Towards an ontology-based knowledge management: An ontology mediation framework to reconcile inter-organizational knowledge

Nelson K. Y Leung
University of Wollongong
UNIVERSITY OF WOLLONGONG

COPYRIGHT WARNING

You may print or download ONE copy of this document for the purpose of your own research or study. The University does not authorise you to copy, communicate or otherwise make available electronically to any other person any copyright material contained on this site. You are reminded of the following:

Copyright owners are entitled to take legal action against persons who infringe their copyright. A reproduction of material that is protected by copyright may be a copyright infringement. A court may impose penalties and award damages in relation to offences and infringements relating to copyright material. Higher penalties may apply, and higher damages may be awarded, for offences and infringements involving the conversion of material into digital or electronic form.
Towards an Ontology-based Knowledge Management: An Ontology Mediation Framework to Reconcile Inter-organizational Knowledge

A thesis submitted in (partial) fulfilment of the requirements for the award of the degree

Doctor of Philosophy

from

University of Wollongong

by

Nelson K. Y. Leung

Master of Information Systems (Research)
Master of Information Systems
Graduate Certificate of Tertiary Teaching and Learning
Bachelor of Information Technology

School of Information Systems and Technology
Faculty of Informatics
2012
I, Nelson K. Y. Leung, declare that this thesis, submitted in partial fulfilment of the requirement for the award of Doctor of Philosophy, in the School of Information Systems and Technology, University of Wollongong, is wholly my own work unless otherwise referenced or acknowledged. The document has not been submitted for qualifications at any other academic institution.

Nelson K. Y. Leung
January 2012
LIST OF PUBLICATIONS

This is a list of refereed journal and conference papers as well as book chapter that are related to this research work.

**Refereed Journal Papers**


**Refereed Conference Papers**


**Refereed Book Chapters**

# TABLE OF CONTENTS

THESIS CERTIFICATION ...................................................................................................................... I
LIST OF PUBLICATIONS .................................................................................................................. II
TABLE OF CONTENTS ..................................................................................................................... IV
LIST OF FIGURES .......................................................................................................................... VI
LIST OF TABLES ............................................................................................................................ VIII
LIST OF ABBREVIATIONS ............................................................................................................... IX
ABSTRACT ...................................................................................................................................... XI
ACKNOWLEDGEMENT ................................................................................................................... XIII

CHAPTER 1: BACKGROUND AND INTRODUCTION ......................................................................... 1
   1.1 RESEARCH PROBLEMS ........................................................................................................... 2
   1.2 RESEARCH AIMS .................................................................................................................... 5
   1.3 RESEARCH OBJECTIVES ....................................................................................................... 5
   1.4 RESEARCH METHOD AND APPROACH .................................................................................. 5
   1.5 RESEARCH CONTRIBUTION ................................................................................................. 6
   1.6 THESIS ORGANIZATION ..................................................................................................... 7

CHAPTER 2: LITERATURE REVIEW ................................................................................................. 9
   2.1 KNOWLEDGE MANAGEMENT ............................................................................................... 9
   2.2 ONTOLOGY .......................................................................................................................... 16
      2.2.1 Background of Ontology ............................................................................................... 18
      2.2.2 Web Ontology Languages ............................................................................................ 23
      2.2.3 Ontology Mismatches and Mediation .......................................................................... 26
   2.3 CONCLUSION ......................................................................................................................... 42

CHAPTER 3: A SURVEY OF ONTOLOGY MATCHING TECHNIQUES ............................................. 45
   3.1 A REVIEW OF ONTOLOGY MATCHING TECHNIQUES .......................................................... 49
      3.1.1 Shvaiko and Euzenat’s Classification of Ontology Matching Techniques ......................... 49
      3.1.2 Literature Survey on Ontology Mediation Approaches ................................................... 52
   3.2 A DESIGN AND INPUT-SPECIFIC CLASSIFICATION OF ONTOLOGY MATCHING TECHNIQUES ...... 65
   3.3 CONCLUSION ......................................................................................................................... 73

CHAPTER 4: METHODOLOGY OF INTEGRATION-ORIENTED ONTOLOGY DEVELOPMENT .......... 75
   4.1 ANALYSIS OF ONTOLOGY DEVELOPMENT METHODOLOGY ........................................... 76
LIST OF FIGURES

Figure 2.1 Five Stages of Knowledge Management ................................................................. 12
Figure 2.2 Ontology Classification by Guarino (1997, p. 7) ....................................................... 21
Figure 2.3 Ontology Classification by Lassila and McGuinness (2001) ..................................... 22
Figure 2.4 Classification of Ontology Mediation ...................................................................... 30
Figure 2.5 Top-level Mapping Approach .................................................................................. 32
Figure 2.6 One-to-one Mapping Approach ............................................................................. 34
Figure 2.7 Generic Mapping Process (Predoiu et al. 2006) .......................................................... 35
Figure 2.8 Mapping Process of MAFRA (Maedche et al. 2002, p. 237) ....................................... 36
Figure 2.9 Merging Process ...................................................................................................... 39
Figure 2.10 Ontology Integration Methodology by Pinto and Martins (2004) ............................. 41
Figure 3.1 Ontology Matching Process (Ehrig and Staab 2004) .................................................... 45
Figure 3.2 Shvaiko and Euzenat’s (2005, p. 154) Classification of Elementary Ontology / Schema Matching Techniques ................................................................. 50
Figure 3.3 Diagrammatic View of DISCOMT ........................................................................... 68
Figure 4.1 Example of Ontology Integration ............................................................................. 83
Figure 4.2 Six Phases of Development Process in MIOD ......................................................... 86
Figure 4.3 Sample Template of Inspiration Scenario ................................................................. 88
Figure 4.4 Ontology (O1) Used to Demonstrate Distribution Level of Semantically Identical Concepts .................................................................................................................. 90
Figure 4.5 Ontology (O2) Used to Demonstrate Distribution Level of Semantically Identical Concepts .................................................................................................................. 90
Figure 4.6 Proposed Integration-oriented Candidate Ontology Evaluation System (ICOES) ........... 91
Figure 4.7 Demonstration of Ontology Matching Techniques in ICOES (Example 1) ................. 92
Figure 4.8 Demonstration of Ontology Matching Techniques in ICOES (Example 2) ................. 92
Figure 4.9 Demonstration of Ontology Matching Techniques in ICOES (Example 3) .................. 93
Figure 5.1 One-to-one/Top-Level Mapping Approach Selection Flow Chart ......................... 104
Figure 5.2 Mediation Method (Mapping/Merging/Integration) Selection Flow Chart ............... 105
Figure 5.3 Fully-/Semi-Automatic Mediation Selection Flow Chart ........................................ 106
Figure 5.4 Single/Multiple Matching Technique(s) Selection Flow Chart ................................ 108
Figure 6.1 Logical Design of the Prototype .............................................................................. 115
Figure 6.2 Physical Design of the Prototype ............................................................................. 116
Figure 6.3 Extended View of the Server of the Prototype ......................................................... 118
Figure 6.4 Sample Screen of the Prototype .............................................................................. 120
Figure 6.5 Four Categories of Help Desk Enquiry in Ontology1 ................................................. 121
Figure 6.6 Extended View of Ontology1 in Hardware Problem Class ....................................... 122
Figure 6.7 Extended View of Ontology1 in IT Admin Issue Class .................................................... 123
Figure 6.8 Extended View of Ontology1 in Software Problem Class ................................................ 123
Figure 6.9 Extended View of Ontology1 in Other Problem Class ..................................................... 124
Figure 6.10 Details of Extontology1 ................................................................................................. 125
Figure 6.11 Details of Extontology2 ................................................................................................. 125
Figure 6.12 Details of Extontology3 ................................................................................................. 125
Figure 6.13 Functionalities of the Prototype ..................................................................................... 127
Figure 6.14 Sample Screen of Navigation Menu ................................................................................. 131
Figure 6.15 Sample Screen of Retrieving Organizational Knowledge in accordance with ITA’s Ontology .................................................................................................................. 132
Figure 6.16 Sample Screen of Organizational Knowledge Retrieved from ITA ................................. 133
Figure 6.17 First Sample Screen of Retrieving Inter-organizational Knowledge in accordance with ITC’s Ontology (Concepts Retrieved from Organizational Ontology) .............................................................. 134
Figure 6.18 Second Sample Screen of Retrieving Inter-organizational Knowledge based on ITC’s Ontology (Mappings between Ontology1 and Extontology3 as well as Concepts Retrieval from Inter-organizational Ontology) ....................................................................................... 135
Figure 6.19 Sample Screen of Inter-organizational Knowledge Retrieved from ITC ......................... 135
Figure 6.20 Sample Screen of User Interface of NeOn Toolkit’s Alignment Plugin ............................... 136
Figure 6.21 Sample Screen Matching Results before Applying Threshold, b) after Applying Threshold ................................................................................................................................. 137
Figure 6.22 Sample Screen of Specifying Threshold .......................................................................... 138
Figure 6.23 Sample Screen Matching Results after Applying Threshold ........................................... 138
Figure 6.24 Sample Screen of Importing Mapping ............................................................................ 138
Figure 6.25 Inspiration Scenario of “Finding Solutions to Web Application Errors” ......................... 140
Figure 6.26 Sample Screen of Candidate Ontology Evaluating Function ........................................ 141
Figure 6.27 Sample Screen of Evaluation Results of IT Admin Key Terms and Ontology1 ................. 142
Figure 6.28 Knowledge Module of Source Ontology from ITA (Ontology1) ...................................... 144
Figure 6.29 Knowledge Module of Source Ontology from ITB (Extontology2) ............................... 144
Figure 6.30 Knowledge Module of Source Ontology from ITC (Extontology3) ............................... 144
Figure 6.31 Knowledge Module of Source Ontology from ITE (Extontology1) ............................... 144
Figure 6.32 Sample of the Integrated Ontology of ITD ..................................................................... 145
LIST OF TABLES

Table 2.1 Summary of Ontology Mapping, Merging and Integration .................................................. 42
Table 3.1 Summary of Mediation Approaches Discussed in Literature Survey ........................................ 54
Table 3.2 Summary of Mediation Tools, Frameworks and Methods, and their Inherent Matching Techniques .................................................................................................................. 64
Table 4.1 Summary of Phases, Activities and Characteristics in Ten Ontology Development Methodologies ........................................................................................................................................... 79
Table 4.2 Summary of Definition and Selection Guidelines of Ontology Integration and Merging........ 84
Table 6.1 Summary of KMS and Ontology Possessed by the KM Network Participants .................... 114
Table 6.2 Summary of Matchers in Alignment Plugin of NeOn Toolkit (Fu 2011) ............................... 116
Table 6.3 Summary of Organizational-level Decision of ITA and ITD ............................................... 130
Table 6.4 Set of Categorised Keywords Identified by University D .................................................. 141
Table 6.5 Summary of Ontology Evaluation for ITD ....................................................................... 143
Table 7.1 Summary of Features and Contributions of DISCOMT, MIOD and CIK-Net ................. 150
**LIST OF ABBREVIATIONS**

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>ASMOV</td>
<td>Automated Semantic Matching of Ontology with Verification</td>
</tr>
<tr>
<td>CIDER</td>
<td>Context and Inference Based Aligner</td>
</tr>
<tr>
<td>CIK-Net</td>
<td>Collaborative Inter-organizational KM Network</td>
</tr>
<tr>
<td>COMA</td>
<td>Combining Match Algorithm</td>
</tr>
<tr>
<td>DICE</td>
<td>Distribution, Interaction, Competition and Evolution</td>
</tr>
<tr>
<td>DISCOMT</td>
<td>Design and Input-Specific Classification of Ontology Matching Technique</td>
</tr>
<tr>
<td>DL</td>
<td>Description Logic</td>
</tr>
<tr>
<td>DTD</td>
<td>Document Type Definition</td>
</tr>
<tr>
<td>GOM</td>
<td>Generic Ontology Matching</td>
</tr>
<tr>
<td>HTML</td>
<td>Hypertext Markup Language</td>
</tr>
<tr>
<td>HTTP</td>
<td>Hypertext Transfer Protocol</td>
</tr>
<tr>
<td>ICOES</td>
<td>Integration-oriented Candidate Ontology Evaluation System</td>
</tr>
<tr>
<td>IF-Map</td>
<td>Information-Flow-Based Method for Ontology Mapping</td>
</tr>
<tr>
<td>IS</td>
<td>Information Systems</td>
</tr>
<tr>
<td>IT</td>
<td>Information Technology</td>
</tr>
<tr>
<td>ITA</td>
<td>IT help desk of University A</td>
</tr>
<tr>
<td>ITB</td>
<td>IT help desk of University B</td>
</tr>
<tr>
<td>ITC</td>
<td>IT help desk of University C</td>
</tr>
<tr>
<td>ITD</td>
<td>IT help desk of University D</td>
</tr>
<tr>
<td>ITE</td>
<td>IT Solution Provider E</td>
</tr>
<tr>
<td>ITTalks</td>
<td>Information Technology Talks</td>
</tr>
<tr>
<td>JAD</td>
<td>Joint Application Development</td>
</tr>
<tr>
<td>JSP</td>
<td>JavaServer Pages</td>
</tr>
<tr>
<td>KM</td>
<td>Knowledge Management</td>
</tr>
<tr>
<td>KMS</td>
<td>Knowledge Management Systems</td>
</tr>
<tr>
<td>KOM-Frame</td>
<td>KM-network-based Ontology Mediation Framework</td>
</tr>
<tr>
<td>LOM</td>
<td>Large Scale Ontology Matching</td>
</tr>
<tr>
<td>MAFRA</td>
<td>Ontology Mapping Framework</td>
</tr>
<tr>
<td>MIOD</td>
<td>Methodology of Integration-oriented Ontology Development</td>
</tr>
<tr>
<td>MOMIS</td>
<td>Mediator Environment for Multiple Information Sources</td>
</tr>
<tr>
<td>NOM</td>
<td>Naïve Ontology Mapping</td>
</tr>
<tr>
<td>OKBC</td>
<td>Open Knowledge Base Connectivity</td>
</tr>
<tr>
<td>OLA</td>
<td>OWL-Lite Alignment</td>
</tr>
<tr>
<td>Acronym</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
</tr>
<tr>
<td>OMEN</td>
<td>Ontology Mapping Enhancer</td>
</tr>
<tr>
<td>OOAD</td>
<td>Object Oriented Analysis and Design</td>
</tr>
<tr>
<td>OWL</td>
<td>OWL</td>
</tr>
<tr>
<td>PSL</td>
<td>Process Specification Language</td>
</tr>
<tr>
<td>RAD</td>
<td>Rapid Application Development</td>
</tr>
<tr>
<td>RDF</td>
<td>Resource Description Framework</td>
</tr>
<tr>
<td>RDFT</td>
<td>RDF Transformation</td>
</tr>
<tr>
<td>QOM</td>
<td>Quick Ontology Mapping</td>
</tr>
<tr>
<td>SBO</td>
<td>Semantic Bridge Ontology</td>
</tr>
<tr>
<td>SDLC</td>
<td>Systems Development Life Cycle</td>
</tr>
<tr>
<td>SOM</td>
<td>Semantic Ontology Matching</td>
</tr>
<tr>
<td>T1</td>
<td>Tier One</td>
</tr>
<tr>
<td>T2</td>
<td>Tier Two</td>
</tr>
<tr>
<td>T3</td>
<td>Tier Three</td>
</tr>
<tr>
<td>UNSPSC</td>
<td>Universal Standard Products and Services Classification</td>
</tr>
<tr>
<td>URI</td>
<td>Uniform Resource Identifier</td>
</tr>
<tr>
<td>WWW</td>
<td>World Wide Web</td>
</tr>
<tr>
<td>XML</td>
<td>Extensible Markup Language</td>
</tr>
</tbody>
</table>
Knowledge enables organizations to utilize and develop resources, enhance their competitiveness and develop sustainable competitive advantage. Knowledge management aims to manage and capitalize on knowledge by organizing formal and direct process to manage organizational knowledge in the workplace. Literature has shown that a number of knowledge management approaches have been developed with the purpose of managing organizational knowledge. However, these designs only focus on managing intra-organizational knowledge which is inadequate in the current business environment because users often require to access inter-organizational knowledge from other knowledge sources in order to complete tasks in current knowledge explosion era. Furthermore, current knowledge management approaches also fail to make various knowledge management systems inter-operable and collaborative in nature due to individual knowledge management system is designed and developed based on business and knowledge management requirement. As a result, knowledge workers need to spend additional effort to search for relevant knowledge from various knowledge bases and knowledge engineers have to spend a lot of resources to create and update organizational knowledge which may also be available in other knowledge management systems.

This research examines an ontology-based knowledge management approach to enable the interoperability of heterogeneous knowledge management systems in the domain of reusing inter-organizational knowledge. An ontology-based Collaborative Inter-organizational Knowledge Management Network is proposed that incorporates ontology and its mediation methods to reuse inter-organizational knowledge to support knowledge creation and dissemination in the organizational knowledge management process. This research also investigates a theoretical ontology mediation framework to develop an integrated ontology by reusing inter-organizational ontologies. A Methodology of Integration-oriented Ontology Development is proposed to address the lack of details and insights in ontology integration in the current literature. The proposed methodology is designed to provide a detailed description of phases on how to incorporate and perform integration in its ontology development process. A semi-automatic tool called
Integration-oriented Candidate Ontology Evaluation System is included as a component of the proposed methodology to assist in finding suitable source ontologies from a group of candidate ontologies using concept distribution counter and ontology matching techniques. In addition, a Design and Input-Specific Classification of Ontology Matching Technique is proposed to provide guidelines on designing a new mediation tool and method to identify appropriate matching technique and its related executive approach. This research makes research contributions in the area of applying ontology and its mediation methods to develop and manage inter-organizational knowledge management process.
I would like to acknowledge a few people who have helped to make my dream come true.

Firstly, I would like to thank my supervisor, Dr Sim Kim Lau, for providing support, guidance and inspiration in my thesis. Her advice did not just limit to my PhD research but also to my career. She always helped me looking for the best option.

Secondly, to my co-supervisor, Dr Joshua Fan, for spending time listening to me and giving constructive advice for my research and career path.

Thirdly, I would like to thank my friends, colleagues and students (especially those from RMIT International University Vietnam) for their advice, support and encouragement throughout my PhD research journey. A special thanks to Prof Julian Teicher, Mr Chris Munro and Ms Kim-Tram Nguyen from RMIT International University Vietnam.

