\{\overline{b}, \underline{b}\}$ is a pair of dual ports that are considered as the channel between the processes, x and y are the name variables that can be instantiated as a specified object for transmission, $A = \overline{a}(x).P$ and B=$\underline{a}(y).Q$ are two processes. The interaction process is a specified process where the properties of the process algebra are retained.

More complicated interaction processes can be constructed via the channels and processes involving multiple processes. The following representation is an example ($m \in \mathsf{N}$).

$$P_I' = \overline{a}(x_1).P|\underline{a}(x_2).Q|\ldots|\underline{a}(x_m).W$$

In this case, there is one sender but with multiple recipients. In contrast, there is a case with one recipient but multiple senders. It is shown below ($n \in \mathsf{N}$).

$$P_I'' = \underline{a}(x_1, x_2, \ldots, x_n).P|\overline{a}(y_1).Q|\overline{a}(y_2).R|\ldots|\overline{a}(y_n).W$$

. To this end, the interaction processes, such as the one below

$$P_I''' = \overline{a}(x_1).P|\underline{a}(y_1).\overline{b}(x_2).R|\underline{b}(y_2).S$$

can be split into two interaction processes with $Q_I' = \overline{a}(x_1).P|\underline{a}(y_1).Q$ and $Q_I'' = \overline{b}(x_2).R|\underline{b}(y_2).S$.

We do not intend to capture information such as who is the sender and who is the recipient. Instead, the interaction between two process is at a high level of abstraction that indicates what the content of the interaction is. It enables more flexibility in configuring implementation at run-time.

Now we consider the transformation of two interaction processes. In process algebra, the reduction rule determines the transformation between two processes. However, the reduction rule cannot be deployed directly for the interaction processes because of the inherent logic of the interaction processes. The transformation of the interaction processes does not follow the logic in processes. For instance, $\overline{a}(x).P|\underline{a}(y).Q \rightarrow P|Q\{x/y\}$ obeys the logic in processes, but the above two interaction processes $\overline{a}(x_1).P|\underline{a}(y_1).Q$ and $\overline{b}(x_2).R|\underline{b}(y_2).S$ have no logic between them because they can be transformed into each other at an unpredicted time. The logical order determines the transformation of the processes. We define "stepping" to represent the logical order of the transformation of interaction processes.

Definition 2: A stepping of interaction processes, denoted as $\searrow$, is responsible for transforming interaction process P_I into Q_I in the form of $P_I \searrow Q_I$, where P_I, $Q_I \in \mathcal{P}_{\mathcal{I}}$.

The stepping does not explicitly state the relationship of the two interaction processes, rather the logical order of the transformation. The information is hidden for the purpose of abstraction. For example, with $P_I = \overline{a}(x_1).P|\underline{a}(y_1).Q$ and $Q_I = \underline{b}(x_2).R|\overline{b}(y_2).S$, after taking a computation step (assuming that they have the same channel), the former one comes to the status: $\overline{a}(x_1).P|\underline{a}(y_1).Q \rightarrow P|Q\{x_1/y_1\}$. According to the stepping definition, the next interaction process is Q_I after P_I. It implies that $P \Rightarrow \underline{a}(x_2).R$ and $Q\{x_1/y_1\} \Rightarrow \overline{a}(y_2).S$. Eventually $Q\{x_1/y_1\}$ will become $\overline{a}(y_2).S$ and P transforms into $\underline{a}(x_2).R$. In fact, the transformation occurs behind the interfaces. It depends on an individual participant's process implementation.

Next, we introduce an interaction process series. this is the abstracted notation for agent negotiation description.

Definition 3: An interaction process series (IPS for short), takes the form $IPS : P_I \searrow Q_I \ldots \searrow W_I$, where P_I, Q_I and $W_I \in \mathcal{P}_{\mathcal{I}}$.

Note that a series of process interactions starts from an interaction process originating a process task, and ends with an interaction process whenever the task is completed.

However, the algebra based interaction processes cannot be directly applied to describe negotiation setting without involving process commitment. Next, we

discuss some basic concepts of process commitment. We assume that the potential agents, apart from their own processes, have an interaction process ready for interaction for a certain circumstance.

Definition 4: A process commitment is a process relation $\succ$ between process P and commitment $\omega.A$, such that P is committed to $\omega.A$, denoted as $P \succ \omega.A$, where ω is the action including the channel and the process that process P will perform and A is the continuation.

Let us recall the interaction process discussed in the above, the interaction process is defined as $\overline{a}(x).P|\underline{a}(y).Q$. If $P_1 \succ \overline{a}(x).P$ and $P_2 \succ \overline{a}(y).Q$, we get $P_1|P_2 \succ \underline{a}(x).P|\overline{a}(y).Q$. Obviously the commitment is made on the processes as well as the action on ports and objects by both parties to the interaction. It is a successful interaction only if both parties agree on the contents of the communication, otherwise either P_1 or P_2 might have additional processes within the continuation P or Q to modify the contents of name x or y. Moreover, in a negotiation setting, it is required that the participating agents adhere to a set of commitments $\mathcal{C}$ for process interaction.

Definition 5: Agent commitment is the relation $\succ$ between the agent's interfaces P_i and a set of commitments $\{\omega_i.A_i\} \in \mathcal{C}$ $(i \in \mathsf{N})$, such that the agent will perform commitments to enable interaction processes. $\{\omega_i.A_i\} \in \mathcal{C}$ $(i \in \mathsf{N})$ is said to have a logical dependent commitment if $\{\omega_i.A_i\}$ $(\{\omega_i.A_i\} \in \mathcal{C})$ can be transformed to another in a logical order for given finite computation steps, namely $\{\omega_i.A_i\} \rightarrow \ldots \rightarrow \{\omega_j.A_j\}$ $(i \neq j)$, otherwise, they are independent.

7.4.3 Interaction Process Evolution

The interaction processes affect not only individual processes belonging to different participating agents but internal interaction processes. In addition to the agent process interaction and the process commitment discussed in Section 7.4.2, we investigate how the interaction processes run logically from one interaction process to another.

Suppose that we have a set of required interaction processes to complete the process of an initial offer (by one of agents), denoted as $P_{Ij} \in \mathcal{P}_I$ $(j \in \mathsf{N})$. Prior to

completing the interactions, the interaction process series is given by

$$IPS : P_{I1} \searrow P_{I2} \searrow \ldots \searrow P_{Ij}$$

By interaction process commitments of agents, each interaction process in the series is defined as: $P_{Ij} = \omega.A_j|\rho.B_j$, where ω and ρ are the communication actions including the channels and processes, and A_j and B_j are two processes performed by individual agents. The stepping of the interaction processes gives an interaction process series in the form of

$$IPS : \omega.A_1|\rho.B_1 \searrow \omega.A_2|\rho.B_2 \searrow \ldots \searrow \omega.A_j|\rho.B_j$$

Because $P_{Ij} \in \mathcal{P}_I$ and $P_{Ij} = \omega.A_j|\rho.B_j$, let $\omega.A_j|\rho.B_j \rightarrow A_j|B_j\{\omega/\rho\}$, if $A_j \Rightarrow \omega.A_{j+1}$ and $B_j\{\omega/\rho\} \Rightarrow \rho.B_{j+1}$, then a new interaction process $\omega.A_{j+1}|\rho.B_{j+1}$ is derived.

Proposition: For sequential interaction process $\mathcal{P}_I$, there must exist two sets of independent processes such that the involved process has the same order as the interaction process. Furthermore, the evolving operators define the additional processes between interface processes that transform from one interface process to another.

Proof: Suppose that interaction process $P_{I1}, P_{I2}, \ldots, P_{In} \in \mathcal{P}_I$, $ISP : P_{I1} \searrow \ldots P_{In}$, $(n \in \mathsf{N})$. By definition of the interaction process, each interaction process $P_{Ij} = \omega.A_j|\rho.B_j$ $(j \in \mathsf{N})$ directs the process stepping $\omega.A_1|\rho.B_1 \searrow \omega.A_2|\rho.B_2 \searrow \ldots \searrow \omega.A_j|\rho.B_j$, where $\omega.A_j$ and $\rho.B_j$ are the communication processes independently performed by different agents.

Now $\omega.A_j|\rho.B_j \rightarrow A_j|B_j\{\omega/\rho\}$, if $\omega.A_j|\rho.B_j \searrow \omega.A_{j+1}|\rho.B_{j+1}$, the independent property of the communication processes gives $\omega.A_j \rightarrow \omega.A_{j+1}$ and $B_j\{\omega/\rho\} \rightarrow \rho.B_{j+1}$. Then it refines $B_j\{\omega/\rho\}$ such that $B_j\{\omega/\rho\} \rightarrow \ldots \rightarrow \rho.B_{j+1}$, an evolving consequence leads to $\rho.B_{j+1}$. Then $P_{Ij} \rightarrow P_{Ij+1} = \omega.A_j|\rho.B_j \rightarrow A_j|B_j\{\omega/\rho\} \Rightarrow \omega.A_j \rightarrow \omega.A_{j+1}$ and $\rho.B_j \rightarrow \rho.B_{j+1}$.

Based on the analysis, it is concluded that for a given logically ordered interaction process series among the participating agents, there are internal processes of each agent such that if the participating agent commits to the interaction process series, then in addition to incorporating the required processes, the internal

process can be constructed in such a way that its evolution could follow the stepping of interaction processes for communication. It it believed that as long as the agent commits to the interaction process, the internal process can be constructed or re-constructed to match the interaction process series.

In addition to the interaction model based on process algebra, how to solve conflicts or misunderstandings in MAS is another issue that needs to be addressed. Before we start to discuss this work, it is worth defining utility functions to tell an agent how "good" the current result/state is.

7.5 Agent Utility Function

In terms of the result/state, as we assume that an agent is proactive, it should obviously act in the direction of maximising its welfare at each stage of a process, by considering environment changes overtly. In this approach, we adopt a utility function defined in [150] as: $u : \Omega \rightarrow R$, where $\Omega = \{\upsilon_1, \upsilon_2, \ldots\}$ of states that agents have preferences over, R is the set of real numbers. For two elements υ and υ', wher $\upsilon, \upsilon' \in \Omega$, if $u_i(\upsilon) > u_i(\upsilon')$, then we say υ is preferred by agent i at least as much as υ'. Every agent is willing to adopt a good strategy to maximise its expected utility but not one has the motivation to deviate and use another strategy because strategies used by agents are in *Nash equilibrium* [103]. Clearly, the optimal agent action Ag_{opt} which leads to the best performance under certain circumstances is defined as: $Ag_{opt} = arg\ max_{Ag \in \mathcal{AG}} \sum_{\upsilon \in \Omega} u(\upsilon)$ ([150], p. 39), where $\mathcal{AG}$ is the set of all agents.

7.6 Scope of Ontology Refinement - Incorporating Ontology Refinement in Ontology Management

Ontologies are subject to evolution over time. There must be a dedicated process to look after ontology evolution to save us from overlooking it. In our framework

(Section 3.3), a `refinement agent` (*RA*) is designed to be responsible for tackling evolution during run-time. Incorporating ontology refinement into ontology mapping and integration to provide a holistic view of ontology management makes our approach go beyond existing work in this field. We attempt to catch changes of involved agents and reflect their changes in ontology mapping and integration and the next actions of *OA*s, which act on behalf of their ontologies in the design. Properly bound ontology refinement with ontology mapping and integration makes our approach unique in that it can directly obtain changes and seamlessly use them to take succeeding actions (Figure 7.3). From previous discussions (Chapter 5), it is clear that ontologies, regardless of existing ones or newly generated ones, can be used/reused in the design. It is envisaged that our design is potentially adequate to cope with ontology management requirements in a dynamic and heterogeneous environment.

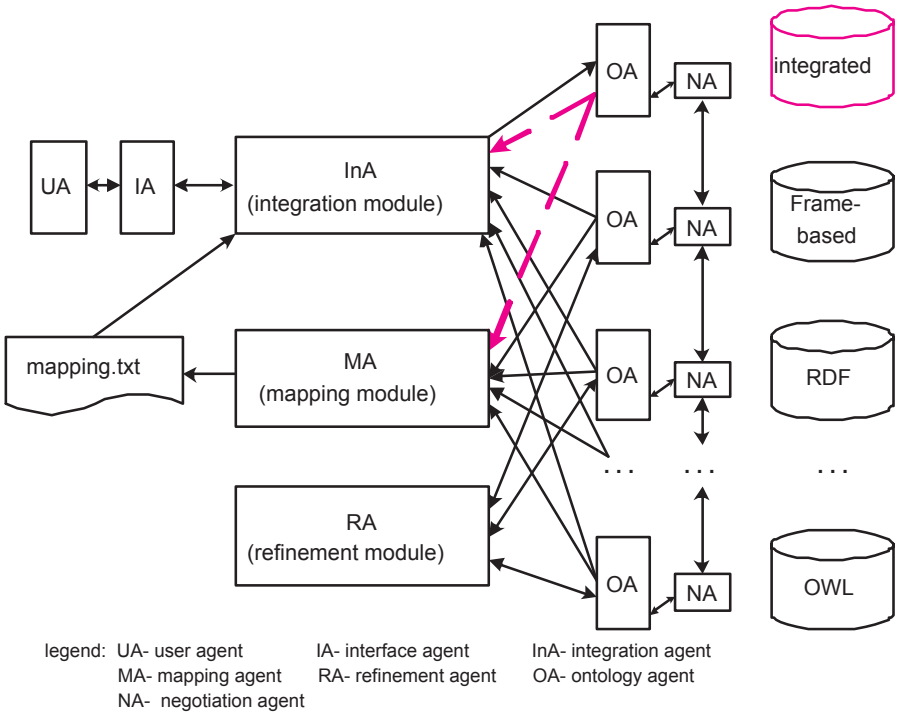

Figure 7.3: Ontology refinement

It is worth noting that we do not intend to expand our work to include ontology change recursively. In other words, we focus on the stage of how an agent refines its own ontology in adaptation to the environment (e.g. ontology) change rather

than tracing the impact of change propagation (to ontology-based application).

7.7 Refinement Process

In this thesis, the refinement module starts when the *RA* is enacted. In contrast to the mapping process (Section 4.4) and the integration process (Section 5.4), which take place whenever they are requested, the refinement process runs in the background to provide *OA*s with ontology modification (inserting/deleting/updating) in a timely fashion. Briefly speaking, the *RA* monitors the *NA*s negotiation processes via individual *OA*s. Agreements reached by *NA*s during negotiation are fed back to corresponding *OA*s, which take appropriate actions to modify their ontologies. We assume that *OA*s are able to present integral and consistent ontologies at a certain point in time whenever they are asked (e.g. by *MA* and *InA*). The refinement process is outlined as follows:

(1) *NA*s get to start when the *RA* starts up;

(2) *NA*s start negotiation according to the strategic-negotiation model (Section 7.3);

(3) *OA*s modify their acting ontologies based on negotiation results respectively.

Figure 7.4 displays interactions among agents involved in the refinement process. The refinement process influences existing ontologies and *OA*s as well. After *NA*s reach agreements or when the negotiation process is terminated, *OA*s then modify corresponding ontologies. As ontologies are subject to change over time, the *CA* is kept running to guarantee the above process presents consistent ontologies for further ontology operations.

7.8 Refinement Mechanisms

The refinement module is responsible for ontology refinement. The *RA* is in charge of refining ontologies whenever possible by means of the negotiation process. The negotiation process includes an agent presenting an initial offer, together with other

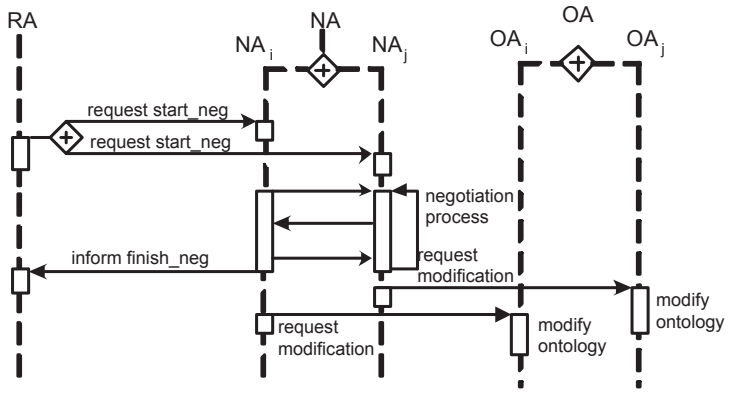

Figure 7.4: Interactions from *RA*'s view

agents providing potential solutions in terms of the initial offer. After that, the same agent checks potential solutions from others to make a decision on its next action (e.g. exit the process or keep track of refinement for the next time). Three relevant functions are presented to describe the negotiation module. They are described in a sequence within each table: (1) defining data structures and variables used; (2) defining some functions; and (3) describing corresponding process algorithms in pseudocode. The pseudocode of an agent's behaviour which is not in turn to give an initial offer, is presented first. The pseudocode of an agent's behaviour which is in turn to give an initial offer is shown next. The pesudocode to describe the negotiation process is then presented followed by the ontology modification pesudocode. Based on the above processes, finally the pesudocode of the refinement process is given.

Pseudocode of agent behaviour without initial offer

/*assume four potential solutions from an agent which is not in turn to give an initial offer: "Yes", "No", "Opt" and "Yess" (Section 7.3);
m: the number of available ontologies, also the number of *OA*s;
j: a randomly generated integer number between 1 and m standing for one of the *OA*s;
solution: a variable for one of the above four solutions;
λ: a variable to estimate percentage (accept to some extent);
`evaluate`: a function to evaluate the initial offer;

`send1`: a function to send back λ percentage to indicate percentages upon agreement;
`send2`: a function to send back *solution*.
*/
Function `solution_offer` {
1. **for** $(i = 1; i < m; i++)$ {
2. **if** $(i! = j)$ {
3. *solution*=`evaluate` initial offer;
4. **switch** (*solution*) {
5. **case** "Yess": `send1`; **break**;
6. **default**: `send2`;
7. } // end switch
8. } // end if
9. } // end for
10. }// end function

Pseudocode of agent behaviour with initial offer

/*assume four possible solutions: the same as definitions in the *Pseudocode of agent behaviour without initial offer*;
m: the number of available ontologies, also the number of *OA*s;
solution: a variable for one of the above four solutions;
*threshold*1: a percentage threshold to filter unexpected items;
*threshold*2: a threshold that the ontology refinement may need;
change_list: a list keeping track of changes;
$k_{j,y}, k_{j,n}, k_{j,o}$: the number of occurrences of potential solutions "Yes", "No" and "Opt", respectively;
λ: a variable of estimated percentage (accepted to some extent);
decision: decisions that an agent makes according to its knowledge and current change of the environment;
`decide_next`: a function to decide what the next action (for a particular agent) is;
`modify`: a function to modify the *change_list* with the potential *decision* made in `decide_next`.