Fourthly, to my parents who supported me financially and mentally for the past years. Without their training and encouragement, I can never be a guy who is tough enough to strive for higher things.

Finally, my deepest appreciation goes to my beloved wife Nicole. She always keeps her strong belief in me especially at the time when I have overwhelming doubt about my ability. Thanks for holding my hands tight when I feel confused, frustrated, hopeless and helpless. It is impossible to make it if she did not accompany with me in this journey. I LOVE YOU……..

Unless the LORD builds the house, the builders labour in vain.

Unless the LORD watches over the city, the guards stand watch in vain. Psalm 129:1
Ontology, a branch of philosophy, was borrowed by artificial intelligence community and is defined as an explicit specification of a conceptualization where a conceptualization is an abstract, simplified view of the world we wish to represent for specific purposes (Gruber 1993a). By representing domain specific knowledge with vocabularies based on objects and their interrelated describable relationships in ontologies, inference engines and other application programs from intelligence systems are able to understand and reuse knowledge of another systems.

The concept of ontology can also be used to overcome the incapability of Hypertext Markup Language (HTML) to represent web content semantically. The application of ontologies in the context of the Semantic Web allows web contents to be represented in a structural form using a finite list of vocabularies and their relationships. In this way, computer programs and software agents are able to understand the semantics of web content, thus making it possible to process web contents. Ontology has been applied to various areas such as knowledge management (KM), business-to-consumer and business-to-business e-commerce, cloud computing as well as enterprise application integration (Abecker and van Elst 2009; Babkin and Potapova 2009; Chen and Pooley 2009; Fensel 2004; Jagdev et al. 2008; Silva et al. 2009; Weinhardt et al. 2009).

As a result, organizations have begun to adopt ontology for their information systems (IS) development. One common approach is to adopt one or a few globally shared ontologies for each domain. However, the globally shared ontology approach may hinder a system from reflecting its actual business requirements, thus compromising the system’s heterogeneity and flexibility. This is because the system is forced to design using taxonomies and terminologies defined in the globally shared ontology (Leung et al. 2009). Researchers believe that communities, organizations, departments or even individuals will begin to develop their own small-scale, domain specific ontologies as it is a non-trivial task to define and maintain large globally shared ontologies (Berners-Lee et al. 2001; Blois et al. 2007; de Bruijn et al. 2006;
Problem arises when multiple ontologies are used for a particular domain because each ontology possesses its own set of vocabularies and relationships. Thus, inconsistency of vocabularies and their relationships defined in multiple ontologies has resulted in failure for systems to understand and reuse other ontologies unless the ontologies are reconciled in some forms to allow system interoperability. The above inconsistent problems are known as ontology mismatches and reconciliation of these mismatches is termed as ontology mediation. Ontology matching is one of the most important phases in ontology mediation which aims to discover similarities and establish semantic relationships between two ontologies (Predoiu et al. 2006).

This research will propose an ontology KM framework with the aim to reuse inter-organizational knowledge to create and disseminate knowledge in an organizational KM process. This research aims to investigate a theoretical ontology mediation methodology through the use of ontology integration to reconcile multiple ontologies. In addition, it also aims to develop a theoretical ontology mediation classification to provide guidelines to identify appropriate ontology matching techniques and designing new mediation tools.

This chapter provides an introduction and overview to the thesis. The chapter is organized as follows. Section 1 presents an introduction to the thesis. Section 2 describes the research problems. The research aim, objective and approach are presented in Section 3, 4, and 5 respectively. Section 6 presents research contribution and Section 7 presents organization of the thesis.

1.1 Research Problems

Knowledge is recognized as one of the most important and valuable management assets because knowledge enables organizations to utilize and develop resources, enhance their competitiveness and develop sustainable competitive advantage (Chan and Chao 2008; Sharkie 2003). KM aims to manage and capitalize on knowledge by
organizing formal and direct process to manage organizational knowledge that accumulates in the workplace (King 2009; Leung 2006; Martenson 2000; Turban and Aronson 2001). Nonaka et al. (2001) suggest that socialization, externalization, combination and internalization can be used to create organizational knowledge by means of interaction between explicit and tacit knowledge.

A number of KM approaches have been developed with the purpose of managing organizational knowledge, for example, the re-distributed KM framework is developed to manage organizational help desk knowledge (Leung 2006). Other examples include the integrative framework that establishes an effective knowledge transfer process within an organization (Goh 2002), the distributed KM framework that allows individual knowledge workers and distributed communities to manage organizational knowledge with the support of ontology (Pirro et al. 2010) and the distribution, interaction, competition and evolution (DICE) model that examines organizational knowledge from an ecological perspective (Chen at al. 2008).

Unfortunately, these designs only focus on managing intra-organizational knowledge which is inadequate in the current business environment because users are often required to access inter-organizational knowledge from other knowledge sources in order to complete the tasks in current knowledge explosion era. In this research, inter-organizational knowledge is referred to a set of explicit knowledge that is formalized and created by other organizations. For instance, an IT security expert retrieves up-to-date security alerts from external knowledge sources such as knowledge bases from Norton and Symantec to safeguard the systems from possible threats.

Furthermore, these KM approaches also fail to collaborate with each other due to the fact that their designs are based on their own business and KM requirement in managing organizational knowledge. The individual business and KM requirement of individual KM approach have limited its capability of managing inter-organizational knowledge. These non-collaborative KM approaches create independent “knowledge islands” which may result in several disadvantages for knowledge workers and knowledge engineers. The knowledge workers need to spend additional effort to search for relevant knowledge from various knowledge bases; the
knowledge engineers have to spend a lot of resources to create and update organizational knowledge which is also available in other knowledge management systems (KMS). This available formalized inter-organizational knowledge is reusable in a way that it can be retrieved by any organizations to support their individual KM processes in terms of knowledge creation and dissemination. The heterogeneity of KMS that were designed and developed based on individual business and KM requirements is an obstacle to reusing inter-organizational knowledge. Thus, heterogeneous KMS must be interoperable before inter-organizational knowledge reuse can occur.

Knowledge reusability and mismatches reconcilability of ontology and its related mediation methods explored in the research aims to enable the interoperability of heterogeneous KMS. Literatures show that there are three major kinds of ontology mediation that can be used to reconcile ontology mismatches that include mapping, merging and integration. Ontology mapping is a process of relating similar concepts and relations from different ontologies to each other in which the correspondences between different entities of the two ontologies are formulated as axioms in a specific mapping language (de Bruijn et al. 2006; Flouris et al. 2008; Klein 2001). Ontology merging, unlike mapping that links two separate ontologies together in a consistent and coherent form, creates a new ontology by unifying two or more different ontologies on that subject and it is usually difficult to identify regions of the source ontologies from the merged ontologies (Linaza et al. 2009; Pinto and Martins 2000; Taboada et al. 2005). Pinto and Martins (2000; 2001a; 2001b) define ontology integration as a process of building an ontology in one subject through reusing one or more ontologies in different subject domains and it is always possible to identify regions of the source ontologies from the integrated ontologies.

Furthermore, the concept of ontology also addresses the research gap to answer researcher questions on how to allow inter-organizational knowledge to be reused in an organizational KM process, thus reforming the existing KM approaches that only focus on managing organizational knowledge.
1.2 Research Aims

This research aims:

1. To investigate a theoretical ontology mediation framework.

2. To examine an ontology-based KM approach in the domain of reusing inter-organizational knowledge.

1.3 Research Objectives

The objectives of this research are as follows:

1. To develop a theoretical ontology mediation classification to provide guidelines to develop ontology mediation for ontology mediation framework.

2. To investigate a theoretical ontology mediation framework.

3. To develop an ontology-based KM approach to reconcile inter-organizational knowledge.

1.4 Research Method and Approach

The research has been conducted as follows:

1. A literature review is conducted to investigate the nature of knowledge as well as processes to create, store, disseminate, use and evaluate knowledge and technologies adopted in the KM process. Theoretical background on ontology as well as its mismatches and mediation methods are also examined. Through the extensive literature review, a classification of ontology matching techniques is proposed to provide guidelines to identify appropriate ontology matching techniques and designing new mediation tools.
2. A theoretical ontology mediation framework on ontology development methodology is developed and a semi-automatic tool is designed to support the framework.

3. An ontology-based KM framework is investigated to provide mechanisms of reusing inter-organizational knowledge to support individual KM process. This framework is developed using design science research methodology. Design science research methodology designs and develops an artifact such as potential constructs, models and methods to provide a solution for a research problem (Hevner and Chatterjee 2010). The artifact is illustrated in experimentation, simulation, case study, proof or scenario to observe and measure how well the artifact supports the research problem. In this research, the ontology-based KM framework incorporating the theoretical ontology mediation framework is developed to provide solutions to address the problems created by non-collaborative KM approach. The ontology-based KM framework is illustrated in an IT help desk application scenario to demonstrate: 1) the collaboration of various KMS, and 2) the reuse of inter-organizational knowledge to support knowledge creation and dissemination in organizational KM process.

4. Proof of concept approach is adopted to illustrate the application of the proposed frameworks in an IT help desk application scenario that includes the proposed ontology mediation framework and ontology-based KM framework. A prototype is also developed to demonstrate the proposed frameworks.

1.5 Research Contribution

The contributions of this research are:

- Formalising a classification framework of ontology matching techniques. The classification lays a foundation for the development of ontology matching techniques used in mediation tools.
• Proposing an integration-oriented ontology development methodology that reuses existing ontology as part of its ontology development process. The proposed ontology development methodology makes an original contribution in applying ontology integration in the development of a new ontology.

• Proposing an ontology-based KM framework that provides mechanisms of reusing inter-organizational knowledge to create and disseminate organizational knowledge. The ontology-based KM framework makes an original contribution to knowledge in the reuse of inter-organization knowledge to support knowledge creation and dissemination in an organizational KM process.

1.6 Thesis Organization

The rest of the thesis is organized as follows. Chapter 2 presents the literature review related to KM and ontology. The first section of Chapter 2 discusses the nature of knowledge and the five stages of KM process to create, store, disseminate, use and evaluate knowledge. In particular, a detailed discussion on how knowledge is created by means of interaction between explicit and tacit knowledge is presented. The second section of Chapter 2 provides background of ontology and an overview of ontology language. A detailed discussion on ontology mismatches and methods to deal with ontology mismatches is also presented.

Chapter 3 conducts a literature survey to examine ontology matching techniques used in some of the most significant mediation tools, frameworks and methods. In particular, Shvaiko and Euzenat’s (2005) classification of ontology matching techniques is analysed. The survey findings and analysis results are used to develop a new classification of ontology matching techniques that aims to provide guidelines for identifying appropriate ontology matching techniques and designing of new mediation tools.

Chapter 4 provides a detailed analysis of ontology integration and merging that includes guidelines for choosing appropriate ontology reuse methods. This chapter
also analyses ten significant ontology development methodologies. Limitations of the
ten significant ontology development methodologies are addressed by proposing an
ontology development methodology that uses ontology integration to reuse existing
ontologies in a development process. The proposed ontology development
methodology consists of six phases which include preparation, analysis, integration,
development, implementation and evaluation, and maintenance. A tool that can assist
to perform integration semi-automatically is also provided.

Chapter 5 presents an ontology-based inter-organizational KM network to provide a
platform to allow KMS of various organizations to collaborate. The proposed KM
network incorporates ontology and its mediation methods to enable organizations to
access, retrieve and reuse inter-organizational knowledge of a similar domain to
support knowledge creation and dissemination in individual organizational KM
process. A framework that provides guidelines for choosing appropriate mediation
approaches in the establishment of the network is also presented in this chapter.

Chapter 6 provides a proof of concept for the proposed methodology and framework
presented in chapters 3, 4 and 5. An application scenario in an IT help desk
environment is developed to illustrate how explicit inter-organizational knowledge
can be created using the proposed ontology development methodology presented on
Chapter 4. The application scenario is also used to illustrate how explicit inter-
organizational knowledge can be disseminated in individual organizational KM
process in the proposed ontology-based inter-organizational KM network presented
in Chapter 5. In addition, a prototype is developed to illustrate the dissemination of
inter-organizational knowledge in the proposed network. The prototype is also used
to illustrate how a semi-automatic tool can be used to support the proposed ontology
development methodology.

Chapter 7 concludes the thesis. A summary of research results, contributions and
future research directions are presented in this chapter.
Chapter 2: LITERATURE REVIEW

This chapter provides theoretical background related to this thesis. Literatures in relation to KM and ontology are presented. As knowledge is considered as one of the most important asset in organization, a detailed discussion regarding the nature of knowledge, process to create, store, disseminate, use and evaluate knowledge and technologies adopted in the KM process are provided in this chapter.

This chapter also provides background of ontology. Ontology is an explicit specification of a conceptualization and can be used to represent knowledge using representational vocabulary to allow systems to understand semantics from each other. Besides, an overview of language used to develop ontology is also given. These include Extensible Markup Language (XML), XML Schema, Resource Description Framework (RDF), RDF Schema and Web Ontology Language (OWL). As organizations and individuals are expected to develop their own ontologies rather than using a common ontology, this chapter discusses mismatches caused by multiple ontologies. Methods used to deal with ontology mismatches including ontology mapping, merging and integration are also discussed here.

This chapter is organized as follows. Section 1 provides an overview of knowledge and KM. This section also discusses how IT is used in KM. Section 2 provides background of ontology which includes the discussion of ontology mismatches, web ontology languages and methods of ontology mediation. Section 3 concludes the chapter.

2.1 Knowledge Management

Back in mid 1980s, management tools and techniques such as total quality management, downsizing and business process reengineering had been developed by western companies to aid in re-gaining market share in automotive and electronic appliance industries which were dominated by the Japanese companies (Chase 1997). However, both input and improvement were short-term. The solution approaches
were generic and easily replicated by rivals (Sharkie 2003). Once an approach was proven successful, the rival company duplicates and adopts the same practice. The practices of downsizing, outsourcing and business process reengineering had resulted in the loss of many experienced employees with their expertise and knowledge (Coulson-Thomas 1997). The practices would further led to the loss of organization’s priceless inspiration and creativity as well as to the failure of securing long term competitive advantage (Chase 1997). Companies are currently using the concept of KM to sustain long term competitive advantage by preserving organizational knowledge (Turban and Aronson 2001). Knowledge is now recognized as one of the most important management assets because knowledge enables organizations to utilize and develop organizational resources, enhance competitive ability and develop sustainable competitive advantage (Neumann and Tome 2011; Plessis 2007; Sharkie 2003; Wu and Lee 2007). In summary, knowledge allows an organization to do better than its rivals.

Before continuing the discussion of KM, it is essential to clarify the meaning of knowledge. Knowledge is not an uncommon word. In a study conducted by Chase (1997), 92% of respondents claimed that they worked in knowledge-intensive organizations, but many people still confuse about differences among data, information and knowledge (Chase 1997). Holsapple (2005) points out that there are three common perspectives to describe data, information and knowledge: identification, inclusive and exclusive. The identification perspective perceives knowledge and information as two identical terms which can be used interchangeably. Advocators of the exclusive perspective argue that knowledge can no longer be considered as knowledge if it has been processed by any form of computer-based technology because the term “knowledge” is reserved for the domain of human or social processing. In this case, computerised knowledge can only be considered as data or information. Alternatively, the inclusive perspective views knowledge as any representations that may be symbolic, visual, audio, mental, digital and behaviour. These representations are usable for processors when knowledge embodied in the representations gives the processors the capacity to take action. According to degrees of usability, data, information, structured information, evaluation, judgement and decision are one of the several states in the progression of knowledge (van Lohuizen 1986).
In this thesis, we consider data, information and knowledge as three different entities but are interrelated with each other. Data are raw facts, whereas information is data that has been refined, processed and organised to support decision (Rob and Coronel 2002; Whitten et al. 2001). Most data is in the form of numeric, basic information or observations of work activities that can be quantified while information is data with relevance, purpose as well as context (Smith 2001). On one hand, information has little value until human intervention is applied to extract its meaning or use on a job. On the other hand, knowledge appears in the forms of facts, attitudes, opinions, issues, values, theories, reasons, processes, tools, relationships, risks and probabilities. Knowledge is often considered as information that contains specific properties (Coulson-Thomas 1997; Lueg 2001). Leonard and Sensiper (1998) further identify knowledge as information that is relevant, actionable and based at least partially on experience. Nonaka et al. (2001) further describe knowledge as justified true belief that is rational, dynamic, humanistic and context-specific; information would become knowledge only if personal interpretation of experiences, beliefs and commitments are added. While Lueg (2001) views information as a kind of preliminary stage to knowledge, Dawson (2000) argues that knowledge and information are linked together through the processes of internalization of information into personal knowledge and externalization of personal knowledge into information. In addition, Polanyi (1962) and Krogh et al. (2000) divide knowledge into tacit and explicit. Tacit knowledge (or know-how) that gains through individual insights overtime, is personal, complex and hard to communicate as well as formalize because it resides in human, mind and body in the focus of beliefs, assumptions, behaviours, perceptions, actions, procedures, routines, commitments, ideals, values and emotions (Goh 2002; Martensson 2000; Nonaka and Takeuchi 1995; Nonaka et al. 2001). Conversely, explicit knowledge (or know-what) is structured and relatively simple. It can be captured, recorded, documented, codified and shared using formal and systematic language in the forms of manuals, patents, reports, documents, assessments, databases, scientific formulas and other information technology (IT) media.

KM seeks to manage and capitalize on knowledge that accumulates in the workplace using appropriate means and technologies (Abdullah et al. 2008; Kant and Singh
This is achieved by organizing formal, systematic and direct process to create, store, retain, evaluate, enhance and increase organizational knowledge for future benefit of the organization (King 2009; Leung 2006; Martensoon 2000; Turban and Aronson 2001). KM also aims to enhance the quality, content, value and transferability of individual and group knowledge within an organization (Mentzas et al. 2001). Therefore, KM is capable of sustaining long term competitive advantage. Sharkie (2003) indicates rival company can easily duplicate and imitate the process of KM or even its technology, but it will be very difficult to copy the knowledge and skills of employees. The spirit of KM encourages organizations to create and use knowledge continuously and also to enable them to take initiative in innovating and enhancing products, services and operations.