*/
Function `check_initial_offer` {
1. **do** {
2. **switch** (*solution*) {
3. **case** "Yes": $k_{j,y}++$; **break**;
4. **case** "No": $k_{j,n}++$; **break**;
5. **case** "opt": $k_{j,o}++$; **break**;
6. **default**: **if** ($\lambda >= threshold1$) $k_{j,y}++$;
7. **else** $k_{j,n}++$;
8. }// end switch
9. **if** ($k_{j,o} == m-1$) **return** $k_{j,o}$;
10. **if** ($k_{j,n} >= threshold2$) {
11. `decide_next`;
12. `modify`;
13. }// end if
14. **if** ($k_{j,y} == m-1$) **return** $k_{j,y}$;
15. } **while** (existing potential solutions from agents involved in the negotiation setting); // end do
16. }// end function

Pseudocode of negotiation process

/* m: the number of available ontologies, also the number of OAs;
j: a randomly generated integer number between 1 and m standing for one of OAs;
`generate_nextInt`: to generate a random integer number;
`initial_offer`: an agent (decided by a randomly generated integer number) to issue an initial offer;
`solution_offer`: function `solution_offer`;
`check_initial_offer`: function `check_initial_offer`.
*/
Function `negotiation` {
1. **do** {
2. j =`generate_nextInt(m)+1`;

```
3.      initial_offer(j);
4.      solution_offer(j);
5.      check_initial_offer(j);
6. } while (negotiation time is not out); // end do
7. } // end function
```

Pseudocode of ontology modification

/* *change_list*: a list keeping track of changes (as defined in the *Pseudocode of agent behaviour with initial offer*);
top: an element from a list;
`insert`: a function of insert operation on a specified ontology hierarchical structure;
`delete`: a function of delete operation on a specified ontology hierarchical structure;
`update`: a function of update operation on a specified ontology hierarchical structure;
`get_head`: get the head element of a list;
*/

```
Function ontology-modification   {
1. do   {
2.    top=get_head(change_list);
3.    switch   {
4.          case "insert": insert; break;
5.          case "delete": delete; break;
6.          default: update;
7.    } // end switch
8. } while (top!=Null); // end do
9. } // end function
```

Pseudocode of refinement algorithm

/* *change_list*: a list keeping track of changes (as defined in the *Pseudocode of agent behaviour with initial offer*);
`initialise`: initialise the process;
`negotiation`: function `negotiation`;
`ontology_modification`: *OA*s modify their acting ontologies according to negotiation results, referring to *change_list* for any changes.
*/
Function `refinement` {
1. `initialise`;
2. `negotiation`;
3. `ontology_modification`;
4. } // end function

7.9 Example - Reaching Agreement between Agents

We consider an example of *beer order* where the domain knowledge of the brewage industry is involved. We have discussed the basic business conducting phases in [72]. Generally speaking, four phases, namely `order`, `fulfillment`, `payment`, and `delivery` are involved. In the real world, every stage might consist of many sub processes that are more complex and beyond the scope of this chapter. In order to highlight our concentration, we focus on these four stages by observing the accomplishment of a task as a process. In saying this, we mean that each agent, which conducts a process or is embedded in a process, performs its internal and coherent processes with respect to business rules. Meanwhile, a MAS also presents a business process from a holistic view. In order to exemplify the case, only different entities are shown in Figure 7.5, with inter-organisational relations simplified as dotted circles (actually it is a VO formed upon requested).

Considering these organisations work together to achieve a defined objective, it is not unusual that they have different ontologies where beer type is concerned. We assume that these four organisations (e.g. MAS) are supported respectively by the four beer ontologies presented in Section 4.6. In addition to ontology mapping

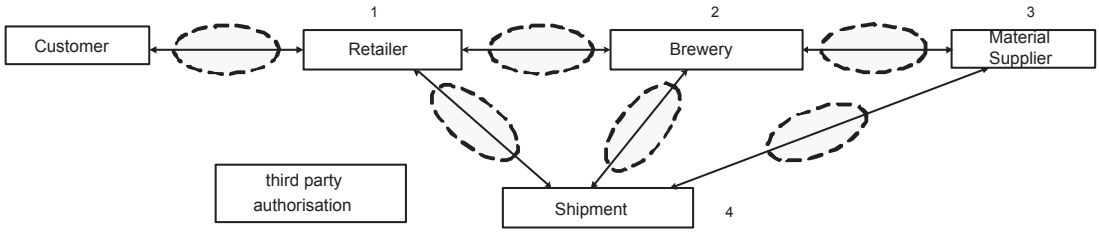

Figure 7.5: Beer order in brewage industry domain

and integration, discussed in Chapters 4 and 5, we focus on an ontology evolution to illustrate the key point of agent negotiation in reflecting ontology evolution.

The negotiation setting of *order fulfillment* where four process agents are involved, consists of *Retailer* (R for short), *Brewery* (B for short), *Supplier* (Su for short) and *Shipment* (Sh for short). We assume that a particular business scenario would provide prerequisites for how the process is to be executed. In this example, let us suppose that the business processes that agents must perform are as follows (modified from the report at *http://www.wfmc.org/standards/docs/IneropChallPublic.PDF*):

$$\begin{aligned}
Retailer &= R(place_order, invoice_customer, pay_invoice, approve_payment,\\
&\qquad print_cheque, \ldots)\\
Brewage &= B(collect_order, order_parts, schedule_production, check_inventory, \ldots)\\
Supplier &= Su(collect_order, prepare_parts, deliver_parts,\\
&\qquad invoice_manufacturer, \ldots)\\
Shipment &= Sh(collect_order, book_delivery, schedule_van,\\
&\qquad confirm_delivery, \ldots)
\end{aligned}$$

Accordingly, the negotiation setting is defined as follows (activities involved are annotated by acronym, for example, *co* is used to notate *collect_order* in the manufacturing process.).

$$setting_{Neg} = R(po, ic, pi, \ldots)|B(co, op, sp, \ldots)|Su(co, pp, dp, \ldots)|Sh(co, bd, sv, \ldots)$$

The processes for these four agents can be defined as (with the prefix such as $\overline{po}$ and $\underline{ap}$ the same as defined in Section 7.4.1, and processes like R_1 denoting internal

processes):

$$R(po, ic, pi, ap, pc, \ldots) = \overline{po}(x_1).R_1 + \overline{ic}(x_2).R_2 + \overline{pi}(x_3).R_3 + \underline{ap}(x_4).R_4 + \overline{pc}(x_5)R_5 + \ldots$$
$$B(co, op, ci, \ldots) = \underline{co}(y_1).B_1 + \overline{op}(y_2).B_2 + \overline{sp}(Y_3)B_3 + \underline{ci}(y_4)B_4 + \ldots$$
$$Su(co, pp, dp, im, \ldots) = \underline{co}(z_1).Su_1 + \underline{pp}(z_2)Su_2 + \overline{dp}(z_3).Su_3 + \overline{im}(z_4).Su_4 + \ldots$$
$$Sh(co, bd, sv, cd) = \underline{co}(d_1).Sh_1 + \overline{bd}(d_2)Sh_2 + \underline{sv}(d_3)Sh_3 + \overline{cd}(d_4)Sh_4 + \ldots$$

where R_i, B_j, Su_k and Sh_l are the processes with the prefix actions specified, x_j, y_j, z_j and d_j are name variables, and i, j, k and $l \in \mathsf{N}$.

In addition to the individual processes above, we take a close look at the interaction processes for the purpose of the negotiation. In the following we demonstrate the *order* related interaction processes of these four participating agents (with the same channel between two interaction agents). They are defined as

$$place\_order(R \mapsto B) = \overline{a_1}(x_1).R_1 | \underline{a_1}(y_1).B_1$$
$$order\_part(B \mapsto Su) = \overline{a_2}(y_2).B_2 | \underline{a_2}(z_1).Su_1$$
$$schedule\_production(B \mapsto Sh) = \overline{a_3}(y_3).B_3 | \underline{a_3}(d_1).Sh_1$$

If we denote the above interaction processes (agents' interfaces) as P_i (one agent) and Q_i (the agent which acts accordingly with the former agent) ($i \in \mathsf{N}$), we can see the commitments in *order* related processes (Figure 7.5) as follows:

$$\begin{array}{lll} P_1 \succ \overline{a_1}(x_1).R_1 & place\_order & Q_1 \succ \underline{a_1}(y_1).B_1 \\ \quad\quad\downarrow & \quad\quad\searrow & \quad\quad\downarrow \\ P_2 \succ \overline{a_2}(y_2).B_2 & order\_part & Q_2 \succ \underline{a_2}(z_1).Su_1 \\ \quad\quad\downarrow & \quad\quad\searrow & \quad\quad\downarrow \\ P_3 \succ \overline{a_3}(y_3).B_3 & schedule\_production & Q_3 \succ \underline{a_3}(u_1).Sh_1 \end{array}$$

With these in mind, now we discuss negotiation for the purpose of reaching an agreement in terms of available ontologies. Suppose that the *Beer Retailer* has been randomly chosen to provide an initial offer, and another three parties are ready to provide potential solutions in terms of the initial offer. With four beer type ontologies being available, the other three agents' potential solutions are identified as follows.

$fs_{B,t_1} = (\lambda_B, G_{index_j}, t_1)$

$fs_{Su,t_2} = (\lambda_{Su}, G_{index_k}, t_2)$

$fs_{Sh,t_3} = (\lambda_{Sh}, G_{index_l}, t_3)$
where j, k and $l \in \mathsf{N}$, λ_B, λ_{Su} and λ_{Sh} are the percentage of the agreement upon particular concepts, respectively. G_{index_j}, G_{index_k} and G_{index_l} denote the subgraphs of the other three ontologies in the taxonomy respectively; t_1, t_2, and t_3 are certain time periods.

We suppose that if the topic of negotiation is about a particular concept, feedback about it would be available after the negotiation. The *Beer Retailer* thus will refine its ontology if possible. The example shows two interesting aspects of agent-based ontology refinement. One aspect is that the ontology refinement can be modelled from the MAS point of view when distributed and dynamic issues are considered. Another one is that communications between agents should be richer and more powerful than KIF (knowledge interchange format) and ACL (agent communication language) performatives if we look back to the interaction between agents which is modelled by process algebra.

7.10 Discussion

In the above discussion (Section 7.9), we mentioned that the agent in turn keeps track of agents' feedback. That is to say, agents need to maintain linkages with other agents in case other agents need to access the recorded information of the previous round of negotiation. It seems that it is a heavy burden for agents to have this kind of one-to-one linkage. Fortunately, the number of agents which are involved in a business scenario and at the same time with a direct linkage with one another is less than expected (it is not a fully connected graph). In this sense, the computation complexity can be reduced.

On the other hand, agents engaged in negotiation are keen to choose "good" strategies to maximise their utilities respectively because strategies used by agents are in *Nash equilibrium*. However, scoring criteria vary from circumstance to circumstance. The negotiation algorithm is neither just simply Tit-For-Tat (against its opponents on all rounds), nor just reciprocal (reciprocating whatever its opponent did on the previous round). In order to prevent agents from opting (e.g. `opt out`) or responding nothing (e.g. `no`), suggesting useful offers (from the point

view of the agent which makes an initial offer) in our case is desirable. Some kind of heuristic algorithm (scoring system) may need to be developed to direct the negotiation to the defined objective for the reason that in negotiation, there is no absolute notion of for and against.

7.11 Summary

In this chapter, our general focus has been on developing flexible mechanisms with each agent having its negotiation strategy to deal with inconsistency or conflicts in understanding meanings of certain concepts, and eventually to achieve the goal of conformance to changes in an environment. We argue that agent interactions at run-time are a suitable way to substantially support environment changes. This mechanism also makes it possible for agents to adapt themselves to changes through communications. As agent communications may take place concurrently, it is better to represent them with process algebra. We have proposed that interaction in the process algebra model symbolises information exchange between agents and the effects it may cause. Each agent's strategy profile is determined from the feedback of the environment and its current local global view. The novelty of our work lies in that we have highlighted the motivation for why ontology evolution is needed and how it adapts in accordance with the ontology evolving cycle. In addition, we have developed strategies to reflect the dynamic changes in this cycle. Finally, the interaction in a MAS has been discussed thoroughly on the basis of process algebra.

Chapter 8

Conclusions and Future Work

Agent technology is a promising point of departure for engineering complex software systems. With industrial-strength, multi-agent system (MAS) software and corresponding systematic methodology becoming mature, we can anticipate that more and more application areas will benefit from applying agent technique in one form or another.

Firstly, in this chapter, a summary of the thesis is presented in Section 8.1, followed by key contributions in Section 8.2. After that, directions for future research are addressed in Section 8.3.

8.1 Summary of Thesis

Based on the analyses conducted in this thesis, we come to the conclusion that the proposed framework has the following characteristics that differentiate it from others:

1. effectively interacting among agents (between `ontology agents` and component agents, and `ontology agents` themselves) to realise ontology dynamics;
2. freely adding/deleting ontology component agents to/from the system;
3. highly isolating ontology related tasks such as traversing the ontology structure, gathering requested information of a particular ontology from external agents' views by using `ontology agents`;

4. easily accessing the `ontology agent`s from other agents in the MAS.

The proposed framework attempts to provide a flexible and effective modelling approach to tackle ontology mapping, integration and evolution over a variety of ontologies. Tasks of ontology management discussed in the thesis are based on interactions between participating `ontology agents` (*OA*s) which provide as much information as possible to suggest better actions under certain circumstances. Ontology mapping is one major task which aims to provide dynamic mapping by enacting the mapping process. This is essential in achieving ontology interoperability. Ontology integration was discussed after ontology mapping. Its main task is to provide a dynamic view (e.g. an integrated ontology) of existing ontologies at an abstract level on demand. For example, a virtual organisation (VO) coalition is frequently required in that an individual may be unable to meet a certain requirement because of limited resources and capabilities. Hence, each participating oraganisation will choose to contribute some complementary information to the business process which consequently enhances overall capabilities to meet the designed objective significantly. Meanwhile, each organisation will benefit from engaging in such a coalition by maximising its potential welfare. It is argued that an integrated ontology is more suitable than any other individual existing ones in this regard. An integration process is responsible for this in our discussion. As a particular `ontology agent` acts on behalf of each ontology to handle ontology related tasks, integrated ontologies can be easily reused later on (to be imported as an available ontology to engage in corresponding tasks in the prototype).

An ontology evolves over time. In the framework, ontology refinement is designed to cope with ontology evolution in the presence of `negotiation agents`. Alternating offering in the proposed negotiation model may reflect any change in an ontology, thus helping the refinement process which deals with ontology evolution to catch instant change whenever required.

8.2 Principal Contributions of This Thesis

This thesis makes two major contributions. The first contribution is a general-purpose framework that provides a foundation and support to heterogeneous and

distributed ontology management. Previous works have turned out to be insufficient to cope with ontology management in a dynamic environment. Here, an agent-based framework is introduced to provide the following key points:

- The framework offers the ability to exchange information via communication channels.
- Each `functionary agent` can easily access other services available in the system.
- The presence of the `ontology agents` in the framework allows system adaptivity possible by:
  - providing `ontology agents` which are responsible for ontology relevant tasks. By doing so, other agents only need to access each `ontology agent` to obtain required information;
  - providing the ability of generating an `ontology agent` to act on behalf of a newly integrated ontology which makes it possible to reuse the ontology in the system in practice.
- The framework is highly modular and easily extensible without expending too much effort. For example, when a new ontology representation language is available, we only need to add a new `ontology agent` to the system. The same is true for agents which provide services to facilitate ontology management as a whole.

The second major contribution of the thesis is a solution to the problem of how to dynamically perform ontology management. The key innovations that we have made in developing this solution are as follows:

- We have proposed mapping mechanisms (Chapter 4) to cope with ontology mapping based on knowledge of the `mapping agent`. Two main semantic relations are involved. They are semantic equivalent and inclusive. The similarity thesaurus dictionary plays an important role in deciding semantic equivalence. The effectiveness of the mapping process can be greatly improved by increasing the number items in the synonym dictionary (through

adding new synonyms) over time. Ontology interoperability can be achieved after mappings between pairs of ontologies are created. The mechanisms are expected to be easily customised to any particular domain.

- We have brought the necessity of exploiting a global view of existing ontologies at an abstract level to the forefront of ontology applications. We have then presented ontology integration mechanisms (Chapter 5) by applying mapping results from the mapping process. The reuse idea yields an outcome in which a new `ontology agent` is created on the fly to act on behalf of the corresponding ontology. Ontology reuse is made possible in practice by `ontology agents` being dedicated to taking care of ontology related work.

- We have developed a prototype (Chapter 6) to demonstrate the feasibility of the framework. Verifications such as query and consistency checking are conducted with the prototype.

- We have investigated a solution to the problem of handling ontology evolution over time. Key ideas underlying our approach rely on agent interactions and the negotiation strategy. By each ontology providing a potential solution alternatively, corresponding to the proposed offer, a broad variety of ontology changes can be captured and reflected through the interactions. We have exploited the refinement process (Chapter 7) to cope with changing ontologies. In order to facilitate negotiation execution, process algebra is applied to describe the information exchange by means of the interaction process. The result is significant because process algebra lays a solid theoretical foundation regardless of what kind of semantic relations there really are between pairs of ontologies.

8.3 Future Work

We have made significant inroads into understanding and developing solutions for ontology management in a dynamic and heterogeneous environment. However, substantial work remains to be done towards the goal of achieving comprehensive semantic mapping and seamless integration in an environment where ontologies are

continuously evolving. In the following, several research directions are indicated for future work.

1. We are aware of relations other than {hyponym(is-a), part-of} existing in ontology description. In addition, the semantic match by considering axioms of an ontology is one of the toughest questions to be dealt with. To our best understanding, no such work has yet been done sufficiently, even though some researchers mention it and propose their solutions. With all these observations in mind, we are investigating the theoretical background of ontology mapping from an algebraic perspective by treating an ontology as an abstraction in the sense that it can represent any relations with complex structures (axioms). We hope that an algebraic abstraction will allow us to deal with ontology mapping in the same way as familiar computational metaphors.