There are variations among researchers in describing processes of KM. For example, Wiig (1997) divides the process into knowledge building, transforming, organizing, deploying and using, whereas Chait (1999) depicts that the KM process is based on capturing, evaluating, cleansing, storing, providing and using of knowledge. In this research, KM process is divided into five stages: create, store, disseminate, use and evaluate knowledge as illustrated in Figure 2.1.
Nonaka et al. (2001) suggest that there are four methods to create organizational knowledge by means of interaction between explicit and tacit knowledge. The first method is socialization. It is the process of developing new tacit knowledge from tacit knowledge embedded in human or organization through experience sharing, observation and traditional apprenticeship. The second method is called externalization. This is the process of changing tacit knowledge into new explicit knowledge simply by transforming tacit knowledge in the form of document such as manual and report. The third method is internalization. This is the process of embodying explicit knowledge as tacit knowledge by learning, absorbing and integrating explicit knowledge into individual’s tacit knowledge base. The last one is called combination. This is the process of merging and editing “explicit knowledge from multiple sources” into a new set of more comprehensive and systematic explicit knowledge. Researchers such as Alavi and Leidner (2001), Harmaakorpi and Melkas (2005) and Nonaka and Konno (1998) state that IS/IT can be used to facilitate the combination method. For instance, intranet can be adopted to collect organizational project plans, business minutes, research reports, action models and best practises, then edit and combine into new forms. However, the level of complexity increases if required knowledge is located out of the organizational boundary. The complexity increases because the required knowledge is managed using different KM approaches designed according to business requirements of individual organization. The heterogeneous KM approaches must be interoperated in some form before the knowledge can be adopted in the combination method.

Store and disseminate of knowledge are often linked with technologies. Explicit knowledge created is collected and stored in database or knowledge base in which users can access knowledge using “search and retrieve” tools through intranets (Abdullah et al. 2008; Alavi and Leidner 1999; Chen and Xu 2011; Prusak 1999; Smith 2001). The retrieved knowledge can then be used by knowledge workers to add value to current business processes, implement and coordinate organizational strategy, predict trends in the uncertain future, deliver new market values, create new knowledge, solve existing problems and so on (Bailey and Clarke 2001; Metaxiotis 2006; Newman 1997; Richtner and Ahlstrom 2010). The fifth stage of KM is knowledge evaluation. This phrase eliminates incorrect or out-dated knowledge (Alavi and Leidner 1999). Organization must continue creating new knowledge to
replace any knowledge that has become invalid or obsolete (Dawson 2000; Leung 2011).

There are a number of KM approaches designed to manage organizational knowledge. Examples include the re-distributed KM framework that manages organizational help desk knowledge (Leung 2006), the distributed KM framework that enables knowledge workers and distributed communities to manage organizational knowledge using ontology (Pirro et al. 2010), the KM framework designed for identifying, organizing, analysing and translating nursing knowledge into daily practice in a clinical organization (Anderson and Willson 2009), the integrative framework that provides an effective knowledge transfer process in an organization (Goh 2002) and the DICE model that examines organizational knowledge from an ecological perspective (Chen et al. 2008). These approaches are designed to manage knowledge within an organization without the concern of system interoperation. Thus, the approaches, in particular KMS, are developed based on organizational requirements rather than adopting a common language or standard. The absence of a common language or standardization has created a barrier to prevent collaboration of different KMS (Sheth 1999). Although the emergence of middleware technology has provided a way to support syntactic and structural interoperability in IS, it cannot be used to enhance semantic interoperability in KMS due to the fact that the middleware is not designed to understand the semantics of knowledge request from users and knowledge sources. Besides, the concept of middleware can hardly be accommodated in the era of the Internet as each pair of KMS are required to implement a customised middleware for interoperation (Leung et al. 2007). Since a single KMS is interconnected with various systems via the Internet, it is impractical to customize and install middleware for each connection. Another deficiency of middleware is that even if the involved systems only undergo a minor modification, the middleware may require a complete re-construction when different KMS are accessed.

The success of KM largely depends on human and social factors such as human relationship, trust, obligation, reputation, motivation, mentality, creativity, interpretation, strategy, support resources, organizational structure, reward and benefit structure, leadership, culture and so forth (Bosch-Sijtsema et al. 2009;
Holsapple 2005; Sulaiman and Burke 2011; Thomas et al. 2001). In contrast, technology is only regarded as a tool to streamline the KM process and technology itself adds no value to knowledge (Ray 2008; Smith 2001). For instance, electronic repository and search engine can be used to facilitate knowledge storing and dissemination but these technologies are meaningless if knowledge workers show no willingness to share or use knowledge. Goh (2002) highlights that cooperation and collaboration among groups, individuals and leaders in knowledge transfer and sharing can add value to knowledge. Level of trust, time availability, leaders’ participation and commitment, environment setting, organizational structure and monetary as well as non-monetary rewards are keys to motivate knowledge transfer and sharing (Coulson-Thomas 1997; Goh 2002; Linder and Wald 2010; Wu and Lee 2007).

A number of researchers view KM as another repackaging project of IT and even confuse KMS with IS because their concepts are alike (Lueg 2001). On one hand, IS utilize data to yield useful information to support and improve business operations, problem-solving and decision-making within an organization. In general, special algorithm or technique is required to combine small chunks of data in order to transfer them into information. On the other hand, KMS is used to store and disseminate knowledge in the KM process (Alavi and Leidner 1999, Whitten et al. 2001). To achieve this, knowledge is required to convert into a particular type of representation that can be stored in the KMS where users can retrieve the stored knowledge using dissemination function embedded in the KMS. The well-developed IT and IS provide many of the foundations for the development of specific KM tools (Jurisica et al. 2004). For example, modern database technology enables enormous amount of explicit knowledge, originally in the form of text, audio or video, to be accommodated within electronic repository in various digital representations. Another example is data communication technology such as the Internet, groupware product, electronic messaging infrastructure and mobile devices that allow knowledge to be disseminated and KM to be performed ubiquitously.

Although advanced search technology allows information to be identified, retrieved and prioritized according to relevancy to search query, current search technique fails to deliver high precision results in response to users’ queries. For example, searching
the web with the keyword “bank” using Google search engine will return any webpages that contain “bank” or with “bank” as one of the indexes, regardless whether “bank” means a financial institution or river bank on the webpages. This conventional keyword search will affect the knowledge dissemination and use if it is applied to KM. To solve this problem, KM researchers have developed ontological search approach to overcome limitation of conventional keyword search. Documents are annotated using machine-processable metadata extracted from a domain specific ontology (Davies et al. 2005; Mentzas et al. 2001; Stabb et al. 2001). Exploitation of ontological metadata allows a more precise collection of knowledge to be retrieved. For example, to look for banking knowledge, user is required to search for “bank” as a concept of “financial institution”, then the search engine will find the relevant pieces of knowledge by examining ontological metadata of each piece of knowledge (Haase et al. 2005). Another way of adopting the ontological metadata in knowledge retrieval is by ontology browsing. For example, in an IT help desk KMS developed by Leung (2006), ontology that contains classification of technical problems and their related symptoms have been constructed.

2.2 Ontology

Over the past two decades, a lot of methods had been used to integrate heterogeneous IS. Heterogeneity of IS is due to various systems were designed and developed based on different business requirements. Thus, interoperation becomes necessary to enable systems of different characteristics to communicate, cooperate and exchange information as well as reuse web services. In the era of the World Wide Web (WWW), a business transaction involves different components and requires different data and information from different sources. For instance, when a customer is shopping in an online store, he may wish to read comments on the quality of a particular product from a forum. When he decides to purchase the product, the online store needs to contact related financial institutions for payment verification and confirmation. The online store is also required to arrange delivery service with shipping company. Such a simple e-commerce transaction has involved interoperation of at least three heterogeneous IS. The complexity can be compounded if it involves participation from more enterprises and companies.
Sheth (1999) classifies four different types of heterogeneity that can occur during system interoperability process: system, syntactic, structural and semantic. The emergence of distributed computing, middleware technology and standardization has provided ways to support three types of heterogeneity (Cui et al. 2002). System heterogeneity refers to hardware, operating systems and communication incompatibilities whereas syntactic and structural heterogeneities refer to different representation languages, data representation and data modeling formats (Aparicio et al. 2005; O’Sullivan and Lewis 2003). As the amount and diversity of information are increased speedily and tremendously through the WWW, there have been more concerns on whether IS can help users at the knowledge level. Therefore it is essential to shift the focus from system, syntactic and structural heterogeneity to semantic heterogeneity.

Semantic heterogeneity within multiple database schemas is well discussed in database community. It refers to the use of different names, data types, values, constraints and decomposition for the same structure during data exchange (Kashyap and Sheth 1996; Kim and Seo 1991; Krishnamurthy et al. 1991). Apart from referring semantic heterogeneity as the fact that data present in different systems may be subjected to different interpretations, Goh (1997) also makes a major classification on its causes. First, naming conflicts are used to describe the application of different names to represent the same concept. For example, “Grade Point Average” can also be represented as “GPA” or “G.P.A.”. Second, scaling and unit conflicts occur when different units of measurements or scales are used to measure the same value. For example, weight can be measured in units of kilogram, gram, pound or ounce. Third, confounding conflicts arise when confounding of concepts are in actual fact distinct. For example, the “latest exchange rate” reported by two data sources may not be the same as one may have a greater temporal delay than the other.

One way to solve the above semantic conflicts is to write code which translates terminologies between pairs of systems (Cui et al. 2002). However, this is a very inefficient and costly development method. This is particularly true in an environment such as the WWW since each pair of systems involved are required to
develop an individual translation program. This static method also fails to support dynamic nature of business environment. Any changes within the IS require substantial efforts and resources to recode the translation program. To achieve semantic interoperation, Sheth (1999) suggests the approach should be capable of supporting high level and context-sensitive information exchanges or requests over heterogeneous information sources in a semantically consistent manner. Researchers such as Noy (2004) and Wache et al. (2001) assert that ontology can be exploited to describe the semantics of information sources and make the content explicit. Ontology also allows information sources to discover semantic equivalence between information concepts which provides the mechanism in solving various conflicts as described above.

2.2.1 Background of Ontology

Ontology, a branch of philosophy, is the science of what is, of the kinds and structures of objects, properties, events, processes and relations in every area of reality (Smith 2003). Ontology can also be defined as a particular system of categories accounting for a certain vision of the world (Guarino 1998). Artificial intelligence (AI) researchers first applied the concept of ontology in intelligent system development so that knowledge can be shared and reused among various AI systems. Tom Gruber’s definition has been widely accepted within the AI community: an ontology is an explicit specification of a conceptualization while a conceptualization is an abstract, simplified view of the world that we wish to represent for some purpose (Gruber 1993a). Borst (1997) then refines Gruber’s definition by labelling an ontology as a formal specification of a shared conceptualization. Based on Gruber’s and Borst’s definitions, Studer et al. (1998) make the following conclusions: 1) an ontology is a machine-readable specification of a conceptualization in which the type of concepts used and the constraints on their use are explicitly defined, and 2) an ontology should only capture consensual knowledge accepted by large group of people rather than just some individuals. By representing knowledge with representational vocabulary in terms of objects and their interrelated describable relationships, inference engines and other application programs from one intelligent system will be able to understand the semantic of
knowledge in another intelligent system.

Heterogeneous AI systems are able to interoperate with each other in terms of communication as well as knowledge reuse and sharing by committing themselves to ontological commitments. The intended meaning of vocabularies for a particular conceptualization are constrained by ontological commitment to guarantee their consistency. Thus, the ontological commitment is an agreement of using shared vocabularies and their relationships in a coherent and consistent manner (Gruber 1993a). To make it clearer, ontological commitment is a function that links terms of ontology vocabulary with a conceptualization to allow the systems to communicate a set of vocabularies and their relationships without using a globally shared ontology (Gomez-Perez et al. 2004; Guarino 1998). Gruber (1993b) further argues that ontological commitment should be defined at the knowledge level (independent of the symbol-level representation specified for agents) and a common ontology can serve as a knowledge-level specification of the ontological commitments of a set of participating agents. The involved systems can still preserve their own heterogeneity without sacrificing the interoperability. This is because sharing of a common ontology does not mean that they have to share their internal encoding of knowledge. Application of ontology is no longer limited to AI systems. Other application areas of ontology include enterprise integration, standardization of product knowledge, medicine, mechanical engineering, electronic commerce, geographic, legal and biological IS (Guarino 1998).

The popularity of the WWW further magnifies the importance of ontology. The WWW is originally designed to allow people to retrieve and browse information on static web pages. Web technology is capable of accommodating a wide variety of flexible, dynamic and interactive activities that ranges from simple applications, multimedia web pages, sophisticated business systems, to complex software applications. However, the HTML-based web content is solely designed for formatting and displaying information on the web and computers have no way of understanding and processing the semantics of these web contents (Antoniou and Harmelen 2004). The disadvantage of HTML-based web content can be reflected when users attempt to retrieve information from the web using a search engine. For example, it is not uncommon for a search query to return more than ten thousand
results. Using search operators in advanced web search engine may narrow down the results to a few hundreds results, but users still require extensive effort to locate the required information within the pool of output returned. It is due to the fact that the search engine can only perform keyword search without understanding the actual semantic of the document. Whenever document contains keywords specified by user, it will be included in the search result. To overcome weaknesses in the current WWW, the Semantic Web is proposed. The Semantic Web is the extension of the current web, in which web content is represented in a structural form within ontologies (Berners-Lee et al. 2001). In this way, ontologies enable computer programs and software agents to understand the semantics, thus making it possible for programs and agents to understand and process web content. Ontologies also provide a shared understanding of a domain which are necessary to overcome differences in terminology from various sources (Antoniou and Harmelen 2004).

There are various different interpretations on how ontologies can be categorised. One common way is to categorise ontologies based on functionalities. For example, the classification of ontologies proposed by Mizoguchi et al. (1995) in which ontologies are divided into four different types: 1) content ontologies for knowledge reuse, 2) communication ontologies for knowledge sharing, 3) indexing ontologies for case retrieval, and 4) meta- or knowledge representation ontologies for knowledge formalization. Guarino (1997) uses two dimensions to describe the classification. Firstly, ontologies can be distinguished based on the level of detail. For example, a very simple ontology (such as a thesaurus) to be shared among users which already agree on the underlying conceptualization. Secondly, ontologies can be distinguished as top-level, domain, task and application based on their level of dependence on a particular task or point of view. The level of dependence is shown in Figure 2.2 in which the top-level ontology is on the first level, the domain and task ontologies are on the second level and the application ontology is on the third level.
Top-level ontology which is independent of a particular domain can be shared by an enormous user base. The top-level ontology can be used to describe very general concept such as time, space, object, event and action. These general concepts can further be specified in next level of domain and task ontologies. Domain ontology specifies vocabularies of a domain using specialized terms obtained from the top-level ontology. This ontology provides vocabularies about concepts and their relationships within a domain, activities that take place in that domain, and theories and elementary principles governing that domain (Perez and Benjamins 1999). Task ontology is used to describe vocabularies related to a task using specialized terms obtained from the top-level ontology. Furthermore, vocabularies defined in the domain and task ontologies can further be extended to more specialized vocabularies in the application ontology. The application ontology should contain all of the required knowledge definitions required to model a particular application (van Heijst et al. 1997).
The scope of ontologies is very broad, ranging from simple notion of a taxonomy, a thesaurus, a conceptual model to logical theory (Daconta et al. 2003). Lassila and McGuinness (2001) use a linear spectrum to classify ontologies into nine types: catalogue, glossary, thesaurus, informal “is a”, formal “is a”, formal “instance”, frames, value restrictions and general logical constraints. The linear spectrum of ontology classification is illustrated in Figure 2.3 in which catalogue is the simplest type of ontology that contains a finite list of controlled vocabularies. Glossary provides more details than catalogue because glossary also defines meanings of vocabularies using natural languages. As compared with catalogue, thesaurus adds an extra dimension of semantics by including information (such as synonym) to describe relationships between vocabularies. Moving to the right of thesaurus in the spectrum are informal and formal “is a” that provide explicit hierarchy of class and relationship. However, informal “is a” only provides general notion of generalization and specialization, whereas formal “is a” includes strict subclass relationships so that class and subclass inheritance can be achieved. Followed by “is a” is formal instance which incorporates formal instance relationships. The next one in the spectrum is frames that provide additional property information used to describe relationship between class and subclass. Similar to class and subclass inheritance, the property information defined at a general class level would as well be inherited by its subclasses and instances. This is followed by the value restrictions in which range restrictions can be added to restrict the properties using value. Finally, the most expressive in the spectrum is general logical constraints that allow first order logic expressions to be used in the ontology.

Ontology can also be categorised as lightweight and heavyweight ontology (Gomez-
Perez et al. 2004). The former includes taxonomies that include concepts, concept taxonomies, relationships (between concepts) and properties (which describe concepts), whereas the latter adds axioms and constraints to the lightweight ontology. Thus heavyweight ontology is able to provide more semantics and apply more restrictions than lightweight ontology. While lightweight ontology can be developed using less expressive ontology language, heavyweight ontology is required to use a more expressive language to construct the axioms and constraints. An example of less expressive ontology language is XML and an example of more expressive one is OWL. A description of ontology language will be provided in the following section.

2.2.2 Web Ontology Languages

The World Wide Web Consortium (W3C) is responsible to develop, standardize and recommend new generation of ontology language. XML was developed by an XML Working Group formed in 1996 with the purpose of setting new standards for defining and exchanging data structures (Bray et al. 2004; Fensel 2004). Rather than label it as a set of web ontology language, it is more appropriate to describe XML as a syntax for other ontology languages to build on (Hunter 2004). Unlike HTML, tags in XML are not pre-defined and no longer used to depict the display format for web content. Instead, the tags in XML enable users to define their own vocabularies. The opening and closing tags in XML are used to label element that contains several descriptive attribute-value pairs. Tagged elements can also be nested to allow elements to be structured in a tree-like hierarchy.

XML Schema uses XML as syntax to define and describe a class of XML documents by constraining and documenting meanings, usage and relationships of datatypes, elements and their contents, and attributes and their values (Thompson et al. 2011). It is important to note that XML Schema is used to specify syntactic and structure conventions for exchanging XML documents, therefore it is not designed to provide a semantic description of the domain (Volz 2004). Although Document Type Definition (DTD) provides similar functionalities, the newer approach, XML Schema, has several advantages over the DTD. XML Schema not only provides a rich set of built-in datatypes but it also allows users to define their own datatypes.
Another advantage of XML Schema is its reusability in which user only requires to extend or restrict the existing one. Besides, the support of namespace mechanism enables an XML document to contain vocabularies from multiple XML Schemas without worrying about vocabulary collision and recognition problem. XML namespace is another W3C standard that uniquely identified a collection of vocabularies by Uniform Resource Identifier (URI) (Bray et al. 2009).