2. We plan to extend the usage of process algebra to theoretically support agent interactions at a high level of abstraction (in both ontology mapping and integration). We expect that this perspective will allow the proposed framework to deal with ontology management in an abstract yet flexible way.

3. We plan to implement the rest of the functionalities presented in this thesis, particularly those of extension. For example, adding ontology refinement to the implemented prototype to enhance the effectiveness is underway. Moreover, `functionary agents` might be service providers on the Web. The problem of deciding which `functionary agent` is a suitable service provider will also become crucial and we may need to augment the proposed framework to include Web services protocols and languages as Web services proliferate and the need for balance among these increases. An interesting direction to follow will be the development of Web services related to ontology management based on what we have done.

4. As services (catering for e-science) might be provided by Grid, another interesting research direction to examine is how mechanisms developed for distributed environments can be transferred or tailored to solve new types of

problems in ontology management. Although Grid research is in its very early stages, the significant advantage is evident. It is known that the Grid middleware provides computational integration to enable information integration with the presence of ontologies.

Bibliography

[1] Arens, Y., Knoblock, A. C., and Shen, W. M., Query Reformulation for Dynamic Information Integration, *Journal of Intelligent Information Systems, Special Issue on Intelligent Information Integration*, 6(2/3), pp. 99-130, 1996.

[2] Baader, F., and Sattler, U., An Overview of Tableau Algorithms for Description Logics, *Studia Logica*, Vol 69, pp. 5-40, 2001.

[3] Baader, F., and Sattler, U., Tableau Algorithms for Description Logics, In R. Dyckhoff (Eds.), *Proc. of the International Conference on Automated Reasoning with Tableaux and Related Methods* (Tableaux 2000), St Andrews, Scotland, UK, LNAI 1847, pp. 118, 2000.

[4] Baeten, J. C. M., and Weijland, W. P., *Process Algebra*, Cambridge University Press, ISBN 0521400430, 1990.

[5] Barwise, J., and Seligman, J., Information Flow: The Logic of Distributed Systems, Cambridge Tracts in Theoretical Computer Science 44, Cambridge University Press, 1997.

[6] Batini, C., Lenxerini, M., and Navathe, S., A Comparative Analysis of Methodologies for Database Schema Integration, *ACM Computing Surveys*, 18(4), pp. 323-364, 1986.

[7] Bellifemine, F., Poggi, A., and Rimassa, G., Developing Multi Agent Systems with a FIPA-Compliant Agent Framework, *Software - Practice And Experience*, 2001 N31, pp. 103-128, 2001.

[8] Benetti I., Beneventano D., Bergamaschi S., and Guerra F., and Vincini M., An Information Integration Framework for e-commerce, *IEEE Intelligent Systems*, January/February 2002.

[9] Bergamaschi, S., Castano, S., and Vincini, M., Semantic Integration of Semistructured and Structured Data Sources, *Special Issue on Semantic Interoperability in Global Information*, *SIGMOD Record*, 28(1), March 1999.

[10] Bergamaschi, S., Castano, S., Beneventano, D., and Vincini, M., Semantic Integration of Heterogeneous Information Sources, Special Issue on Intelligent Information Integration, *Data & Knowledge Engineering*, 36(3), pp. 215-249, Elsevier, 2001.

[11] Berners-Lee, T., Hendler, J., and Lassila, O., The Semantic Web, *Scientific American*, 284(5), pp.34-43, 2001(5).

[12] Boccara, N., Modelling Complex Systems, Graduate Texts in Contemporary Physics, Springer: New York, NY, USA, 2004.

[13] Bouquet, P., Magnini, B., Serafini, L., and Zanobini, S., A SAT-based Algorithm for Context Matching, *Proc. of 4th International and Interdisciplinary Conference on Modeling and Using Context* (CONTEXT'2003), USA, June 2003.

[14] Byrne, J. A., Brandt, R., and Bort, O., The Virtual Corporation, *Business Week*, Vol. 8, pp.36-40, February 1993.

[15] Calvanese, D., Giacomo, D. G., and Lenzerini, M., A Framework for Ontology Integration, in: Isabel Cruz, Stefan Decker, Jérome Euzenat, and Deborah McGuinness (Eds.), *Proc. of the 1st Semantic Web Working Symposium at the Emerging Semantic Web*, pp. 201-214, 2002.

[16] Castano, S., de Antonellis, V., Fugini, MG, and Pernici, B., Conceptual Schema Analysis: Techniques and Applications, *ACM Trans on Database System*, 23(3), pp. 286-332, 1998.

[17] Castano, S., and Antonellis, V., A Schema Analysis and Reconciliation Tool Environment for Heterogeneous Databases, *Proc. of 1999 International Symposium on Database Engineering & Applications*, IEEE Computer Society, 1999.

[18] Chandrasekaran, B., Johnson, T. R., and Benjamins, V. R., Ontologies: What Are They? Why Do We Need Them?, *IEEE Intelligent Systems and Their Applications, Special Issue on Ontologies*, 14(1), pp. 20-26. 1999.

[19] Chaudhri, V., Farquhar, A., Fikes, R., Karp, P., and Rice, J., OKBC: A Programmatic Foundation for Knowledge Base Interoperability, AAAI-9, 1998.

[20] Compatangelo, E., and Meisel, H., Intelligent Support to Knowledge Sharing through the Articulation of Class Schemas, *Proc. of the 6th International Conference on Knowledge-Based Intelligent Information & Engineering Systems* (KES'02), 2002.

[21] Doan, A., Domingos, P.,and Halevy, A. Y., Reconciling Schemas of Disparate Data Sources: A Machine-Learning Approach, *Proc. of the ACM-SIGMOD Conference on Management of Data* (SIGMOD), Aref WG (Eds.), Santa Barbara, CA, 2001.

[22] Doan, A., Madhavan, J., Dhamankar, R., Domingos, P., and Halevy, A., Learning to Match Ontologies on the Semantic Web, *VLDB Journal*, *Special Issue on the Semantic Web*, 12(4), pp. 303-319, 2003.

[23] Doan, A., Madhaven, J., Domingos, P., and Halevy, A., Ontology Matching: A Machine Learning Approach, *Handbook on Ontologies in Information systems*, Springer-Verlag, pp. 397-416, 2004.

[24] Domingos, P., and Pazzani, M., Conditions for the Optimality of the Simple Bayesian Classifier, *Proc. of the 13th International Conference on Machine Learning*, Bari, Italy, pp. 105-112, July 1996.

[25] Duineveld, J. A., Stoter, R., Weiden, R. M., Kenepa, B., and Benjamins, R. V., WonderTools? A Comparative Study of Ontological Engineering Tools,

International Journal of Human-Computer Studies, 52(6), pp. 1111-1133, June 2000.

[26] Durfee, E. H., and Lesser, V., Negotiating Task Decomposition and Allocation Using Partial Global Planning, *Distributed Artificial Intelligence Volume II*, L. Gasser and M. Huhns (Eds.), Pitman Publishing and Morgan Kaufmann, pp. 229-244, 1989.

[27] Finin, T., McKay, D., Fritzson, R., and McEntire, R., KQML: An Information and Knowledge Exchange Protocol, *Knowledge Building and Knowledge Sharing*, Kazuhiro Fuchi and Toshio Yokoi (Eds.), Ohmsha and IOS Press, 1994.

[28] Foster, I., Jennings, N., and Kesselman, C., Brain Meets Brawn: Why Grid and Agents Need Each Other, *Proc. 3rd International Conference on Autonomous Agents and Multi-Agent Systems*(AAMAS'04), New York, USA, pp. 8-15, 2004.

[29] Fowler, J., Nodine, M., Perry, B., and Bargmeyer, B., Agent Based Semantic Interoperability in InfoSleuth, *SIGMOD Record*, 28(1), pp. 60–67, 1999.

[30] Goasdoué, F., A Knowledge Based Approach for Information Integration: The PICSEL System, in: Nicolas Spyratos, K. Vidyasankar and Gottfried Vossen (Eds.), Declarative Data Access on the Web, Dagstuhl-Seminar-Report 251, p. 7, Dagstuhl Castle, Germany, September 1999.

[31] Goasdoué, F., Lattes, V., and Rousset, M -C., The Use of CARIN Language and Algorithms for Information Integration: The PICSEL Project, *International Journal of Cooperative Information Systems* (IJCIS), World Scientific Publishing Company, 9(4), pp. 383-401, 2000.

[32] Gal, A., Anaby-Tavor, A., Trombetta, A., and Montesi, D., A Framework for Modeling and Evaluating Automatic Semantic Reconciliation, *VLDB Journal*, 14(1), pp. 50-67, 2005.

[33] Gangemi, A., Pisanelli, M. D., and Steve, G., An Overview of the ONIONS Project: Applying Ontologies to the Integration of Medical Terminologies, *Data & Knowledge Engineering*, Vol 31, pp.183-220, 1999.

[34] Ganter, B., and Wille, R., Formal Concept Analysis: Mathematical foundations, Springer, Berlin-Heidelberg, 1999.

[35] Genesereth, M., and Fikes, R., Knowledge Interchange Format (KIF). Draft proposed American National Standard, NCITS T2/98-004, 1998.