According to a research conducted by Lee and Goodwin (2005), 98% of the Semantic Web pages are written in RDF which makes RDF the most popular ontology language. Recommended by W3C as a lightweight ontology, RDF uses XML-based syntax to provide a common framework to represent information and knowledge on the web that can be identified, exchanged and processed by applications without losing the inherent semantic (Manola and Miller 2004). To express semantic, RDF exploits an object-attribute-value statement that contains resource, property and value (Yao and Etzkorn 2004). A resource can be any object or RDF statements whereas a property is a special resource that used to describe relationship between resources and its value. Consider the following statement “http://www.uow.edu.au/~knl164 is created by Nelson”. This statement shows that “http://www.uow.edu.au/~knl164” is a resource, “is created by” is a property and “Nelson” is a value. In a RDF statement, value can be represented by either a literal or resource. “Nelson” is considered as a resource if it can be identified by an URI, otherwise “Nelson” is just a literal.

RDF Schema is a semantic extension of RDF which provides mechanisms to describe groups of related resources and the relationships (properties) between these resources in a RDF document (Brickley and Guha 2004). To define the semantics of a resource, RDF Schema utilizes superclass, class and subclass concepts which are very similar to the concept used in object-oriented programming. In general, a class contains a set of resources. By establishing relationships between classes, a domain-specific hierarchy is formed. The resulting hierarchy is able to restrict the interpretation of the resources to their intended semantics in a RDF document. This hierarchy concept can also be applied to properties by organizing them into super-properties, properties and sub-properties within a RDF Schema. To ensure consistency of semantic interpretation, RDF Schema allows property to define its
domain and range specifications.

RDF and RDF Schema are only capable of representing partial semantics because the set of features possessed by RDF and RDF Schema is inadequate to describe classes, properties and the relations between them within the hierarchy. Thus, the emergence of OWL is designed to provide a richer language for defining structured, web based ontology (Miller 2004). OWL facilitates greater machine interpretability of web content than XML, RDF and RDF Schema by providing additional vocabularies to describe properties and classes. These include 1) relations between classes such as disjointness, 2) cardinality, 3) equality, 4) richer types of properties, 5) characteristics of properties such as symmetry, and 6) enumerated classes (McGuinness and Van Harmelen 2004). To fulfil different requirements on semantic expressiveness and reasoning efficiency, OWL is divided to three subsets: OWL Full, OWL Description Logic (DL) and OWL Lite. OWL Full is fully compatible with RDF and RDF Schema and it guarantees maximum semantic expressiveness in sacrifice for reasoning efficiency. While OWL DL provides a balance on both semantic expressiveness and computation capability, OWL Lite sacrifices expressiveness in exchange for reasoning support. Due to potential extendibility of expressiveness in RDF and RDF Schema, OWL DL and OWL Lite are only partly compatible with RDF and RDF Schema in order to retain their computation ability.

OWL 2 which is an extension and revision of OWL is designed to address the insufficiency of the previous version. This includes expressivity limitations in cardinality, relational and datatype as well as problems of annotation, syntax, import and versioning, and metamodel compatibility (Grau et al. 2008). To address the insufficiency, OWL 2 inherits OWL’s language features, design decisions and use cases but also adds several new features in the language (Golbreich and Wallace 2009). To increased expressive power for properties, OWL 2 adds new constructs for new characteristics of properties, property disjoints, property chains, keys and expressing additional restrictions on properties. Besides, a richer set of datatypes and restrictions are included in OWL 2 to improve expressiveness of datatype. Other than that, the syntax problems are minimized by adopting shorthand in the language to allow some common patterns easier to code. Other features added to improve the shortcomings include simple metamodelling capabilities and extended annotation
2.2.3 Ontology Mismatches and Mediation

It is impractical to expect all individuals and organizations to agree on using one or a small subset of ontologies (de Bruijn et al. 2006; Mascardi et al. 2010; Noy 2004). The adoption of such an approach is problematic. It is a non-trivial task to define and maintain a large globally shared ontology. A globally shared ontology approach may hinder a system from reflecting its actual business requirements due to the fact that the design of the system is restricted by terminologies defined in the ontology (Hasson et al. 2008; Horrocks 2008; Leung et al. 2009; Seidenberg and Rector 2006; Torres et al. 2008). Instead of using a large globally shared ontology, there will be a large number of small domain specific ontologies developed by various communities, organizations, departments or even individuals (Berners-Lee et al. 2001; Blois et al. 2007; de Bruijn et al. 2006; Ding and Fensel 2001; Hendler 2001). However, data heterogeneity caused by multiple ontologies can become an obstacle for systems interoperability (Visser and Cui 1998) because vocabularies and their relationships defined in the ontologies can be inconsistent, resulting in difficulty for one system to understand and to reuse another ontology when multiple ontologies are used. Thus, ontologies need to be reconciled in some forms. The above inconsistent problem caused by multiple ontologies is commonly termed as ontology mismatches.

Visser et al. (1997) categorise ontology mismatches into explication and conceptualization mismatches. There are six different types of explication mismatches in accordance with definiendum, definiens and the explicated ontological concept. Visser et al. (1997) has developed a guideline to deal with six different levels of difficulties caused by the six explication mismatches. The conceptualization mismatches can be classified into five different types:

1. Categorization mismatch occurs when two or more ontologies of a domain distinguish the same class but divide this class into different subclasses. For example, two conceptualizations both contain knowledge “computer”. Mismatch occurs when one ontology defines “laptop” and “desktop” as its
subclasses and the other includes “notebook” and “workstation” as the subclasses.

2. Aggregation-level mismatch occurs when two or more ontologies of a domain recognize the existence of a class but define the class at different levels of abstraction. For instance, one conceptualization defines “computer” while the other defines “laptop” and “desktop” without having “computer” as their superclass.

3. Structure mismatch occurs when two or more ontologies of a domain use different relations to structure the same set of classes. For example, one conceptualization uses relation “contain” to relate “computer” and its subclass “CPU” whereas the other ontology uses relation “has-component” to relate these two classes.

4. Attribute-assignment mismatch occurs when two or more ontologies of a domain differ in the way they assign an attribute to other classes. For instance, two conceptualizations both define classes “computer” and “laptop” as instances. Mismatches occur when one conceptualization assigns attribute “processor speed” to “computer” and the other assigns “CPU speed” to “laptop”.

5. Attribute-type mismatch occurs when two or more ontologies of a domain use different measurement units to instantiate the same class. For example, two conceptualizations both define the class “processor speed”. Mismatch occurs when one conceptualization assigns the number of gigahertz as its instances and the other assigns the number of megahertz as its instances.

Another significant categorization on ontology mismatches is defined by Klein (2001). Klein (2001) points out that ontology mismatches can appear at language level when two or more ontologies written in different languages are combined. There are four types of language level mismatches: 1) syntax mismatch which happens because different ontology languages often use incompatible syntax; 2) logical representation mismatch in which ontologies of different languages use
different representation of logical notions to stand for the same concept; 3) semantics of primitives mismatch which can occur due to different languages having different semantic interpretations; 4) language expressivity mismatch occurs when one language fails to express something that is expressible in another language.

Syntax, logical representation and semantics of primitives mismatch are relatively easy to resolve. Various researchers have proposed several mechanisms to solve ontology mismatches at the language level. For instance, the superimposed metamodel (Anicic et al. 2007; Bowers and Delcambre 2000) uses a metamodel to uniformly represent model, schema and instance data from diverse model-based applications. Explicit representation allows data to transform from one representation to another using inherent mapping rules specified in the metamodel. Layered approach is another example which facilitates data interoperation at the language layer. This approach proposes to divide a complex data model into syntax, object and semantic layers in which each layer contains a number of sublayers (Cruz and Xiao 2009; Melnik and Decker 2000). As a result, ontology mapping can be performed by establishing bridges and gateways between the less complex layers or sublayers of different data models. Open Knowledge Base Connectivity (OKBC) is also capable of solving some of the language level mismatch problems (Chaudhri et al. 1998). The application of the OKBC requires assumptions about representation of a specific ontology language to be made explicitly. These explicit assumptions make it possible to define a mapping from OKBC to that ontology language. Users are able to access the knowledge representation system through OKBC because they are bounded logically.

Language expressivity mismatch is considered the most complex to resolve among the categorization of language level mismatches because partial semantics may disappear during the translation process (de Bruijn et al. 2004; Enrech 2005). The adoption of XML, XML Schema, RDF, RDF Schema and OWL may not be possible to eliminate the effect caused by language level mismatches but such an action may help to define and standardize solution on the mismatches. For example, to setup a standard translation mechanism between OWL and DAML+OIL to resolve expressivity mismatches caused by the two languages.
Ontology level mismatches may occur when two or more ontologies programmed in the same or different languages with overlapping domain are combined. Klein (2001) divides mismatches into conceptualization and explication. There are two types of conceptualization mismatches: 1) scope mismatch which refers to classes that have the same concept but not the instances, 2) model coverage and granularity mismatch which occurs when only part of the domain is covered by the ontology or when the level of detail used to model the domain is different. For example, one ontology may define “computers” into “desktop”, “laptop” and “tablet”, whereas another ontology may classify computers in a very detailed manner based on specifications of processor, RAM, hard drive, wireless network interface and operation system.

Klein (2001) describes the explication mismatches using paradigm, concept description, terminological and encoding mismatches. Paradigm and concept description mismatches are closely related to the developer’s modelling style. Paradigm mismatch occurs when ontology developers choose to use different paradigms to represent concepts. For example, one ontology may use temporal representations based on interval logic while another may use a representation based on time points (Chalupsky 2000).

Concept mismatch occurs when different modelling conventions is used to represent the same concepts. For example, one class can be modelled using qualified attributes and the other is modelled by introducing a separate class. For instance, one ontology developers may choose to use “type” as an attribute of class “computer” to distinguish various types of computers such as desktop, laptop and tablet, whereas another ontology developer may choose subclasses for class “computer” using “desktop”, “laptop” and “tablet”.

Terminological mismatch occurs when the same concepts are represented using synonym terms (for example couch and sofa are two interchangeable words) or when a homonym term is used to represent two different contexts (for example bank can be used to describe river bank or financial institute). Encoding mismatch occurs when values are encoded in different formats. For instance, one ontology may use kilogram as its unit of measurement for weight whereas the other may use pound.
Based on actual business or organizational requirements, organizations and individuals are expected to develop individual ontologies of different languages, scopes, coverage and granularities, modelling styles, terminologies, concepts and encodings. To reuse ontologies of different types, ontology mediation is required to reconcile mismatches between heterogeneous ontologies so that data sharing and reuse among multiple data sources as well as communication between heterogeneous applications can be achieved (Predoiu et al. 2006; Scharffe and de Bruijn 2005). Figure 2.4 shows that there are three major kinds of ontology mediations: mapping, merging and integration.

Figure 2.4 Classification of Ontology Mediation

Spaniol and Klamma (2004) explain ontology mediation can be performed manually, semi-automatically or automatically. Manual mediation requires involvement of ontology engineer and domain expert. The ontology engineer who has in-depth knowledge on ontology languages and formalization aspects, is responsible to analyse, design and conduct the mediation process. The domain expert assists ontology engineer by providing interpretation on the semantic of concepts in a particular domain. In contrast, semi-automatic mediation process requires the support of automatic tools. For instance, predefined rules and algorithms together with the embedded techniques (such as machine learning and heuristic) within the tool can be used to identify similarities among ontologies. The tool will then be able to make recommendations on how to perform mediation based on computational results derived from similarities and differences of the ontologies such as Chimaera and PROMPT (McGuinness et al. 2000a; McGuinness et al. 2000b; Noy and Musen 2000). Hence, the final decision on choosing the most suitable mediation method will
be left to human users. Other forms of support provided by semi-automatic tools include post-mediation verification, validation, critiquation as well as conflicts recognition and resolution (Leung et al. 2009).

Automatic mediation relies on fully automated tools that can resolve conflicts between different ontologies without human intervention. Compare to manual mediation, automatic mediation is relatively cheap, less time consuming and less tedious especially when this approach is applied to ontologies of larger size (Doan et al. 2002; Jung 2010; Mao et al. 2010). Although it will be ideal to perform mediation automatically, most researchers agree that human efforts are still required to justify the mediation results generated by automated tools since fully automated tools are unable to detect and interpret concepts that do not have close correlations (Cui et al. 2002; Liu et al. 2007; Noy 2009). It may also fail to handle any unforeseeable situations since it is designed to perform mediation under specific condition (Uschold 2000).

2.2.3.1 Ontology Mapping

Ontology mapping identifies semantic overlap in terms of concepts or relations between two ontologies in which the correspondence is formulated in a specific mapping language (Abels et al. 2005; Predoiu et al. 2006). Some researchers (Calvanese et al. 2001; Kalfoglou and Schorlemmer 2002; Madhavan et al. 2002) argue that mapping should include mapping of concepts, relationships, ontological axioms and logical sentences. However, most researches focus on the first two techniques. The resultant mapping can be used to perform query rewrite and answer, instance transformation and web-service composition (Kiryakov et al. 2004; Noy 2004). Since the involved ontologies do not require any adaptation, ontology mapping often specifies partial overlap between ontologies which is relevant for mapping application (Scharffe et al. 2006). There are two common approaches used to establish mapping between ontologies, namely top-level and one-to-one mapping approaches.
Top-level mapping approach relates all ontologies to a shared ontology so that different local ontologies are mapped together indirectly through the shared ontology as illustrated in Figure 2.5 (Choi et al. 2006; Predoiu et al. 2006; Wache et al. 2001). Hence, it is relatively easy to resolve conflicts and ambiguities since concepts used in different ontologies are based on primitives of the top ontology. Thus, shared ontology provides a reference point to analyse, harmonize and compare existing local ontologies (Mika et al. 2004). There are two different types of shared ontologies in which the line of distinction between them depends on the existence of local ontologies (Noy 2004). Foundational and reference ontologies are the first type and they are usually created before the existence of inherited local ontologies. As it describes very general formalizing notions such as processes and events, time, space and physical objects, it is exploited to provide the grounding in common and domain independent vocabulary for “yet to be developed” inherited local ontologies. The second type is integrated ontology which generally is developed after the local ontologies are built. To build an integrated ontology, it requires common views of individual local ontologies to be combined. As integrated ontology only aims to provide access to all local ontologies that it integrates, the set of vocabularies defined in this ontology is usually domain- or industry-specific.

Researchers such as Cali et al. (2002), Cruz et al. (2004), Levy (2000) and Li and Chang (2000) point out that global-as-view and local-as-view are two apposite methods to design mappings with respect to top-level ontology. The global-as-view method provides specific mappings between integrated ontology and local ontologies. These mappings enable every element of top-level ontology to associate
with a view over local ontologies so that the meaning of each element in the global ontology is further defined at the local ontologies. On the contrary, local-as-view is more suitable for specifying mappings between common ontology and local ontologies. This method requires top-level ontology to be defined independently from local ontologies. In this way, the contents of local ontologies are detailed in terms of a view over the top-level ontology. When comparing the two top-level ontology mapping methods, it is relatively more complex to maintain a top-level ontology in the local-as-view approach than in the global-as-view (Ullman 1997). The former method may require the top-level ontology to undergo a major reconstruction, but the latter does not necessarily involve changes in the top-level ontology when there are changes in the local ontologies.

There are many shared ontologies that have been conducted based on the top-level ontology mapping approach. For instance, the Process Specification Language (PSL) is developed by the National Institute for Standards and Technology as a mediating ontology to formalize and structure manufacturing terms and concepts along with their definitions (Gruninger 2009). The PSL can then be used to facilitate correct and complete exchange of process information among application ontologies of heterogeneous manufacturing systems. ABC Model is another example that adopts the idea of top-level ontology (Jeong and Kim 2010; Lagoze et al. 2000). Individual communities are encouraged to create and maintain their own metadata modules in accordance with functional and community needs. However, many entities and relationships in between are so general that do not belong to a particular module but can be applied to all. Therefore, the ABC Model is developed with the aim to include all of the common entities and relationships. The resultant model allows individual communities to extend these common semantics for developing individual domain- and application-specific metadata modules.

However, top-level mapping approach has three major drawbacks. Firstly, constructing a large-scale top-level ontology from scratch is never a simple task, even if we take a simpler path by merging or integrating various local or reference ontologies. The experiences of building the Air Campaign Planning Ontology (Valente et al. 1999) and the Suggested Upper Merged Ontology (Niles and Pease 2001) have shown that the actual merging processes are trickier than expected.
Reasons include inconsistency between chunks of theoretical content as well as structural differences between local ontologies. Secondly, this approach can only be adopted in a relatively stable environment where maintenance is minimal because a substantial amount of resources and overheads are required to maintain the top-level ontology. Thirdly, mappings established between local ontologies and top-level ontology can easily be affected by the elimination and addition of local ontologies as well as changes in either local or top-level ontologies because local ontologies are related indirectly with each other through the top-level ontology.

![Figure 2.6 One-to-one Mapping Approach](image)

The second approach is one-to-one mapping approach. This approach requires mappings to be created between each pair of ontologies as shown in Figure 2.6 (Predoiu et al. 2006). Mena et al. (2000) adopt the one-to-one mapping approach to develop OBSERVER which is capable of browsing and querying information scattered across multiple heterogeneous repositories. The lack of a top-level ontology in this approach makes it possible to be used in a highly dynamic environment. This advantage may be offset by the lack of common terminologies which increases the complexity of defining mapping between local ontologies. Another major drawback of this approach occurs when a large number of heterogeneous ontologies are involved in the interoperation. Such an interoperation will greatly increase the amount of mappings and extra efforts are required to control and maintain the mappings. Furthermore, one-to-one approach is also less scalable to interoperate compared to top-level ontology approach.

Predoiu et al. (2006) develop a generic mapping process which contains three
different phases as shown in Figure 2.7. In the first phase (*import ontologies*), ontologies are required to convert to a common format if they have been specified in different languages. For example, iPROMPT and AnchorPrompt require ontologies of different formats to be converted to RDF Schema (Noy and Musen 2003), whereas in Information-Flow-Based Method for Ontology Mapping (IF-Map), ontologies are required to partially translate into Prolog (Kalfoglou and Schorlemmer 2002). This format translation can be performed using either a self-developed translator or a public translator. In the second phase (*finding similarities*), ontology matching is performed to discover similarities between two ontologies. In the third phase (*specifying mapping*), user needs to select and execute an appropriate set of mapping operations, either manually or aided by a tool, so that the mapping between the ontologies can be specified.