[36] Gómez-Pérez, A., and Richard Benjamins, V., Applications of Ontologies and Problem-Solving Methods, *AI Magazine*, 20(1), pp. 119-122, 1999.

[37] Gruber, T. R., Toward Principles for the Design of Ontologies Used for Knowledge Sharing, KSL-93-04, Knowledge Systems Laboratory, Stanford University, *http://ksl-web.stanford.edu/*, 1993.

[38] Gruber, T. R., A Translation Approach to Portable Ontology Specifications, *Knowledge Acquisition*, 5(2), pp. 199-220, 1993.

[39] Guarino, N., and Giaretta, P., Ontologies and Knowledge Bases, towards a Terminological Clarification, in: N. Mars (Eds.), *Towards Very Large Knowledge Bases*, Amsterdam: IOS Press, pp. 25-32, 1995.

[40] Grüninger, M., Fox, S. M., Methodology for the Design and Evaluation of Ontologies, *Proc. of IJCAI'95 Workshop on Basic Ontological Issues in Knowledge Sharing*, Montreal, Canada, 1995.

[41] Grüninger, M., and Kopena, B. J., Semantic Integration, Position Statement, *Proc. of the Workshop on Semantic Integration, the 2nd International Semantic Web Conference*, Sanibal Island, Florida, USA, October 2003.

[42] Haarslev, V., and Möller, R., Practical Reasoning in Racer with a Concrete Domain for Linear Inequations, in: I. Horrocks and S. Tessaris(Eds.), *Proc. of the International Workshop on Description Logics* (DL-2002), Toulouse,France, April 2002.

[43] Haarslev, V., and Möller, R., RACER User's Guide and Reference Manual: Version 1.7.6, December 2002.

[44] Hendler, J., Agents and the Semantic Web, *IEEE Intelligent Systems*, 16(2), March/April 2001.

[45] Horrocks, I., Sattler, U., and Tobies, S., Practical Reasoning for Expressive Description Logics, *Proc. of 6th International Conference on Logic for Programming and Automated Reasoning*(LPAR'99), LNAI, Springer-Verlag, pp. 161-180, 1999.

[46] Hovy, E., Combining and Standardising Large-scale, Practical Ontologies for Machine Translation and Other Uses, *Proc. of the 1st International Conference on Language Resources and Evaluation* (LREC), pp. 535-542, 1998.

[47] Iglesias, C., Garijo, M., and Gonzales, J., A Survey of Agent-Oriented Methodologies, in: A. Rao, J. Muller, and M. Singh (Eds.), *Intelligent Agents IV* (ATAL'98), LNAI, Springer, pp. 317-330, 1999.

[48] Itoh, F., Ueda, T., and Ikeda, Y., Example-Based Frame Mapping Applied to Information Agents for Distributed Information Sources, Systems and Computers in Japan, 30(14), pp.1-11, 1999.

[49] Jennings, N., Sycara, K., and Wooldridge, M., A Roadmap of Agent Research and Development, *Autonomous Agents and Multi-Agent Systems*, Vol. 1, pp.7-38, 1998.

[50] Jennings, N., Agent-Based Computing: Promise and Perils, *Proc. of 16th International Joint Conference on Artificial Intelligence*, AAAI Press, pp. 1429-1436, 1999.

[51] Jennings, N., Agent-Oriented Software Engineering, *Proc. of 12th International Conference on Industrial and Engineering Applications of Artificial Intelligence*, Cairo, Egypt, pp. 4-10, 1999.

[52] Jennings, N., On Agent-Based Software Engineering, *Artificial Intelligence*, Vol 117, pp. 277-296, 2000.

[53] Jennings, N., An Agent-based Approach for Building Complex Software Systems, *Communication ACM*, 44(4), pp. 35-41, 2000.

[54] Jennings, N., and Wooldridge, M., Agent-Oriented Software Engineering, in: J. Bradshaw (Eds.), *Handbook of Agent Technology*, AAAI/MIT Press, 2001.

[55] Jennings, N., An Agent-Based Approach for Building Complex Software Systems, *Communications of the ACM*, 44(4), pp. 35-41, 2001.

[56] Jurisica, I., Mylopoulos, J., and Yu, E., Ontologies for Knowledge Management: An Information Systems Perspective, *Knowledge and Information Systems*, 6(4), pp. 380-401, Springer-Verlag, July 2004.

[57] Kalfoglou, Y., and Schorlemmer, M., Information-Flow-based Ontology Mapping, *Proc. of International Conference on Ontologies, Databases and Applications of Semantics*, California, USA, October 2002.

[58] Kalfoglou, Y., and Schorlemmer, M., Ontology Mapping: The State of the Art, *Knowledge Engineering Review*, 18(1), pp. 1-31, 2003.

[59] Kalfoglou, Y., and Schorlemmer, M., IF-Map: An Ontology-mapping Method Based on Information-flow Theory, *Journal of Data Semantics*, 1(1), pp. 98-127, 2003.

[60] Kiryakov, A., Simov,K., and Dimitrov, M., Ontomap: the Upper-Ontology Portal, *Proc. of Formal Ontology in Information Systems*, Ogunquit, Maine, 2001.

[61] Klein, M., Combining and Relating Ontologies: An Analysis of Problems and Solutions, in: Gomez-Perez, A., Gruninger, M., Stuckenschmidt, H. and Uschold, M. (Eds.), *Workshop on Ontologies and Information Sharing*(IJCAI01), Seattle, WA., August 2001.

[62] Klein, M., and Bernstein, A., Searching for Services on the Semantic Web Using Process Ontologies, *Proc. of 1st Semantic Web Working Symposium*(SWWS-1), Stanford, 2001.

[63] Klein, M., and Fensel, D., Ontology versioning for the Semantic Web, *Proc. of International Semantic Web Working Symposium*, USA, July 2001.

[64] Klein, M., and Noy, F. N., A Component-Based Framework for Ontology Evolution, *Proc. of the IJCAI'03 Workshop: Ontologies and Distributed Systems, International Joint Conference on Artificial Intelligence 2003* (IJCAI'03), Acapulco, Mexico, 2003.

[65] Kraus, S., Automated Negotiation and Decision Making in Multiagent Environment, *Proc. of Advanced Course on Artificial Intelligence* (ACAI 2001), LNAI 2086, pp.150-172, 2001.

[66] Lacher, M., and Groh, G., Facilitating the Exchange of Explicit Knowledge through Ontology Mappings, *Proc. of the 14th International FLAIRS Conference*, 2001.

[67] Labrou, Y., Finin, T., and Peng, Y., Agent Communication Languages: The Current Landscape, *IEEE Intelligent Systems*, March/April 1999.

[68] Langley, P., Iba, W., and Thompson, K., An Analysis of Bayesian Classifiers, *Proc. of the 10th National Conference on Artificial Intelligence*, San Jose, CA, July, pp. 223-228, 1992.

[69] Lenat, B. D., and Guha, V. R., *Building Large Knowledge-Based Systems, Representation and Inference in the CYC project*, Addison-Wesley, Reading, Massachusetts, 1990.

[70] Levenstein I. V., Binary Codes Capable of Correcting Deletions, Insertions, and Reversals, *Cybernet Control Theory*, 10(8), pp. 707-710, 1966.

[71] Li, L., Wu, B., and Yang, Y., Semantic Mapping with Multi-Agent Systems, *Proc. of the IEEE International Conference on e-Technology, e-Commerce and e-Service* (EEE'05), pp. 54-57, Hong Kong, March 2005.

[72] Li, L., Wu, B., and Yang, Y., Ontology-based Matchmaking in e-Marketplace with Web Services, *Proc. of the 6th Advanced Web Technologies and Applications* (APWeb05), LNCS 3399, Springer-Verlag, pp.620-631, Shanghai, China, April 2005.

[73] Li, L., Yang, Y., and Wu, B., Agent-Based Approach towards Ontology Refinement in Virtual Enterprises, *Proc. of the 3rd International Conference on Active Media Technology* (AMT 2005), pp. 220-225, Kagawa, Japan, May 2005.

[74] Li, L., Wu, B., and Yang, Y., Agent-Based Approach for Dynamic Ontology Management, *Proc. of the 9th International Conference on Knowledge-Based Intelligence Information & Engineering Systems*(KES2005), LNCS 3683, Part III, Springer-Verlag, Melbourne, Australia, pp. 1-7, September 2005.

[75] Li, L., Yang, Y., and Wu, B., Agent-Based Ontology Mapping towards Ontology Interoperability, in: S. Zhang and R. Jarvis (Eds.), *Proc. of the 18th Australian Joint Conference on Artificial Intelligence* (AI05), LNAI 3809, Springer-Verlag, Sydney, Australia, pp. 843-846, Sydney, Australia, December 2005.

[76] Li, L., Yang, Y., and Wu, B., Agent-Based Ontology Integration for Ontology-Based Application, in: T. Meyer, M. Orgun (Eds.), *Proc. of Australasian Ontology Workshop* (AOW 2005), the 18th Australian Joint Conference on Artificial Intelligence, Conferences in Research and Practice in Information Technology (CRPIT) series by Australian Computer Society, Vol 58, pp. 53-59, Sydney, Australia, December 2005.

[77] Li, L., Yang, Y., and Wu, B., Implementation of Agent-based Ontology Mapping and Integration, Technical Report, Swinburne University, *http://www.it.swin.edu.au/personal/yyang/papers/2005TR-Li-1.pdf*.

[78] López, F., Overview of Methodologies for Building Ontologies, *Proc. of the IJCAI'99 Workshop on Ontologies and Problem-Solving Methods: Lessons Learned and Future Trends*, CEUR Publications, 1999.

[79] Luck, M., and d'Inverno, M., A Formal Framework for Agency and Autonomy, *Proc. of the 1st International Conference on Multi-Agent Systems*, AAAI Press/MIT Press, pp. 254-260, 1995.

[80] Luck, M., Munroe, S., and d'Inverno, M., Autonomy: Variable and Generative, in: H. Hexmoor, C. Castelfranchi, and R. Falcone, (Eds.), *Agent Autonomy*, Kluwer, pp. 9-22, 2003

[81] Luck, M., McBurney, P., and Preist, C., A Manifesto for Agent Technology: Towards Next Generation Computing, *Autonomous Agents and Multi-Agent Systems*, 9(3), pp. 203-252, 2004.

[82] MacGregor, R., Using A Description Classifier to Enhance Deductive Inference, *Proc. 7th IEEE Conference on AI Application*, Florida, pp. 93-97, 1991.

[83] Madhavan, J., Bernstein, P. A., Rahm, E., Generic Schema Matching with Cupid, *Proc. of the 27th International Conference on Very Large Data Bases* (VLDB), Rome, Italy, pp. 49-58, September 2001.

[84] Maedche, A., *Ontology Learning for the Semantic Web*, Kluwer Academic Publishers, 2002.

[85] Maedche, A., Motik, B., Silva, N., and Volz, R., MAFRAA Mapping FRAmework for Distributed Ontologies, in: A. Gomez-Perez and V.R. Benjamins (Eds.), *Proc. of the 13th International Conf. of Knowledge Engineering and Knowledge Management* (EKAW 2002), pp. 235-250, Siguenza, Spain, October 2002.

[86] Maedche, A., and Staab, S., Measuring Similarity between Ontologies, in: A. Gomez-Perez and V.R. Benjamins (Eds.), *Proc. of the 13th International Conference on Knowledge Engineering and Knowledge Management*(EKAW2002), Siguenza, Spain, pp. 251-263, October 2002.

[87] Maedche, A., Motik, B., Stojanovic, L., Studer, R., and Volz, R., Managing Multiple Ontologies and Ontology Evolution in Ontologging, *Proc. of the Conference on Intelligent Information Processing, World Computer Congress*, Montreal, Canada, pp. 51-63, 2002.

[88] Maedche, A., Motik, B., Stojanovic, L., Studer, R. and Volz, R., An Infrastructure for Searching, Reusing and Evolving Distributed Ontologies, *Proc. of International Conference of WWW* (WWW2003), pp. 439-448, 2003.

[89] McGuinness, D., Conceptual Modelling for Distributed Ontology Environments, *Proc. of International Conference on Conceptual Structures*, pp. 100-112, 2000.

[90] McGuinness, D., Fikes, R., Rice, J., and Wilder, S., The Chimaera Ontology Environment, *Proc. of the 7th National Conference on Artificial Intelligence* (AAAI 2000), Austin, Texas, August 2000.

[91] McGuinness, D., Fikes, R., Rice, J., and Wilder, S., An Environment for Merging and Testing Large Ontologies, *Proc. of the 7th International Conference on Principles of Knowledge Representation and Reasoning* (KR2000), pp. 483-493, Breckenridge, Colorado, USA, April 2000.

[92] Mena, E., Illarramendi, A., Kashyap, V., and Sheth, A.P. OBSERVER: An Approach for Query Processing in Global Information Systems Based on Interoperation across Pre-existing Ontologies, *International Journal Distributed and Parallel Databases* (DAPD), 8(2), pp. 223-271, ISSN 0926-8782, April 2000.

[93] Menzis, T., Knowledge Maintenance: The State of the Art, *The Knowledge Engineering Review*, 10(2), 1998.

[94] Miller, R., Haas, L., and Hernández, L., Schema Mapping as Query Discovery, *Proc. of the International Conference on Very Large Databases* (VLDB), 2000.

[95] Miller, R., Hernández, M., Haas, L., Yan, L., Howard Ho, C. T., Fagin, R., and Popa, L., The Clio Project: Managing Heterogeneity, *SIGMOD Record*, 30(1), pp. 78-83, 2001.

[96] Milner, R., Parrow, J., and Walker, D., A Calculus of Mobile Processes, *Pars I and II, Journal of Information and Computation*, 100(7), PP. 1-77, 1992.

[97] Mitra, P., Wiederhold, G., and Kersten, M., A Graph-Oriented Model for Articulation of Ontology Interdependencies, *Proc. of Conference on Extending Database Technology* (EDBT 2000), Konstanz, Germany, 2000.

[98] Mitra, P., and Wiederhold, G., An Algebra for Semantic Interoperability of Information Sources, *IEEE International Conference on Bioinformatics and Biomedical Engineering*, pp. 174-182, 2001.

[99] Mitra, P., Wiederhold, G., and Decker, S., A Scalable Framework for Interoperation of Information Sources, *The Emerging Semantic Web*, Cruz, I., Decker, S., in: Euzenat, J., and McGuinness, D. (Eds.), selected Papers from the first Semantic Web Symposium, Amsterdam (NL), ISBN: 1586032550, IOS press, 2002.

[100] Mitra, P., and Wiederhold, G., *An Ontology Composition Algebra: Handbook on Ontologies in Information Systems*, in: S. Staab, and R. Studer(Eds.), Springer-Verlag series on International Handbooks on Information Systems, ISBN 3540408347, pp. 93-117, 2004.

[101] Modica, G., Gal, A., and Jamil, H., The Use of Machine Generated Ontologies in Dynamic Information Seeking, in: Batini C, Giunchiglia F, Giorgini P, MecellaM(Eds.), *Proc. of the 9th International Conference on Cooperative Information Systems* (CoopIS 2001), Trento, Italy, LNCS 2172, Springer, pp. 433-448, September 2001.

[102] Motta, E., Reusable Components for Knowledge Modelling: Case Studies in Para-metric Design Problem Solving, *Frontiers in Artificial Intelligence and Applications*, Vol 53, IOS Press, 1999.

[103] Nash, F. J., Equilibrium Points in N-Person Games, *Proc. of NAS*, 1950.

[104] Nestorov, S., Abiteboul, S., and Motwani, R., Extracting Schema from Semistructured Data, in: Haas LM, TiwaryA (Eds.), *Proc. of the ACM-SIGMOD Conference on Management of Data* (SIGMOD), Seattle, ACM Press, New York, pp. 295-306, June 1998.

[105] Nodine, M., Fowler, J., Ksiezyk, T., Perry, B., Taylor, M., and Unruh, A., Active Information Gathering in InfoSleuth, *International Journal of Cooperative Information Systems*, 9(1-2), pp. 328, 2000.

[106] Noy, F. N., and Musen, M. A., PROMPT, Algorithm and Tool for Automated Ontology Merging and Alignment, *Proc. of 17th National Conference on Artificial Intelligence (AAAI-2000)*, Austin, TX, pp. 450-455, August 2000.

[107] Noy, F. N., What Do We Need for Ontology Integration on the Semantic Web, Position Statement, *Proc. of the Workshop on Semantic Integration, the 2nd International Semantic Web Conference*, Sanibal Island, Florida, USA, October 2003.

[108] Noy, F. N., and Musen, M. A., The PROMPT Suite: Interactive Tools for Ontology Merging and Mapping, *International Journal of Human-Computer Studies*, 59(6), pp. 983-1024, 2003.

[109] Noy, F. N., Semantic Integration: A Survey of Ontology-Based Approaches, *SIGMOD Record, Special Issue on Semantic Integration*, 33(4), pp. 65-70, December, 2004.

[110] Noy., F. N., and Klein, M.,. Ontology Evolution: Not the Same as Schema Evolution, *Knowledge and Information Systems*, 6(4), pp. 428-440, 2004.

[111] Obitko, M., and Marik, V., Mapping between Ontologies in Agent Communication, Springer-Verlag, LNAI 2691, pp. 191-203, 2003.

[112] Odell, J., Parunak, H. V. D., and Bauer, B., Extending UML for Agents, in: G. Wagner, Y. Lesperance, and E. Yu (Eds.), *Proc. of the Agent Oriented Information Systems Workshop (AOIS) at the 17th National Conference on Artificial Intelligence*, Austin, Texas, pp.3-17, 2000.

[113] Omelayenko, B., and Fensel, D., A Two-layered Integration Approach for Product Information in B2B E-Commerce, *Proc. of the 2nd Intenational Conference on Electronic Commerce and Web Technologies* (EC WEB-2001), Munich, Germany, Springer-Verlag, 2001.

[114] Omelayenko, B., Integrating Vocabularies: Discovering and Representing Vocabulary Maps, *Proc. of the 1st International Semantic Web Conference* (ISWC2002), Sardinia, Italy, 2002.

[115] Omelayenko, B., RDFT: A Mapping Meta-Ontology for Business Integration, *Proc. of the Workshop on Knowledge Transformation for the Semantic Web* (KTSW 2002) at the 15th European Conference on Artificial Intelligence, Lyon, France, pp. 76-83, July 2002.