![Figure 2.7 Generic Mapping Process (Predoiu et al. 2006)](image)

Another significant mapping process is described in the horizontal dimension of the Ontology Mapping Framework (MAFRA). MAFRA is designed to support interactive, incremental and dynamic ontology mapping process so that instance transformation can be performed in the Semantic Web (Maedche et al. 2002). The five modules of MAFRA, namely lift and normalization, similarity, semantic bridging, execution and post-processing, are illustrated in Figure 2.8. The first module (*lift and normalization*) requires two ontologies of different formats to be normalized to a uniform representation. This module applies natural language processing techniques which include tokenization, elimination and acronyms expansion to cope with syntactic heterogeneity. To make semantic differences more apparent between the two ontologies, natural language processing techniques such as tokenization of entities, elimination of stop words and expansion of acronyms, are
performed in this module too. In the second module (similarity), MAFRA adopts lexical, property, bottom-up and top-down similarities algorithms to calculate similarities between ontology entities.

![Diagram](image)

Figure 2.8 Mapping Process of MAFRA (Maedche et al. 2002, p. 237)

In the third module (semantic bridging), semantic bridges that specify mapping axioms are created to establish correspondences between similarities of the ontologies calculated in the previous module. Semantic bridges can be represented in ontology mapping languages that are identical to the inherent ontology languages. For example, ontology language ODL is used to specify ontology and its mappings in Mediator environment for Multiple Information Sources (MOMIS) approach (Bergamaschi et al. 1999). Mapping languages can also be independent from any ontology languages. For example, mapping language developed by the Ontology Management Working Group, Semantic Bridge Ontology (SBO) and RDF Transformation (RDFT) (Maedche et al. 2002; Omelayenko 2002; Scharffe and de Bruijn 2005). de Bruijn (2003) argues that mapping language should be able to perform value transformation as well as define equivalence, subclass, superclass, partial, union and disjointness relationships. In many mapping tools, there is a feedback loop iterated from this phrase to the previous one which can provide more
accurate similarity measures when part of the mapping has been specified by the user. In the fourth module (*execution*), operations are executed to transform instances from one ontology to another via the semantic bridges defined in Module 3. Finally, the fifth module (*post-proceeding*) is to check and improve quality of the transformation results.

### 2.2.3.2 Ontology Merging

The second type of ontology mediation is merging. Unlike mapping that links two separate ontologies together in a consistent and coherent form, ontology merging creates a new ontology by unifying two or more different ontologies and it is usually difficult to identify regions of the source ontologies from the merged ontologies (Noy and Musen 1999; Pinto and Martins 2001a; Pinto and Martins 2001b). Compare to ontology mapping that keeps the original ontologies unchanged, merging requires at least one of the original ontologies to be adapted so that conceptualization and vocabularies can be matched in the overlapping sections of the ontologies (Ding et al. 2002). In general, the source ontologies will disappear or become unavailable after merging. This is because all correspondences and differences between the source ontologies are reflected in the merged ontology, thus making the merged ontology as the only ontology available for adoption (de Bruijn et al. 2006; Predoiu et al. 2006). In some cases, the source ontologies remain and the merged ontology takes up the role of top-level ontology with mappings established between them.

While a majority of the Semantic Web researchers foresee mainstream ontology developer would adopt the approach of developing enormous amount of small domain specific ontologies, McGuinness et al. (2000a) argue that some industries and organizations still need to develop very large and standardized ontology. For instance, SNOMED CT is a comprehensive clinical ontology developed by the College of American Pathologists that contains about 344,549 distinct concepts and 913,697 descriptions (Lussier and Li 2004) and YAGO is another large ontology (used in numerous major ontology projects) consists of 1.7 million entities and 15 million facts derived from Wikipedia and WordNet (Suchanek et al. 2008). Universal Standard Products and Services Classification (UNSPSC) is the third example of
large ontology that provides a common coding scheme for classification of products and services between buyers and sellers (Granada Research 1998). Theoretically, it is more efficient and effective to merge existing ontologies than to build a large ontology from scratch. The involved ontologies are often developed by different people for different purposes with different assumptions and using different vocabularies (Lambrix et al. 2003; Pinto and Martins 2001a; Pinto and Martins 2001b). Thus, the process of ontology merging is more than just simple revisions, improvements or variations of the source ontologies in practice.

To meet the growing needs of ontology merging, the Stanford University Knowledge Systems Laboratory developed a semi-automatic merging tool called Chimaera. The purpose of Chimaera is to support two major tasks of merging: 1) coalesce two semantically identical terms from different ontologies so that they can be referred to by the same name in the resulting ontology, 2) identify terms that should be related by subsumption, disjointness or instance relationships (McGuinness et al. 2000b). The first major task corresponds to the second phase (finding similarities) of the generic mapping process illustrated in Figure 2.7. Chimaera suggests potential merge candidates by generating name resolution list and taxonomy resolution list using structural and linguistic similarity measurement. The first list includes terms of different ontologies that are candidates to be merged or to have taxonomic relationships not yet included in the merged ontology, whereas the second list includes taxonomy areas that are candidates for reorganization. Other than ontology merging, Chimaera also provides diagnostic support which incorporates incompleteness checking, syntactic analysis, taxonomic analysis and semantic evaluation.

Noy and Musen (2003) developed a semi-automatic tool, iPROMPT, to guide user in the process of merging two ontologies. iPROMPT is a component of the PROMPT Framework that contains a suite of tools for managing multiple ontologies. Similar to Chimaera, iPROMPT supports merging first by generating an initial list of potential candidates and merge operations based on linguistic similarity measurement among concept names. The second step requires user to choose either the suggested operations from the list or to specify the desired operation directly. As a result, iPROMPT executes the requested operation and automatically performs additional
changes originated from the operation. It then generates a new list of suggested operations based on the new ontology structure, determines inconsistencies and potential problems introduced by the previous operation and proposes possible solutions for the identified problems.

Based on the two merging tools, the process of ontology merging should consist of four phases as shown in Figure 2.9. The first phase (import ontologies) requires two ontologies to be converted to a common format if they have been developed in different languages. The second phase (finding similarities) requires ontology matching to discover similarities between two ontologies. The third phase (specifying merging) requires user to select and execute an appropriate set of merging operations, either manually or aided by a tool.

Other than tools, researchers have also developed methods for merging ontologies. For instance, Stumme and Maedche (2001) propose FCA-MERGE as a method to merge ontologies semi-automatically. Unlike Chimaera and iPROMPT that rely on syntactic and semantic matching heuristics, FCA-MERGE uses mathematical techniques of Formal Concept Analysis (Wille 1982; Granter and Wille 1999) to produce a lattice of concepts based on a shared set of documents relevant to the source ontologies. The lattice is derived from natural language processing techniques instances to extract instances from the shared documents. While the previous tasks are performed automatically, the last step requires background knowledge of the domain expert to analyse and transform the lattice into the merged ontology.
2.2.3.3 Ontology Integration

Pinto and Martins (2000, 2001a, 2001b, 2004) define ontology integration as a process of building an ontology in one subject by reusing one or more ontologies in different subjects. In this approach, it is always possible to identify regions of the source ontologies from the integrated ontologies. Source ontologies may need some forms of refinements before they can be aggregated, combined and assembled together to form the resultant ontology. It is also important to include ontology integration in the early stage of the ontology building process (preferable during conceptualization and formalization stage) in order to simplify the overall ontology building procedure. To perform ontology integration, Pinto and Martins (2001a, 2001b) develop a methodology that comprises the following ten activities:

1. Identify whether it is possible to build ontology using integration.
2. Identify modules that should be included in the future ontology.
3. Identify assumptions and ontological commitments that each module should comply to.
4. Identify knowledge to be represented in each module.
5. Perform a general analysis to eliminate candidate ontologies that are of no use.
6. Get candidate ontologies in an appropriate form preferable at the knowledge level.
7. Evaluate and access candidate ontologies by domain experts and ontologists using specialized sets of criteria.
8. Choose an ontology or a set of source ontologies.
9. Apply integration operations.
10. Analyse resulting ontology.
Figure 2.10 shows the ten activities. The first four activities can be performed simultaneously while the remaining activities should be executed sequentially. However, exceptions can occur. For example, if the candidate ontologies have been developed in an appropriate language, then the activity to “get candidate ontologies in an appropriate form” can be ignored as language translation process does not need to take place. Another example is if a candidate ontology is found to fulfil all of the requirements of the resultant ontology, the activity to “apply integration operations” and “analyse resulting ontology” can be eliminated. A summary of ontology mapping, merging and integration is given in Table 2.1
Table 2.1 Summary of Ontology Mapping, Merging and Integration

<table>
<thead>
<tr>
<th>Ontology mediation</th>
<th>Mapping</th>
<th>Merging</th>
<th>Integration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of source ontology required</td>
<td>At least two source ontologies</td>
<td>At least two source ontologies</td>
<td>At least one source ontology</td>
</tr>
<tr>
<td>Process</td>
<td>Three phases: 1) Import ontologies 2) Finding similarities 3) Specifying mapping operations</td>
<td>Three phases: 1) Import ontologies 2) Finding similarities 3) Specifying mapping operations</td>
<td>Ten Activities: 1) Identify possibility 2) Identify modules 3) Identify assumptions/ontological commitments 4) Identified knowledge to be represented 5) Eliminate candidate ontologies 6) Convert candidate ontologies in an appropriate form 7) Evaluate and access candidate ontologies 8) Choose source ontologies 9) Apply integration operations 10) Analyse resulting ontology</td>
</tr>
<tr>
<td>Mediation results</td>
<td>Correspondence is formulated between identical terms from two source ontologies</td>
<td>A merged ontology is created</td>
<td>An integrated ontology is created</td>
</tr>
<tr>
<td>Identification of Source Ontology(ies)</td>
<td>Able to identify source ontologies as they are not combined with each other</td>
<td>Difficult to identify source ontologies from the merged ontology</td>
<td>Able to identify source ontologies from the integrated ontologies</td>
</tr>
</tbody>
</table>

2.3 Conclusion

This chapter has shown that knowledge is one of the most important management assets to organizations. KM enables organizations to formalize the process to create, store, make available, use and evaluate organizational knowledge. Managing knowledge effectively enables organizations to sustain long term competitive advantage because knowledge can be used to innovate and enhance products, services and operations.

However, the design of many KM approaches is inadequate to meet the current business environment because these approaches are developed to manage organizational knowledge based on requirements of individual organization without
the concern of managing inter-organizational knowledge from external source. In the current dynamic business environment, employees of an organization often require accessing and retrieving both organizational and inter-organizational knowledge to support, improve and complete tasks and activities. Besides, the heterogeneous nature of organizational KMS developed for current KM approaches also fails to enable collaboration to occur with each other even though the required knowledge is available within inter-organizational KMS.

As external source of knowledge is essential for organizational performance, a new inter-organizational KM practice is required to enhance the interoperability among independent KMS. This enhancement enables organizations to reuse inter-organizational knowledge rather than spending time and resources to create additional KMS. This enhancement also encourages the sharing of knowledge across organizational boundaries in business networks because knowledge-sharing in an inter-organizational network allows a richer and more diverse body of knowledge to be created (Wagner and Buko 2005). An approach towards this direction of allowing organizations to manage inter-organizational knowledge in their KM process is therefore desirable.

The literature review in this chapter shows that the concept and theoretical framework of ontology can be used to represent knowledge formally and explicitly. Although ontologies developed by different organizations can cause ontology mismatches, ontology mediation provides a way to resolve ontology mismatches caused by the heterogeneous ontologies. The application of ontology and its mediation methods enable systems of the organization to understand and reuse the inherent semantic of other organizations, thus making it possible for organizational systems to interoperate and collaborate with each other.

By adopting ontology to represent organizational knowledge in each knowledge source and ontology mediation methods to resolve ontology mismatches caused by organizational ontologies, organizations can access and retrieve inter-organizational knowledge. An ontology-based approach is therefore a feasible approach to reconcile inter-organizational knowledge from heterogeneous KMS. The ontology approach can be investigated to manage inter-organizational knowledge in the organizational
KM process.
Ontology matching is one of the most important phases in the process of ontology mapping and merging. The main purpose is to establish semantic relationships between two ontologies. In general, ontology matching can be defined as the process of discovering similarities between two ontologies (Predoiu et al. 2006). It determines relationship between two sets of entities that belong to two discrete ontologies (Shvaiko 2004). The process involves finding a corresponding entity in the second ontology for each entity in the first ontology that has the same or the closest intended meaning. The matching process can be achieved by using one or multiple ontology matching techniques to analyse similarity of the entities (Ehrig and Sure 2004; INTEROP 2004). The correspondence can either be expressed as one-to-one function or one-to-many function. One-to-one function denotes that an entity in an ontology can have only one similar entity in another ontology whereas one to many function addresses the fact that an entity may have more than one similar entities in another ontology (Castano et al. 2007).

Figure 3.1 Ontology Matching Process (Ehrig and Staab 2004)

In this research, we will use the mapping process developed by Ehrig and Staab (2004) to describe the five essential tasks of performing ontology matching as shown in Figure 3.1. To make the matching process more understandable, the above five tasks were categorised into three stages, namely pre-matching, matching and post-matching. In the pre-matching stage, some preparation works are required before the actual similarity computation can take place. The pre-matching stage starts with feature engineering in which initial representations of two ontologies are transformed into a common format suitable for similarity computation. In some cases, syntactic
normalization is involved in the feature engineering task; that is, natural language processing techniques such as tokenization, lemmatization and elimination are used to normalize syntactic heterogeneity (Maedche et al. 2002; Giunchiglia et al. 2004). The second task in the pre-matching stage is to determine what the next search step is in order to find the matching candidate. The most common approach is to compare all entities of the first ontology with all entities of the second ontology (Noy and Musen 2000; McGuinness et al. 2000a; Ehrig and Sure 2004). Other advanced approach allows the matching tool to find similarities from a subset of candidate concept pairs and to ignore others (Rahm et al. 2004).

The second stage is matching. This is where the actual similarity computation is conducted to determine similarity values between matching candidates. Many researches have focused on developing mediation tools that adopt multiple matching techniques as it is unlikely for a single technique to achieve as many good matching candidates as the multiple techniques do (Rahm and Bernstein 2001). There are two ways to combine matching techniques, either by integrating multiple matching criteria in a hybrid matcher or by combining the results of independently executed matchers within a composite matcher (Di Martino 2006). As there may be more than one similarity values for a candidate concept pair, the post-matching stage requires the matching tool to aggregate different similarity values into a single aggregated value for one candidate pair. The final task of the post-matching stage requires the matching tool to determine a suitable cut-off point to interpret the similarity value to derive the best matching pair(s) among concepts in the first ontology and a set of concepts in the second ontology. These five tasks of ontology matching are iterated until no new similarities can be matched.

Some state of the art mediation tools combine various types of techniques, either in parallel or sequentially, to increase their versatility. For instance, IF-Map employs two techniques that include string-based and model-based (Kalfoglou and Schorlemmer 2003); QOM (Quick Ontology Mapping) uses a combination of three strategies which consist of string-based, linguistic resources and taxonomy-based technique (Ehrig and Staab 2004) and S-Match exploits string-based, language-based, linguistic resources and model-based as its matching techniques (Giunchiglia et al. 2004). Although multiple strategies are expected to achieve higher accuracy in
ontology matching compare to the single technique (Tang et al. 2005b), experiment results have shown multiple techniques do not always perform better than the single technique (Muslea 2002). To outperform the single technique, it is very important to select the most appropriate strategy combination for specified ontologies. For example, iRiMOM employs string-based, language-based, linguistic resources, constraint-based and taxonomy-based technique in its matching strategy (Tang et al. 2005a). The multi-strategy detection algorithm enables iRiMOM to select the most promising techniques as the final technique combination in accordance with the nature of the ontologies. Evaluation results show that iRiMOM improves precision and recall by 8.9% and 9.3% respectively using multi-techniques detection algorithm (Tang et al. 2005a). Precision and recall are used extensively in measuring performance of ontology matching in which precision is the measurement of correctness and recall is the measurement of completeness.

Other than choosing a suitable combination of matching technique, Ehrig and Sure (2004) argue that the selection of an appropriate similarity aggregation algorithm and a proper cut-off point are also critical to the performance of the techniques. To support their argument, Naïve Ontology Mapping (NOM) is developed to combine seventeen different similarity measures to find the matching candidates between two or more ontologies. Besides, they have also conducted a series of evaluations to test the integrated strategies with different aggregation algorithms and cut-off points against the one that uses only one similarity measure. The overall results in the evaluations show that the only scenario the integrated techniques can outperform the single similarity measure is when sigmoid function and a constant are used as the aggregation algorithm and the cut-off point respectively. In other words, all other combinations of integrated measures, aggregation strategies and cut-off points have only small gains or are outperformed by the single similarity measure in terms of precision, recall and f-measure (a measure combining precision and recall).

Castano et al. (2007) and Ehrig (2007) present some of the common algorithms to aggregate similarity values and to determine cut-off point. The simplest way to aggregate similarity values is by adding results of every single matching technique for one candidate pair. This simple addition algorithm can be improved by assigning weights to different matching techniques in accordance with their importance either
manually or using machine learning. The adoption of sigmoid function as an aggregation algorithm provides a more sophisticated way to compute the aggregated value. As sigmoid function emphasizes high individual similarities and deemphasizes low individual similarities, similarity values can be adjusted so that high values are further increased whereas low values are decreased by performing functional computation on each of the techniques. Other possible aggregation algorithm includes the use of machine learning and neural networks.

The purpose of determining a cut-off point is to select the best matching pair(s) among a concept in the first ontology and a set of concepts in the second ontology. Thus, candidate pair with similarity value above the cut-off point indicates a match whereas candidate pair with similarity value below the cut-off point is discarded. To determine a cut-off point, the simplest way is to choose a constant value but it is difficult to determine what this value is. A cut-off point can also be determined using the Delta method in which the value is defined by taking the highest similarity of all and subtracting a fixed value from it (Ehrig and Sure 2004). Another similar method is n-percent. Instead of subtracting a fixed value, n-percent subtracts a fixed percentage from the highest similarity.

This chapter conducts a literature survey with the purpose of examining the ontology matching techniques applied to some of the most significant mediation algorithms, tools, and methods based on Shvaiko and Euzenat’s (2005) classification of ontology matching techniques. This chapter also analyses this classification. The analysis results are used to develop a new ontology matching technique that aims to provide guidelines for identifying the matching technique and designing new mediation approach.

This chapter is organized as follows. Section 1 reviews ontology matching techniques. This section also includes a discussion on Shvaiko and Euzenat’s (2005) classification of ontology matching techniques. In addition, this section provides a literature survey on eighteen significant ontology mediation approaches to demonstrate how matching techniques defined in Shvaiko and Euzenat’s (2005) classification are performed. Section 2 proposes a new classification of ontology matching techniques. Section 3 concludes the chapter.
3.1 A Review of Ontology Matching Techniques

In this section, we present a literature survey on some of the most significant mediation algorithms, tools and methods. The focus is to examine the inherent matching process, in particular the similarity computation task at the matching stage of the process based on Shvaiko and Euzenat’s (2005) classification of ontology matching techniques.

3.1.1 Shvaiko and Euzenat’s Classification of Ontology Matching Techniques

Ontology matching (or similarity computation) can be processed by exploiting a number of different techniques. Basically, these techniques describe various components of ontology used as inputs to determine correspondences in the matching process. For instance, schema-based technique takes different aspects of the concepts and relations (Noy and Musen 2003), instance-based technique takes the instances (Doan et al. 2004), element-based technique takes properties of the particular concept and relation (Noy and Musen 2003) and structure-based technique takes the structures (Giunchiglia and Shvaiko 2003) of the ontologies. To provide a common conceptual basis, researchers have started to identify different types of ontology matching techniques and propose classifications to distinguish them. For example, Abels et al. (2005) propose a classification that consists of nine matching techniques based on existing literature studies. Another example is the classification developed by Shvaiko and Euzenat (2005). Building on the foundation of classifying schema matching techniques of Rahm and Bernstein (2001), Shvaiko and Euzenat (2005) develop a meticulous classification to categorise elementary ontology and schema matching techniques. Their classification focuses on techniques that exploit ontology-level information excluding instance data. As illustrated in Figure 3.2, there are two synthetic classifications that can be viewed in top-down and bottom-up manners.
The top-down view is called “granularity and input interpretation layer” which is based on granularity of matching and then on how input information is interpreted. Element-level and structure-level strategies are used to describe the granularity layer. Element-level matching deals with analysing entities in isolation. On the contrary, structure-level matching focuses on analysing combinations of elements that appear together in a structure. Moreover syntactic, external and semantic are the three criteria used to subcategorise the element-level and structure-level techniques which provide a detailed depiction for the input interpretation layer. Here, syntactic is the technique that adopts some clearly stated algorithms to interpret the input in function of its sole structure. External technique interprets the input using external resources of domain and common knowledge. Semantic technique interprets the input and justifies their results by exploiting some formal semantics.
The bottom-up view is called “kind of input layer” and it is based on the kind of input requires in the matching process. The first level of this layer describes the kind of data that the algorithm works on. While terminological and structural data are found within the ontology descriptions, semantic data is derived from semantic interpretation of the ontology. In addition, the first level can be decomposed if necessary. Terminological data can be either string-based or linguistic resources object, whereas structural data can be derived from either the internal structure of entities or the relations of entities with other entities. “Granularity/input interpretation layer” and “kind of input layer” are further divided into one common layer called “basic techniques layer”. Ten different types of elementary matching techniques are identified in this layer:

1. String-based technique is used to match names and name descriptions of ontology entities in terms of a sequence of alphabet letters. The intuition behind this strategy is that the more similar the strings, the more likely the concepts are similar.

2. Language-based technique uses natural language processing techniques such as tokenization, lemmatization and elimination to exploit morphological properties of the input words. The technique is usually applied before string-based technique in order to improve results.

3. Constraint-based technique is used to match the definitions of properties in terms of their internal constraints such as datatypes and cardinality.

4. Linguistic resources technique utilizes common knowledge or domain specific thesauri such as WordNet (Miller 1995) to analyse linguistic relations in the word matching process.

5. Alignment reuse technique exploits the idea of reusing alignments of previously matched ontologies as many ontologies to be matched are similar to the already matched ontologies with the same application domain. This technique is very effective in large matching problem which involves large
ontologies consisting of huge number of entities.

6. Upper level formal ontologies technique uses external source of common knowledge in the form of ontology such as SUMO and DOLCE (Gangemi et al. 2003; Niles and Pease 2001) within the matching process.

7. Graph-based technique considers the input as labelled graphs containing terms and their inter-relationships. Basically, the similarity is obtained through the analysis of the positions of a pair of nodes (from two ontologies) on the graphs. The intuition behind this technique is that if two nodes are similar, their neighbouring nodes will also be similar.

8. Taxonomy-based technique considers the input as graphs but the technique concerns only with the specialization relation. The intuition behind this technique is that is-a links connect terms that are already similar (being a subset or superset of each other) and their neighbours will also be similar.

9. Repository of structures technique stores ontologies and their fragments together with pair-wise similarities between them. Dissimilar to alignment reuse technique, repository of structures technique does not include any alignment information. When new ontologies are to be matched, the stored similarities can first be checked to avoid the matching operation to be performed over the dissimilar fragments. The available similarities can help to identify fragments that are worth carrying out the matching in more details.

10. Model-based technique deals with input based on its semantic interpretation using well-grounded deductive methods such as propositional satisfiability and description logics.

### 3.1.2 Literature Survey on Ontology Mediation Approaches

By conducting a literature review on eighteen significant ontology mediation approaches, a detailed description can be provided to demonstrate how the ten
matching techniques defined in Shvaiko and Euzenat’s (2005) classification are performed. The mediation algorithms, tools and methods discussed in the literature review is summarised in Table 3.1
<table>
<thead>
<tr>
<th>Mediation Approach</th>
<th>Reference</th>
<th>Description</th>
</tr>
</thead>
</table>
| Chimaera           | McGuinness et al. 2000a  
McGuinness et al. 2000b | A merging tool that generates a list of merging candidates. |
| iPROMPT            | Noy and Musen 2000  
Noy and Musen 2003 | A merging tool that suggests matching candidates, identifies inconsistencies and problems, and provides solutions. |
| AnchorPrompt       | Noy and Musen 2001  
Noy and Musen 2003 | A matching algorithm that finds semantically similar terms automatically. |
| MAFRA              | Maedche et al. 2002 | A mapping framework that provides a generic view onto the overall mapping process. |
| NOM                | Ehrig 2007  
Ehrig and Sure 2004 | A mapping tool that integrates 17 manually coded matching rules to identify possible mappings. |
| QOM                | Ehrig 2007  
Ehrig and Staab 2004  
Ehrig and Sure 2004 | A mapping tool that integrates 17 manually coded matching rules to identify possible mappings. Its focus is to improve quality and efficiency of mapping results. |
| ITTalks            | Prasad et al. 2002 | A semi-automatic mapping mechanism that produces a set of possible mappings using heuristics and probabilistic reasoning approach. |
| GLUE               | Doan et al. 2002 | A semi-automatic mapping tool that applies machine learning to perform taxonomy-based matching. |
| FCA-MERGE          | Stumme and Maedche 2001 | A merging method that uses language-based and linguistic resources techniques in the merging process. |
| IF-Map             | Kalfoglou and Schorlemmer 2003 | A mapping method that uses string-, taxonomy- and model-based techniques to derive partial mappings. |
| COMA               | Do and Rahm 2002 | A matching tool that provides hybrid matchers and also allows users to choose a combination of matching techniques. |
| COMA++             | Aumueller et al. 2005 | A matching tool that extends COMA to include a comprehensive graphical user interface and two additional matchers. |
| OMEN               | Mitra et al. 2005 | A matching enhancing tool designed to improve existing mappings using Bayesian Net. |
| OLA                | Euzenat et al. 2004 | A matching method that finds similarities between 2 OWL-Lite ontologies. |
| S-Match            | Giunchiglia and Shvaiko 2003  
Giunchiglia et al. 2004 | A matching tool that takes 2 graph-like taxonomies as input to compute the strongest semantic relations. |
| LILY               | Wang and Xu 2008 | A mapping tool that discovers mappings for normal and large scale ontologies. |
| CIDER              | Gracia and Mena 2008 | A schema-based matching tool that relies on schema-level input information to perform matching. |
| ASMOV              | Jean-Mary et al. 2009 | A matching algorithm that uses a comprehensive set of similarity measures and formal semantics technique to generate semantic consistent correspondences. |

Developed by the Stanford University Knowledge Systems Laboratory, Chimaera is
an ontology editing, merging and diagnosis tool with a browser-based user interface (McGuinness et al. 2000a; McGuinness et al. 2000b). To assist user in the merging tasks, Chimaera generates a list of merging candidates that contains names of classes and slots from different source ontologies. Exploiting simple heuristic approach, Chimaera performs string-based ontology matching to search for candidate sets of terms between two ontologies by comparing their term names, presentation names, term description, possible acronym and expanded forms.

PROMPT framework is a multiple-ontology management tool that provides support to ontology versioning, ontology merging, ontology matching and other related management tasks. As one of the integrated components in the framework, iPROMPT assists users in the ontology merging process by suggesting matching candidates, identifying inconsistencies and potential problems as well as suggesting possible solution to resolve them (Noy and Musen 2000; Noy and Musen 2003). Similar to Chimaera, iPROMPT uses simple heuristic approach to perform string-based ontology matching in the process of ontology merging. However, iPROMPT is only capable of matching classes with identical names from two different ontologies with no spelling deviation.

AnchorPrompt is another key component within the PROMPT framework that adopts heuristic approach to provide additional possible points of similarity between ontologies (Noy and Musen 2001; Noy and Musen 2003). AnchorPrompt views each ontology as a directed labelled graph with node (class) and edge (slot) to represent the taxonomy of the ontology. To provide suggestions on matching candidates, AnchorPrompt first needs to identify (either manually or automatically) a set of pairs of related terms (anchors) from the source ontologies as input. With a pair of anchors, it is possible to define a number of paths that consist of different sets of nodes and edges for each ontology. AnchorPrompt then traverses the paths in the corresponding ontologies and compare the nodes to find similar terms. As it traverses the two paths, it increases the similarity score for the pairs of terms in the same position on the paths. This process is repeated for each pair of paths to generate the final score by aggregating the similarity score from all the traversals. In this way, AnchorPrompt is able to produce a set of semantically related concepts from the source ontologies.
MAFRA is a mapping framework that provides a generic view onto the overall mapping process for distributed ontologies in the Semantic Web (Maedche et al. 2002). MAFRA uses a relatively more complex heuristic approach to perform ontology matching. On one hand, MAFRA makes use of two external linguistic resources, WordNet and altered Resnik Algorithm (Resnik 1999) to find lexical similarities between two ontologies in terms of concepts, attributes and relations. On the other hand, MAFRA exploits constraint-based techniques to acquire similarity between concepts based on their properties. Furthermore, taxonomy-based matching techniques are applied to the bottom-up and top-down strategies. The bottom-up strategy takes property similarity as input to propagate from lower parts of the ontology to the upper concepts. This strategy can provide a good overall view of similarity of attributes and relations between two ontologies. Simultaneously, the top-down strategy also allows MAFRA to propagate the similarity from top to bottom. This strategy enables MAFRA to assume special relevance when top level concepts have a higher or lower similarity. The similarity results computed by the above strategies are used to establish correspondence between entities of two ontologies in the form of semantic bridges.

NOM is an ontology mapping approach that integrates various similarity measuring methods to identify possible mappings (Ehrig 2007; Ehrig and Sure 2004). At present, seventeen manually encoded matching rules are used to measure similarities of two ontologies in the aspects of concepts, relations, instances and files. Other than the two matching rules designed for measuring similarities of hash-code and MIME-type files, the remaining fifteen matching rules are developed based on four types of matching techniques. First of all, string-based techniques are adopted to compare labels or URI of concepts, relations and instances between two different ontologies. For example, if labels are the same, the entities are likely to be the same. Second, external linguistic resources such as WordNet can further be used for comparisons across languages. Third, constraint-based techniques are applied to match properties between two ontologies. For example, if the domain and range of two properties are equal, the properties will also be equal. Fourth, taxonomy-based techniques are used to acquire similarity derived from the super(sub)-concepts and super(sub)-properties relationships. For example, if super-concepts are the same, the actual concepts will
be similar to each other. If sub-concepts are the same, the compared concepts will also be similar. If super-properties are the same, the actual properties are similar to each other, and if sub-properties are the same, then the compared properties are similar. In NOM, the similarity results of the seventeen rules are aggregated and interpreted using different combinations of aggregation algorithms and cut-off point determination strategies. The evaluation shows that the integrated measuring methods can achieve a better result in terms of precision, recall and f-measure when sigmoid function and a constant are chosen as the aggregation algorithm and the cut-off point respectively (Ehrig and Sure 2004).

QOM builds upon the success of NOM and is grounded on the seventeen matching rules of NOM (Ehrig 2007; Ehrig and Staab 2004; Ehrig and Sure 2004). Instead of focusing solely on the quality of mapping results, QOM is a mapping approach that considers both effectiveness and efficiency. To improve the mapping efficiency, QOM uses heuristic to discard the less promising mapping candidates in order to lower the number of candidates to be compared in the matching process. In addition, the use of some of the costly features in the rules has been restricted in order to optimize the matching efficiency. For example, it avoids the complete pair-wise comparison of trees in favour of a top-down strategy. Ehrig and Staab (2004) have demonstrated that QOM is on par with other state of the art algorithms (such as PROMPT, AnchorPrompt and NOM) in terms of matching quality. However, QOM outperforms other algorithms with respects to efficiency.

Prasad et al. (2002) combine text classification techniques, simple heuristic and probabilistic reasoning approach to develop a semi-automatic ontology mapping mechanism for Information Technology Talks (ITTalks). The mapping mechanism is intended to be used as a web-based system for automatic and intelligent notification of ITTalks. The first step in the mapping process requires each ontology to build a model that contains statistical information about the exemplar documents associated with each concept in the ontology using Rainbow classifier. Rainbow classifier is an application program that performs statistical text classification and can be used to support various classification methods (McCallum 1998). This classifier is then used again to compare exemplar of one ontology with statistical information of other ontology. The returned similarity scores together with a set of landmarks (user-
assigned mappings between two ontologies) are used to produce a set of possible mappings between the two ontologies. Here, Prasad et al. (2002) propose two graph-based algorithms to perform subsumption checking and to synthesis the similarity scores toward the final mappings.

The first algorithm is based on simple heuristic, which states that a concept from one ontology should match with a parent concept if it matches with the majority of children of the parent concept in another ontology. The second algorithm is based on Bayesian reasoning which is a probabilistic algorithm used to solve uncertainty in similarity comparisons. In this case, Bayesian reasoning is adopted to look for the best mapping concept that is lower in the hierarchy and with the posterior probability greater than 0.5. According to the evaluation conducted by Prasad et al. (2002), Bayesian approach outperforms the heuristic approach in terms of accuracy because the former approach has a stricter constraint.

GLUE is a semi-automatic system which applies machine learning to perform taxonomy-based ontology matching with the purpose of creating semantic mappings between two ontologies (Doan et al. 2002). The focus of the system is on finding correspondence between concepts of two given ontologies. The approach is, for each concept in one ontology, to find the most similar concept in the other ontology. To do so, GLUE first takes two taxonomies (for example $O_1$ and $O_2$) of two ontologies and their instances as input. Given that A and B are concepts of $O_1$ and $O_2$ respectively, machine learning is applied to allow a set of base learners to classify whether every instance of $O_1$ is also an instance of concept A and whether every instance of $O_2$ is also an instance of concept B. Meta-learner then combines the classifications from multiple base learners. The combined classifications are used to compute the joint probability distribution that consists of four probabilities: $P(A,B)$, $P(A,\neg B)$, $P(\neg A,B)$, and $P(\neg A,\neg B)$. Subsequently, GLUE employs a user-supplied similarity algorithm to compute a similarity value for each pair of concepts in the form of a matrix. Finally, GLUE applies relaxation labelling technique to search for an appropriate set of mapping configurations based on similarity values in the matrix, domain constraints and common knowledge. Evaluation results show that GLUE can accurately match 66% to 97% of the concepts on several real world domains (Doan et al. 2002). These domains include course category domain that describes courses at
Cornell University and University of Washington, and company profile derived from Yahoo.com and TheStandard.com that describes current status of companies.

FCA-MERGE is a semi-automatic merging method that employs language-based and linguistic resources ontology matching techniques in the merging process (Stumme and Maedche 2001). FCA-MERGE applies these matching techniques to the initial step of the merging process. This can be achieved by first taking two source ontologies and a set of documents that are relevant to both ontologies as input. Using the embedded tokenizer, word and domain lexicon as well as chunk parser, FCA-MERGE conducts a lexical analysis on the documents with the purpose of extracting instances for every concept in the ontologies. Two formal contexts which describe the extraction results of two ontologies are generated as output for this step. The second step of the merging process involves application of the FCA-MERGE core algorithm to merge the two contexts and computes a pruned concept lattice. The ontology engineer is responsible to create the merging ontology based on the depiction of the concept lattice.

IF-Map is an automatic mapping method that uses model-based matching technique in its mapping process (Kalfoglou and Schorlemmer 2003). The technique is grounded on Channel Theory which provides a mathematical model to depict the flow of information in the connection channel between communities by means of tokens and types (Barwise and Seligman 1997). IF-Map associates ontology with local logics (a set of concepts, instances and relations) and formalizes the mappings in terms of logic informorphisms (morphisms between local logics). IF-Map also exploits a set of heuristics as well as string-based and taxonomy-based techniques to derive partial mappings between concepts of the ontologies.

Combining Match Algorithm (COMA) is an ontology matching system that provides a platform to combine multiple matchers in a flexible way (Do and Rahm 2002). Its flexibility not only allows user to choose from a wide variety of simple (single) matchers such as string-based, constraint-based and linguistic resources matchers, it also provides a number of hybrid matchers where different match criteria or properties are implemented in an algorithm to serve in a specific way. For example, name matcher combines several string-based, language-based and linguistic
resources matchers in order to derive similarities between element names. In addition, COMA offers an innovative reuse-oriented matcher in which user can reuse previous match results or alignments for the entire new ontologies or its fragments. Furthermore, the above simple matchers and hybrid matchers can be combined to form a composite matcher in accordance with the nature of the matching task at hand.

COMA divides matching process into three phrases: 1) an optional user feedback phase, 2) the execution of various matchers, and 3) aggregation and interpretation of similarity results. User can specify the matching process to take place in one or multiple iterations by selecting from either the interactive or automatic mode. In the interactive mode, user has to determine a combination of matchers, similarity aggregation algorithm and its cut-off point in each iteration. The user also needs to define match or mismatch relationships and accept or reject matching candidates from previous iteration. In the automatic mode, the matching process iterates only once with a default matching strategy.