[116] Palopoli, L., Terracina, G., and Ursino, D., The System DIKE: towards the Semi-Automatic Synthesis of Cooperative Information Systems and Data Warehouses, *Proc. of Current Issues in Databases and Information Systems, East European Conference on Advances in Databases and Information Systems*, jointly held with the International Conference on Database Systems for Advanced Applications (ADBIS-DASFAA 2000), Prague, Czech Republic, pp. 108-117, September 2000.

[117] Pinto, H. S., Gómez-Pérez, A., and Martins, P. J., Some Issues on Ontology Integration, *Workshop: Ontologies and Problem Solving Methods, International Joint Conference on Artificial Intelligence* (IJCAI'99), Stockholm, Sweden, 1999.

[118] Pinto, H. S., and Martins, P. J., A Methodology for Ontology Integration, *Proc. of the International Conference on Knowledge Capture*, Victoria, British Columbia, Canada, pp. 131-138, 2001.

[119] Pinto, H. S., and Martins, P. J., Ontologies: How Can They Be Built?, *Knowledge and Information Systems*, 6(4), pp. 441-464, Springer, July 2004.

[120] Pnueli, A., Specification and Development of Reactive Systems, *Information Processing 86*, pp. 845-858, Elsevier, Amsterdam.

[121] Prasad, S., Peng, Y., and Finin, T., Using Explicit Information to Map between Two Ontologies, Proc. of the AAMAS 2002 Workshop on Ontologies in Agent Systems (OAS'02), pp. 52-57, 2002.

[122] Popa, L., Velegrakis, Y., Hernández, M., Miller, R., and Fagin, R., Translating Web Data, *Proc. of the International Conference on Very Large Databases*(VLDB), 2002.

[123] Rahm, E., and Bernstein, P., A Survey of Approaches to Automatic Schema Matching, *The VLDB Journal*, Vol 10, pp. 334-350, 2001.

[124] Rodrguez, A. M, and Egenhofer, J. M., Determining Semantic Similarity among Entity Classes from Different Ontologies, *IEEE Transactions on Knowledge and Data Engineering*, 2000.

[125] Rubinstein, A., Perfect Equilibrium in a Bargaining Model, *Econometrica*, 50(1), pp. 97-109, 1982.

[126] Russell, S., and Norvig, P., *Artificial Intelligence: A Modern Approach*, New York Prentice-Hall, 1995.

[127] Schorlemmer, M., Duality in Knowledge Sharing, *Proc. of the 17th International Symposium on Artificial Intelligence and Mathematics*, Fort Lauderdale, Florida (USA), January 2002.

[128] Sheth, A., and Larson, J., Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases, *ACM Computing Surveys*, 22(3), pp. 183-236, 1990.

[129] Sheth, A., and Rusinkiewicz, M., On Transactional Workflows, *The Data Engineering Bulletin*, 16(2), pp. 37-40, 1993.

[130] Silva, N. and Rocha, J., Complex Semantic Web Ontology Mapping, *Web Intelligence and Agent Systems: An International Journal*, 1(3/4), pp. 234-248, 2003.

[131] Silva, N., and Rocha, J., Ontology Mapping for Interoperability in Semantic Web, In: *Proc. of the IADIS International Conference WWW/Internet 2003* (ICWI2003), Algarve, Portugal, 2003.

[132] Stojanovic, L., Maedche, A., Motik, B., and Stojanovic, N., User-driven Ontology Evolution Management, *Proc. of the 13th European Conference*

on Knowledge Engineering and Knowledge Management (EKAW), Madrid, Spain, 2002.

[133] Stumme, G., and Maedche, A., FCA-Merge: Bottom-Up Merging of Ontologies, *IJCAI'01 Workshop on Ontologies and Information Sharing* (jointly held with the 17th International Joint Conference on Artificial Intelligence) (IJCAI'01), Seattle, USA, August 2001.

[134] Sure, Y., Erdmann, M., Angele, J., Staab, S., Studer, R., and Wenke, D., OntoEdit: Collaborative Ontology Engineering for the Semantic Web, *Proc. of the International Semantic Web Conference 2002*(ISWC 2002), LNCS 2342, Springer-Verlag, pp. 221-235, Sardinia, Italy, June 2002.

[135] Tamma, V., and Bench-Capon, T., A Conceptual Model to Facilitate Knowledge Sharing in Multi-agent Systems, *Proc. of the OAS 2001*, Montreal, pp. 69-76, 2001.

[136] Teknowledge: High-Performance Knowledge-Bases (HPKB), *http://projects.teknowledge.com/HPKB/*, accessed on 15th May 2005.

[137] Tveit, A., A Survey of Agent-Oriented Software Engineering, *http://www.jfipa.org/publications/AgentOrientedSoftware Engineering*.

[138] Uschold, M., and King, M., Towards a Methodology for Building Ontologies, *Workshop on Basic Ontological Issues in Knowledge Sharing*, held in conjunction with IJCAI-95, Montreal, Canada, 1995.

[139] Uschold, M., and Grueninger, M., Ontologies: Principles, Methods and Applications, *Knowledge Sharing and Review*, 11(2), pp. 93-155, 1996.

[140] Uschold, M., Healy, M., Williamson, K., Clark, P., and Woods, S., Ontology Reuse and Application, in: N. Guarino(Eds.), *Proc. of Formal Ontology in Information Systems* (FOIS'98), Treno, Italy, June 1998.

[141] Wang, J., and Gasser, L., Mutual Online Ontology Alignment, In: Stephen Cranefield, Tim Finin, and Steve Willmott (Eds.), *Proc. of the Workshop*

on Ontologies in Agent Systems, 1st International Joint Conference on Autonomous Agents and Multi-Agent Systems, CEUR Workshop Series Volume 66, Bologna, Italy, July 2002.

[142] Wache, H., Vögele, T., Visser, U., Stuckenschmidt, H., Schuster, G., Neumann, H., and Hübner, S., Ontology-Based Integration of Information - A Survey of Existing Approaches, *Proc. of the IJCAI-01 Workshop: Ontologies and Information Sharing, International Joint Conference on Artificial Intelligence* (IJCAI'01), 2001.

[143] Wiederhold, G., An Algebra for Ontology Composition, *Proc. of Monterey Workshop on Formal Methods*, Monterey CA, pp. 56-61, September 1994.

[144] Wiesman, F., and Roos, N., Domain Independent Learning of Ontology Mappings, *Proc. of the 3rd International Joint Conference on Autonomous Agents & Multi Agent Systems* (AAMAS04), New York, USA, 2004.

[145] Wille, R., Restructuring Lattice Theory: An Approach Based on Hierarchies of Concepts, In: I. Rival(Eds.), *Ordered Sets*, Reidel, Dordrecht-Boston, pp. 445-470, 1982.

[146] Wooldridge, M., and Jennings, N. R., Intelligent Agents: Theory and Practice, *Knowledge Engineering Review*, 10(2), pp. 115-152, 1995.

[147] Wooldridge, M., Agent-Based Software Engineering, *IEE Proc. Software Engineering*, 144(1), pp. 26-37, 1997.

[148] Wooldridge, M., Jennings, N., and Kinny, D., A Methodology for Agent-Oriented Analysis and Design, *Proc. of the 3rd International Conference on Autonomous Agents*, Seattle, WA, pp. 69-76, 1999.

[149] Wooldridge, M., and Ciancarini, P., Agent-Oriented Software Engineering: The State of the Art, in: P. Ciancarini and M. Wooldridge (Eds.), *Agent-Oriented Software Engineering*, Springer, LNAI 1957, pp. 1-28, January 2001.

[150] Wooldridge, M., *An Introduction to MultiAgent Systems*, John Wiley & Sons, ISBN 047149691X, 2002.

[151] Wu, B., Dewan, M., Li, L. and Yang, Y., Supply Chain Protocolling, *IEEE Conference on E-Commerce Technology* (CEC'05), pp. 314-321, München, Germany, July 2005.

[152] Zambonelli, F., and Parunak, H., Signs of a Revolution in Computer Science and Software Engineering, in: P. Petta, R. Tolksdorf, and F. Zambonelli (Eds.), *Engineering Societies in the Agents World* (ESAW 2002), Springer: Berlin, Germany, pp. 13-28, 2003.

[153] Zambonelli, F., Jennings, N., Omicini, A., and Wooldridge, M., Agent-Oriented Software Engineering for Internet Applications, in: A. Omicini, F. Zambonelli, M. Klusch, and R. Tolksdorf (Eds.), *Coordination of Internet Agents: Models, Technologies and Applications*, Springer, pp. 326-346, 2001.

[154] Zambonelli, F., Jennings, N., and Wooldridge, M., Organizational Abstractions for the Analysis and Design of Multi-Agent Systems, in: P. Ciancarini and M. Wooldridge (Eds.), *Agent-Oriented Software Engineering*, Springer, LNAI 1957, January 2001.

[155] Zou,Y., Finin T., and Chen, H., F-OWL: An Inference Engine for the Semantic Web, in: M. Hinchey, J. Rash, W. Truszkowski, and C. Rouff (Eds.), *Formal Approaches to Agent-Based Systems*, the 3rd International Workshop (FAABS2004), Greenbelt, MD, USA, April, 2004, revised paper, LNCS 3228, Springer, ISBN 3-540-24422-0, pp. 238-248, 2005.

[156] *http://www-ksl.stanford.edu/kst/what-is-an-ontology.html.*