The experiments show the superiority of the composite matching approach. The best composite matching approach achieves 95%, 80% and 70% in the average precision, recall and overall evaluation (Do and Rahm 2002). Moreover, the composite matching approach also provides a more stable and accurate similarity results in contrast to the simple one. The experiments also indicate the high value of reuse-oriented matcher. The reuse-oriented matcher combinations increase by more than 10% of matching quality as compared with the best no-reuse approaches.

COMA++ includes all of the matching methods from COMA. These include simple matchers, hybrid matchers, reuse-oriented matchers and composite matching approach (Aumueller et al. 2005). In addition, COMA++ extends COMA by providing a comprehensive graphical user interface with access to its five main components: 1) the repository to store various types of match-related data, 2) the model pool to import, load and save external ontologies, 3) the execution engineer to perform automatic ontology matching, 4) the match customizer to configure matchers, similarity aggregation algorithms and cut-off point, and 5) the mapping pool to maintain all generated mappings. Moreover, COMA++ includes a new string-
based and linguistic resources matcher that makes use of domain specific thesauri to find ontological similarity. Evaluation results show that COMA++ provides high quality matching results even on large scale real-world ontologies. COMA++ also outperforms COMA in execution times when dealing with large matching problems (Aumüller et al. 2005).

Ontology Mapping Enhancer (OMEN) is a matching enhancing tool designed to improve existing ontology mappings using a Bayesian Net (Mitra et al. 2005). As mappings generated automatically or manually are sometimes imprecise and attached with some uncertainties, OMEN analyses the neighbourhood of these mappings to derive missed matches and invalidate existing false matches. The enhancing process is initiated by taking two source ontologies, a set of pairs of matches and their probability values (generated by a priori matcher) as inputs. OMEN uses a node to represent each pair of matches in the BayesNet graph and marks the node as evidence node if its probability value exceeds a predefined threshold. If the distance between a node and the evidence node is greater than a certain value, the particular node is pruned to control the size of the BayesNet graph. Before running the Bayesian Net, OMEN needs to generate conditional probability tables using a set of meta-rules in which the tables describe how a match between two classes affects other matcher. With the BayesNet graph and conditional probability tables, OMEN starts the inference process on the Bayesian Net and its output is a new set of matches. Evaluation results show that OMEN performs reasonably well in identifying and enhancing existing ontology mappings especially when the priori matchers are underperformed in the first place (Mitra et al. 2005).

OWL-Lite Alignment (OLA) is designed to provide an environment to manipulate alignments of ontologies expressed in OWL, with an emphasis on OWL-Lite (Euzenat et al. 2004). In OLA, entities are compared according to their categories (class, object, property, relation, property instance, datatype, data value and property restriction label) using the same similarity function and on the same feature space. OLA uses a labelled graph to describe the ontology in which nodes and edges are used to represent entities and relationships respectively. Similarity measure of the OLA is then defined by a system of quasi-linear equations and its similarity values are derived by means of an iterative approximation process which starts with
measuring labels of the nodes and gradually expands to their neighbouring nodes. The similarity measuring model used to compare two nodes of different ontologies depends on: 1) similarity of their labels, 2) similarity of their neighbouring nodes, and 3) similarity of other descriptive knowledge. Hence, OLA adopts string-based, language-based, linguistic resources matching techniques to execute the above measuring model. Evaluation results show that OLA performs well when comparing two ontologies with closed or identical structures. However, Euzenat et al. (2004) emphasize that further experiments are required to determine if OLA can produce meaningful alignment on real ontologies.

S-Match is an automatic matching system that takes two graph-like taxonomies as input and computes the strongest semantic relations holding between any pair of nodes from two taxonomies (Giunchiglia and Shvaiko 2003; Giunchiglia et al. 2004). The matching approach of S-Match is based on two notions, the notion of concept of a label and the notion of concept of a node. While the former is defined as the set of documents that reflects the meaning of the labels, the latter is defined as the set of documents classified under this node. Before executing the semantic matching process, the system uses natural language processing techniques to tokenize and lemmatize labels of the taxonomies. Subsequently, the labels are required to translate from natural language into more precise internal format using WordNet. For complex concepts, logical connectives are used to replace tokens that contain prepositions, punctuation marks and conjunctions. To derive the strongest relationship, S-Match computes the concept of label matrix (which contains relations between any two concepts of labels in the two taxonomies) using both the string-based and linguistic resources matchers. Then information obtained from the above matrix together with concepts of labels and concepts of nodes are codified as a set of complex propositional formulas. Finally, S-Match generates another matrix containing the strongest relations holding between concepts of nodes by applying model-based techniques to process the propositional formulas. Evaluation results show that S-Match is able to generate high quality matching results in terms of precision, recall, overall and f-measure (Giunchiglia et al. 2004). However, its complexity sacrifices the efficiency of the process required to produce matchings compare to other matching systems.
LILY is an automated ontology mapping system that is designed to discover mappings for both normal and large scale ontologies (Wang and Xu 2008). To perform ontology matching, the system first extracts semantic subgraphs from the matched ontologies. The system then computes the similarity of entities from different ontologies by analysing the literal and structural information on the semantic subgraphs. There are three matching techniques embedded to find similarity of entities: Generic Ontology Matching method (GOM), Large Scale Ontology Matching method (LOM) and Semantic Ontology Matching Method (SOM). GOM adopts semantic description document (SDD) matcher to measure the literal similarity between small scale ontologies using literal information for concepts (such as class hierarchies, related properties and instances) and properties (such as hierarchies, domains, ranges, restrictions and related instances) presented in semantic description documents. LOM uses negative and positive reduction anchors to reduce time complexity in matching. The negative anchor uses the locality of matching to predict ignorable similarity calculations and the positive anchor uses concept hierarchy to predict ignorable similarity calculations. Instead of using common knowledge or a domain specific thesaurus to derive meanings of entities in ontologies, SOM uses search results gained from search engine as a large knowledge base to find semantic relationships between ontologies.

Context and Inference Based Aligner (CIDER) is a schema-based ontology matching system that relies on schema-level input information for performing ontology matching (Gracia and Mena 2008). CIDER discovers similarity by first extracting the ontological context of each involved term including its synonyms, textual descriptions, hypernyms, properties, domains, roles, associated concepts and so on. Similarity computation is performed by comparing the extracted information from a pair of terms originated from two ontologies using string-based matcher. It is then followed by another computation that explores the structural similarity of the terms exploiting their ontological contexts. This also includes comparison of taxonomies and relationships among terms using vector space modelling and the degree of similarity is provided for each compared pair based on the matching performance. At the end, a matrix consisting of all similarities is generated where the compared pairs are filtered if the similarities fall below a given threshold.
Automated Semantic Matching of Ontology with Verification (ASMOV) is an ontology matching algorithm that combines a comprehensive set of element- and structure-level similarity measure together with a technique that uses formal semantics to ensure computed correspondences do not contain semantic inconsistencies (Jean-Mary et al. 2009). ASMOV computes a similarity value between all possible pairs of entities iteratively, one from each of the two ontologies. The similarity value is calculated using: 1) a linguistic resources matcher to analyse labels, identities and comments, and 2) a taxonomy-based matcher to analyse specialization relationships of all parents and children, property restrictions for concepts, types, domains and ranges for properties, data values for individuals as well as data instances. The measurements obtained from the above analysis are aggregated into a single value using a weighted sum.

Table 3.2 Summary of Mediation Tools, Frameworks and Methods, and their Inherent Matching Techniques

<table>
<thead>
<tr>
<th>Tool</th>
<th>Automation</th>
<th>String-based</th>
<th>Language-based</th>
<th>Constraint-based</th>
<th>Linguistic Resources</th>
<th>Alignment Reasoning</th>
<th>Upper Level Reasoning</th>
<th>Graph-based</th>
<th>Taxonomy-based</th>
<th>Repository of Structures</th>
<th>Model-based</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chimaera</td>
<td>Semi</td>
<td>Semi</td>
<td>Semi</td>
<td>Semi</td>
<td>Semi</td>
<td>Semi</td>
<td>Heuristic on Protoset Reasoning</td>
<td>Heuristic on Protoset Reasoning</td>
<td>Heuristic</td>
<td>Heuristic on Protoset Reasoning</td>
<td>Heuristic on Protoset Reasoning</td>
</tr>
<tr>
<td>jPROMPT</td>
<td>Semi</td>
<td>Semi</td>
<td>Semi</td>
<td>Semi</td>
<td>Semi</td>
<td>Semi</td>
<td>Heuristic</td>
<td>Heuristic</td>
<td>Heuristic</td>
<td>Heuristic on Protoset Reasoning</td>
<td>Heuristic on Protoset Reasoning</td>
</tr>
<tr>
<td>MAFA</td>
<td>Full</td>
<td>Semi</td>
<td>Semi</td>
<td>Semi</td>
<td>Semi</td>
<td>Semi</td>
<td>Heuristic</td>
<td>Heuristic</td>
<td>Heuristic</td>
<td>Heuristic on Protoset Reasoning</td>
<td>Heuristic on Protoset Reasoning</td>
</tr>
<tr>
<td>NOM</td>
<td>Full</td>
<td>Heuristic</td>
<td>Heuristic</td>
<td>Heuristic</td>
<td>Heuristic</td>
<td>Heuristic</td>
<td>Heuristic</td>
<td>Heuristic</td>
<td>Heuristic</td>
<td>Heuristic on Protoset Reasoning</td>
<td>Heuristic on Protoset Reasoning</td>
</tr>
<tr>
<td>QOM</td>
<td>Full</td>
<td>Heuristic</td>
<td>Heuristic</td>
<td>Heuristic</td>
<td>Heuristic</td>
<td>Heuristic</td>
<td>Heuristic</td>
<td>Heuristic</td>
<td>Heuristic</td>
<td>Heuristic on Protoset Reasoning</td>
<td>Heuristic on Protoset Reasoning</td>
</tr>
<tr>
<td>iTalka</td>
<td>Semi</td>
<td>Semi</td>
<td>Semi</td>
<td>Semi</td>
<td>Semi</td>
<td>Semi</td>
<td>Heuristic on Protoset Reasoning</td>
<td>Heuristic on Protoset Reasoning</td>
<td>Heuristic</td>
<td>Heuristic on Protoset Reasoning</td>
<td>Heuristic on Protoset Reasoning</td>
</tr>
<tr>
<td>GLUE</td>
<td>Semi</td>
<td>Semi</td>
<td>Semi</td>
<td>Semi</td>
<td>Semi</td>
<td>Semi</td>
<td>Heuristic</td>
<td>Heuristic</td>
<td>Heuristic</td>
<td>Heuristic on Protoset Reasoning</td>
<td>Heuristic on Protoset Reasoning</td>
</tr>
<tr>
<td>F-Map</td>
<td>Full</td>
<td>Semi</td>
<td>Semi</td>
<td>Semi</td>
<td>Semi</td>
<td>Semi</td>
<td>Heuristic</td>
<td>Heuristic</td>
<td>Heuristic</td>
<td>Heuristic on Protoset Reasoning</td>
<td>Heuristic on Protoset Reasoning</td>
</tr>
<tr>
<td>COMA</td>
<td>Full / Semi</td>
<td>Semi</td>
<td>Semi</td>
<td>Semi</td>
<td>Semi</td>
<td>Semi</td>
<td>Heuristic</td>
<td>Heuristic</td>
<td>Heuristic</td>
<td>Heuristic on Protoset Reasoning</td>
<td>Heuristic on Protoset Reasoning</td>
</tr>
<tr>
<td>COMA++</td>
<td>Full / Semi</td>
<td>Semi</td>
<td>Semi</td>
<td>Semi</td>
<td>Semi</td>
<td>Semi</td>
<td>Heuristic</td>
<td>Heuristic</td>
<td>Heuristic</td>
<td>Heuristic on Protoset Reasoning</td>
<td>Heuristic on Protoset Reasoning</td>
</tr>
<tr>
<td>QMEN</td>
<td>Full</td>
<td>Heuristic</td>
<td>Heuristic</td>
<td>Heuristic</td>
<td>Heuristic</td>
<td>Heuristic</td>
<td>Heuristic on Protoset Reasoning</td>
<td>Heuristic on Protoset Reasoning</td>
<td>Heuristic</td>
<td>Heuristic on Protoset Reasoning</td>
<td>Heuristic on Protoset Reasoning</td>
</tr>
<tr>
<td>OCLA</td>
<td>Full</td>
<td>Semi</td>
<td>Semi</td>
<td>Semi</td>
<td>Semi</td>
<td>Semi</td>
<td>Heuristic</td>
<td>Heuristic</td>
<td>Heuristic</td>
<td>Heuristic on Protoset Reasoning</td>
<td>Heuristic on Protoset Reasoning</td>
</tr>
<tr>
<td>S-Match</td>
<td>Full</td>
<td>Semi</td>
<td>Semi</td>
<td>Semi</td>
<td>Semi</td>
<td>Semi</td>
<td>Heuristic</td>
<td>Heuristic</td>
<td>Heuristic</td>
<td>Heuristic on Protoset Reasoning</td>
<td>Heuristic on Protoset Reasoning</td>
</tr>
<tr>
<td>Lily</td>
<td>Full</td>
<td>Heuristic</td>
<td>Heuristic</td>
<td>Heuristic</td>
<td>Heuristic</td>
<td>Heuristic</td>
<td>Heuristic</td>
<td>Heuristic</td>
<td>Heuristic</td>
<td>Heuristic on Protoset Reasoning</td>
<td>Heuristic on Protoset Reasoning</td>
</tr>
<tr>
<td>ODER</td>
<td>Full</td>
<td>Semi</td>
<td>Semi</td>
<td>Semi</td>
<td>Semi</td>
<td>Semi</td>
<td>Heuristic</td>
<td>Heuristic</td>
<td>Heuristic</td>
<td>Heuristic on Protoset Reasoning</td>
<td>Heuristic on Protoset Reasoning</td>
</tr>
<tr>
<td>ASMOV</td>
<td>Full</td>
<td>Semi</td>
<td>Semi</td>
<td>Semi</td>
<td>Semi</td>
<td>Semi</td>
<td>Heuristic</td>
<td>Heuristic</td>
<td>Heuristic</td>
<td>Heuristic on Protoset Reasoning</td>
<td>Heuristic on Protoset Reasoning</td>
</tr>
</tbody>
</table>

Table 3.2 shows the summary of the above survey consisting of eighteen mediation tools, frameworks and methods with their inherent matching techniques. The most popular ontology matching techniques are string-based, taxonomy-based, constraint-
based as well as linguistic resources techniques and each of these techniques is used by at least nine out of the eighteen mediation systems. In contrast, the least popular matching techniques are repository of structures technique and upper level formal ontologies. While the former technique is adopted by only one mediation system, the latter is not adopted by any system at all. Almost all systems in the survey incorporate a graph algorithm as their matching technique (either graph-based or taxonomy-based technique) with the exception of FCA-MERGE, iPROMPT and Chimaera. For those systems which use graph algorithm as a matching technique (except GLUE and ITTalks), they include one additional matching technique in the system. Most of the mediation systems exploit multiple matching strategy consisting of more than one matching technique. For instance, both COMA and COMA++ include six matching techniques in their inherent matching strategy. Thus leaving ITTalks, iPROMPT, Chimaera and GLUE to engage with a single strategy in which only one matching technique is included in each system. In terms of execution approach, heuristic is widely implemented for conducting string-based, language-based, constraint-based, linguistic resources, alignment reuses, graph-based, taxonomy-based and repository of structures matching technique. Probabilistic reasoning approach, such as Bayesian network and machine learning, also plays a role in the execution of taxonomy-based technique whereas semantic reasoning is the dedicated approach used to execute model-based technique. Of the eighteen mediation systems, eleven of these systems are capable of performing ontology matching automatically, five of the systems still rely on human intervention and the remaining two systems allow user to execute ontology matching either automatically or semi-automatically.

3.2 A Design and Input-specific Classification of Ontology Matching Techniques

Shvaiko and Euzenat’s (2005) classification is designed to standardize a conceptual basis for comparing different existing ontology mediation systems as well as for designing a new one. Firstly, it is necessary to identify the types of ontology matching techniques implemented in the mediation tools. One way to identify the matching technique is by examining its input. For example, adoption of a taxonomy-
based matcher is substantiated when a mediation tool takes two taxonomies that contain subset and superset relationships as inputs. However, the two synthetic views (top-down and bottom-up views) of the classification only provide a very general description on classifying the inputs which may become an obstacle for carrying out identification task in an effective manner. The use of ambiguous categories to classify inputs such as terminological, structural, semantic, syntactic and external offers only limited help in the process of matcher type identification. Hence, it is desirable to replace the ambiguous categories with a more specific input layer in the classification.

In the aspect of designing new mediation approach, a more specific input layer is essential to illustrate the relationship between the ten elementary ontology matchers and their inputs. Apart from that, this classification lacks the depiction of executive approach used to perform the related ontology matching technique. For instance, semantic reasoning approach is always used to execute model-based techniques (Giunchiglia and Shvaiko 2003; Kalfoglou and Schorlemmer 2003). It is necessary for mediation tool designers and developers to understand the relationship between the executive approaches and elementary matching techniques. Such an understanding not only provides guidelines for designing new tools, but it also helps to speed up the time it takes for design and development. Thus, an additional level of executive approach should be included in the classification.

Although the ten elementary matching techniques are categorised conscientiously, there are still several misidentifications. The first misidentification is the language-based matching technique. As mentioned earlier, this technique is normally performed prior to string-based technique and has no direct engagement in the actual similarity computation between two ontologies. In fact, this technique should be used to normalize the syntactic heterogeneity within the matching process. It is more appropriate to consider this natural language processing technique as part of the matching process (refer to feature engineering at the pre-matching stage of the ontology matching process in Figure 3.1), rather than labelling it as one of the matching techniques.

The second misidentification is the repository of structures technique. Instead of
finding a corresponding entity (or entities) in the second ontology for each entity in the first ontology, this technique compares fragments of two ontologies and eliminates dissimilar portions to improve computational efficiency and cost in the matching process. Hence, the repository of structures technique is a dynamic approach to select a more suitable set of matching candidates and should not be classified as one of the matching techniques. In fact, it should be regarded as part of the matching process (refer to search step selection at the pre-matching stage of the ontology matching process in Figure 3.1)

The third misidentification is upper level formal ontologies technique. Shvaiko and Euzenat (2005) state that there is no mediation system using this technique. The finding is confirmed through the analysis conducted in the previous section when reviewing some of the most significant mediation approaches. Consequently, it should not include a non-existing technique in the classification.

The classification defined by Shvaiko and Euzenat (2005) comprehensively itemizes ten elementary matching techniques. However, three of the techniques are improperly identified, namely language-based matching, repository of structures and upper level formal ontologies techniques. Moreover, the lack of an executive approach layer and a detailed input type layer magnifies its incapability of allowing researchers, scholars, tool designers and system developers to perform the technique identification and tool designing tasks.

To address the graph in this area, this research proposes to develop a new Design and Input-Specific Classification of Ontology Matching Technique (DISCOMT) that addresses the three misidentifications. DISCOMT consists of the executive approach and input type layers. The proposed DISCOMT aims to provide clear guidelines on designing new mediation approach. It also aims to provide a relatively easier method to identify the type of the matching technique and its related executive approach.
Figure 3.3 shows DISCOMT which is divided into three tiers. Tier One (T1) is executive the approach layer, Tier Two (T2) is the basic technique layer and Tier Three (T3) is the input type layer. These three layers are included in DISCOMT because executive approach, matching technique(s) and inputs are three major components of any ontology mediation approaches. Therefore, it is essential for ontology developers to consider when designing an ontology mediation approach.

In T2, language-based, upper level formal ontologies and repository of structures matching techniques as described in Shvaiko and Euzenat’s (2005) classification are not included in the proposed DISCOMT classification to address the three misidentifications. Thus, only string-based, linguistic resources, constraint-based, alignment reuse, graph-based, taxonomy-based and model-based matching technique are included. In T3, a more detailed input type layer is specified by dividing into two levels. These eight input types are further subcategorised into elementary and structural input types.
There are two different ways to study the proposed DISCOMT either by viewing from T3 or T2. The T3 view provides an easier way to identify the type of ontology matching technique and its executive approach simply by comparing input of the mediation system or tool with the input types on T3. For example, the matcher is most likely string-based (T2) if it takes names and descriptions of entities (T3) as input.

The T2 view describes the relationships between basic technique (T2) and executive approach (T1) layers, and between basic technique (T2) and input type (T3) layers. This view provides guidelines on designing new mediation approach. For instance, to exploit model-based (T2) as a mediation system’s matching technique, tool designer must ensure propositional formulas (T3) and semantic reasoning (T1) are used for input type and executive approach respectively. This view also indicates the approach requires to execute a particular matching technique. For example, string-based technique (T2) can only be executed by heuristic approach (T1).

In T1, heuristic, probabilistic reasoning and semantic reasoning are identified as the three major executive approaches to execute the seven elementary ontology matching techniques specified in T2. Heuristic approach exploits rules for comparing syntactic features, properties, linguistic and structural information of two or more different ontologies. This approach is widely used in the execution of string-based, constraint-based, linguistic resources, alignment reuse, graph-based and taxonomy-based matching technique. In terms of string-based matching technique, heuristic approach establishes rules to determine the matching entities based on the similarity computation of representational strings from two ontologies. Examples of string-based rules include:

- Two entities are identical if their representational strings are identical.
- Two entities are identical if their representational strings contain the same prefix or suffix.
- The similarity of two entities is higher if it requires fewer steps to convert one representational string to another.
While the string-based matching technique focuses only on calculating the string similarity of properties between two ontologies, the constraint similarity of the properties are handled using the constraint-based technique. Here, heuristic approach applies rules to find matching properties based on internal constraints applied to each property. Examples of constraint-based rules are:

- The level of similarity between two properties is higher if their datatypes are more compatible (for example, double and integer are compatible as double can be casted to integer or vice versa).
- The level of similarity between two properties is higher if the ranges of their values are closer.

Linguistic resources matching technique uses a common knowledge or a domain specific thesaurus to derive meanings of entities in ontologies. By taking these meanings as input, heuristic rule is capable of determining the linguistic relations (such as synonyms, hyponyms and hypernyms) among the entities. For example, if a linguistic resources matcher derives from a common knowledge thesaurus that Laptop in Ontology A is a hyponym of Computer of Ontology B, heuristic can determine Laptop in A is subsumed by Computer in B. Another example is to find semantic relationships between ontologies using search results gained from search engine as a thesaurus.

Alignment reuse matching technique makes use of previously matching results at the level of ontology fragments or entire ontologies to derive new matching results. Heuristic of the technique is built on a transitive nature of the similarity relation between elements. This transitive nature means that if x is similar to y and y is similar to z, then x is very likely similar to z. It allows heuristic to reuse the available alignment information for matching analysis when Ontology B and C are required to match with each other, given that the matching results between A and B as well as between A and C were stored.

Graph-based matching technique takes two ontologies in the form of labelled graphs as input and nodes from the ontologies are compared and analysed to derive similarity of their neighbouring nodes. Examples of rules adopted by heuristic of
graph-based technique include:

- Two nodes are similar if their immediate children nodes are highly similar.
- Two nodes are similar if their leaf nodes are highly similar.
- Two nodes are similar if their relations are similar.

Similar to graph-based technique, taxonomy-based matching technique also takes graph as input. However, the graph input is more rigorous because neighbouring nodes on the graph are connected with is-a links to indicate they are superset/subset of each other. Heuristic can be applied to compare and identify similar nodes along the paths connected by is-a links. Examples of rules adopted by heuristic of taxonomy-based technique are:

- Two nodes are similar if their super-nodes are the same.
- Two nodes are similar if their sub-nodes are the same.

In T1, probabilistic reasoning approach, such as Bayesian network and machine learning, can also be used to execute taxonomy-based technique. Probabilistic reasoning uses probability measurement to calculate similarity of two concepts from two different taxonomies that are similar or having the same instances. When two independent taxonomies contain a pair of similar nodes, for example Node A and B, it is possible to induce new set(s) of similar nodes from the taxonomies by considering the probabilistic similarity measured between Node A and neighbours of Node B, and between Node B and neighbours of Node A.

To execute model-based matching technique from T1, semantic reasoning approach first needs to translate relationships of all possible matching candidates of two ontologies into some forms of propositional formula, such as axioms and local logics. Subsequently, the approach adopts sound deduction method to validate matching between two ontologies based on the semantic of propositional formulas. For instance, propositional satisfiability solver is used to check possible matching candidates by validating their propositional formulas.
In T3, input types are further classified into two levels. The second level of this layer contains two keywords used to sum up the characteristics of the actual inputs on the first level: elementary and structural. Elementary input represents input that undergoes analysis in isolation during the matching process without the need of considering its relations with other entities. Names and descriptions of entities as well as datatypes and values of properties are classified as elementary input and description of the inputs is listed as follows.

- Name of an entity refers to a sequence of characters that are used to name any entity (such as label, relation and property) in an ontology.
- Description of an entity is defined as a sequence of characters that are used to describe any entity (such as label, relation and property) in an ontology.
- Datatype of a property refers to the type of data associated with a property.
- Value of a property is the numerical quantity assigned to a property.

In contrast, mediation systems analyse structural input in accordance with its relations with other entities in the process of ontology matching. Alignments, graphs, taxonomies and propositional formulas are categorises as structural input and description of the inputs is listed as follows.

- Alignment is the matching result obtained from previously matched ontologies.
- Graph is a type of structural representation that contains nodes and their inter-relationships.
- Taxonomy is similar to a graph except its nodes are connected with each other by is-a links. The connected nodes are either subset or superset of one another.
- Propositional formula refers to a semantic statement that describes a pair of possible matching candidates and their relations.

Let us consider the following three examples to demonstrate the use of the DISCOMT in the real world. In example one, an ontology developer chooses to use string-based matching technique to develop a simple ontology mapping tool to be used in an environment that only contains simple lightweight ontologies. Using the proposed DISCOMT, the ontology developer can design a string-based heuristic
rule(s) matcher that takes both names and descriptions of entities between two ontologies as input for comparison. Depending on the level of complexity in the environment, the developer can choose to implement one rule or a set of rules such as comparing representational strings or counting number of steps required to convert one representational string to another.

In example two, an ontology developer wants to build a mapping tool for a relatively more complex environment with a combination of matching techniques including string-based and taxonomy-based executed by heuristic approach. According to the DISCOMT, the tools must be able to convert input ontologies to taxonomies consisting of nodes interconnected with is-a links for taxonomy-based matcher to act. The ontology developer chooses to implement two taxonomy-based rules in which super-nodes and sub-nodes are compared to identify similar nodes. Besides, the developer also implements a string-based matcher to compare representation strings of entities names from two ontologies. In example three, an ontology developer wants to develop an ontology merging tool that can be used among a group of complicated heavyweight ontologies, the ontology developer can adopt model-based matching technique executed by semantic reasoning approach that takes propositional formulas from two ontologies as input.

Whether to adopt a single or a combination of matching techniques is another major concern when designing a mediation tool. Multiple matching techniques can outperform a single matching technique in terms of accuracy if a suitable combination of techniques is selected together with an appropriate similarity aggregation algorithm and a proper cut-off point.

### 3.3 Conclusion

Shvaiko and Euzenat’s (2005) classification aims to provide a conceptual basis of ontology matching techniques for comparing existing ontology mediation system and for designing a new mediation tool. Upon conducting an analysis on their classification and a literature survey on eighteen ontology mediation approaches, this research has identified a number of weaknesses which include misidentification of
matching techniques, and the lack of an executive approach layer and a detailed input layer. These problems result in classification being impossible to use as intended.

To address the research gap, this research proposes a new guideline called DISCOMT to address the above problems. The proposed DISCOMT consists of three tiers. T1 consists of three executive approaches. The first executive approach is heuristic where rules are implemented to compare syntactic features, properties, linguistic and structural information of two or more ontologies. The second approach is probabilistic reasoning that uses probabilistic measurement to calculate similarity of concepts from two ontologies with similar instances. The third approach is semantic reasoning that adopts deduction method to validate semantic of propositional formulas between two ontologies.

The executive approaches are used to execute the seven matching techniques presented in T2. However, each executive approach is specialised to perform some matching techniques due to the design of the approach. For example, semantic reasoning approach is designed to execute model-based matching technique. These seven matching techniques specify the techniques used in existing mediation algorithms, tools and methods and lay a solid foundation for ontology developers who want to build ontology mediation tools. T3 is a two-level input type layer in which the eight input types are further subcategorised into elementary and structural input types.

The proposed DISCOMT framework aims to provide a simpler and more effective way to identify the type of ontology matching technique through the use of the executive approach by comparing input of mediation approaches with the input types. This can be achieved by a T3 view. DISCOMT also aims to provide a more comprehensive guideline on designing new mediation approach. This can be achieved by a T2 view as this view describes the relationships between basic technique and executive approach layers, and between basic technique and input type layers. DISCOMT framework has been demonstrated to develop ontology mediation tools that include a single or a combination of matching techniques.
Ontology building is sometimes considered as an art rather than a science, thus there is not a single correct ontology development methodology (Grubic and Fan 2010; Jones et al. 1998; Mizoguchi 2003). To build high quality ontologies, ontology developers are required to choose and follow a suitable methodology which contains a series of steps, activities and guidelines (including purpose articulation, knowledge acquisition and formalization and evaluation) in an organised and systematic manner. Each methodology may have its own features but the majority agree ontology reuse is essential in their ontology development process such as Methontology (Fernandez-Lopez et al. 1997), Collaborative Design Approach (Holsapple and Joshi 2002), Scenario-based Ontology Development (Lee 2006), Ontology Development 101 (Noy and McGuinness 2001) and Enterprise Ontology Approach (Uschold and King 1995). Researchers point out that building ontology by reusing existing ontologies is more cost effective than building it from scratch (Pinto and Martins 2001a). Time, manpower and other resources can be saved if suitable knowledge modules can be found from existing ontologies. Savings depend on whether the knowledge modules need to be modified before reuse. In general, saving decreases if the modules require more modifications.

This chapter aims to propose an ontology development methodology that integrates ontology reuse methods and a system that can assist to perform integration semi-automatically. The system adopts the ontology matching technique to allow ontology reuse to be performed semi-automatically. Hence, the DISCOMT presented in Chapter 3 can provide a reference point for choosing the appropriate ontology matching techniques. The proposed ontology development methodology aims to address the limitation of the existing ontology development methodologies that fail to provide in-depth information about how to perform integration in their ontology development processes.

This chapter is organized as follows. Section 1 analyses ten significant ontology development methodologies. A detailed analysis of ontology integration and merging...
that includes guidelines for choosing ontology reuse methods are also provided in Section 1. Section 2 discusses the proposed methodology of integration-oriented ontology development and the candidate ontology evaluation systems. Finally, conclusion is presented in Section 3.

4.1 Analysis of Ontology Development Methodology

Ten different ontology development methodologies were studied that include Methontology (Fernandez-Lopez et al. 1997), Enterprise Ontology Approach (Uschold and King 1995), Toronto Virtual Enterprise (TOVE) (Gruninger and Fox 1995), Ontology Development 101 (Noy and McGuinness 2001), Scenario-based Ontology Development (Lee 2006), Collaborative Design Approach (Holsapple and Joshi 2002), Unified Methodology (Uschold 1996), DILIGENT Knowledge Processes (Pinto et al. 2009), Ontology-based KM Systems Methodology (Staab et al. 2001) and Domain Ontology Development Process (Brusa, et al. 2008).

Methontology is developed at the Polytechnic University of Madrid which provides a number of development, supporting and project management activities in its ontology development life cycle (Fernandez-Lopez et al. 1997). Enterprise Ontology Approach provides ontology development guidelines developed based on the experience of building the ontologies for enterprise modelling (Uschold and King 1995). TOVE is another methodology developed based on development experience of a set of ontologies for TOVE project including Enterprise Design, Project, Scheduling and Service Ontology (Gruninger and Fox 1995). Ontology Development 101 is designed for the development of ontology for declarative frame-based systems in which some of its design guidelines are based on object OOAD (Noy and McGuinness 2001). Scenario-based Ontology Development follows phases of Methontology’s development life cycle but is focused on the adoption of scenarios in each phase (Lee 2006).

Collaborative Design Approach provides a mechanism for members of ontology development team to share and exchange their perspectives and expertise in developing, refining and modifying ontology (Holsapple and Joshi 2002). Unified
Methodology merges Enterprise Ontology Approach and TOVE into a coherent ontology building framework to broaden the range of circumstances supported by the methodology (Uschold 1996). DILIGENT Knowledge Processes is an ontology engineering methodology which focuses on the evolution of ontologies in a distributed, loosely-controlled and ontology development environment rather than the initial design with no further improvement in a centralised environment (Pinto et al. 2009). Ontology–based KM Systems Methodology is designed to develop an application-oriented ontology used in ontology-based KM system (Staab et al. 2001). Domain Ontology Development Process takes advantage of various ontology development methodologies and software engineering techniques to define a unified process for the development of a domain specific ontology in public section (Brusa, et al. 2008).

Each of the ontology development methodologies is established under different scenarios with various characteristics: some methodologies are derived from the experience of building a particular ontology when requirements are clearly understood at the very beginning such as Ontology-based KM Systems Methodology (Staab et al. 2001), TOVE (Gruninger and Fox 1995) and Enterprise Ontology Approach (Uschold and King 1995), some are developed using iteration approach when requirements are not clearly understood initially such as Ontology Development 101 (Noy and McGuinness 2001), Methontology (Fernandez-Lopez et al. 1997), Collaborative Design Approach (Holsapple and Joshi 2002) and DILIGENT (Pinto et al. 2009), some use collaborative approach to evolve ontologies according to users’ usage and feedbacks such as DILIGENT Knowledge Processes (Pinto et al. 2009) and Collaborative Design Approach (Holsapple and Joshi 2002), some are structured by combining two to three other methodologies such as Unified Methodology (Uschold 1996) and Domain Ontology Development Process (Brusa et al. 2008) and Scenario-based Ontology Development (Lee 2006) puts its focus on using scenarios in different development activities.

Ontology development methodologies generally consist of several phases in their development process. For instance, Methontology has three phases in its process, namely pre-development, development and post-development and each phase contains a number of activities (Fernandez-Lopez et al. 1997). There are two
activities that need to be executed, environmental and feasibility study, in the pre-development stage of Methontology. In the development phase, the first activity that needs to be completed is to specify purpose, intended use and potential users of the ontology, followed by conceptualising the domain knowledge into meaningful knowledge-level models. After formalizing the conceptualized models into formal or semi-computable models, the development phase can be concluded by building computable models using an ontology language. In the post-development phase, the resulting ontologies are validated and updated if needed. Besides, the authors also added a series of management (such as scheduling, control and quality assurance) and supporting activities (such as integration, merging, alignment and documentation) that can be performed throughout the development process to make sure the development process is on track.

If compare to other ontology development methodologies with the three-phases Methontology, most of the ontology development methodologies only specify activities in the development phase. Of the ten methodologies, only three methodologies include all three phases and the remaining seven methodologies either exclude pre-development phase, post development phase or both, for example, Diligent knowledge process has only the development phase with the following activities: 1) it starts by building a core ontology using combined effort of domain experts, users, knowledge engineers and ontology engineers, 2) users can start to use and localise core ontology according to their own business requirement, 3) the control board analyses the local ontologies and change requests to decide which changes will be introduced in the next version of the core ontology, 4) the control board has to revise the core ontology regularly so that it does not diverge too far from the local ontologies, and 5) the users can update their own ontologies once a new version is released (Pinto et al. 2009). Phases, activities and other characteristics of the ten ontology development methodologies are summarised in Table 4.1.
Table 4.1 Summary of Phases, Activities and Characteristics in Ten Ontology Development Methodologies

<table>
<thead>
<tr>
<th>Methodology</th>
<th>Phases / Activities</th>
<th>Include all three phases (Pre-development, Development and Post-development)</th>
<th>Ontology Reuse</th>
</tr>
</thead>
<tbody>
<tr>
<td>Methontology (Fenandez-Lopez et al. 1997)</td>
<td>• Pre-development: Environment and feasibility study</td>
<td>Yes</td>
<td>Yes. Describe the process required to ensure candidate ontology must be coherent with concepts identified in ontology conceptualization activity</td>
</tr>
<tr>
<td></td>
<td>• Development: Specification</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Development: Conceptualization</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Development: Formalization</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Development: Implementation</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Post-development: Maintenance</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Plus additional 5 supportive activities and 3 Management activities</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Enterprise Ontology Approach (Uschold and King 1995)</td>
<td>• Development: Identify the purpose</td>
<td>No</td>
<td>Yes. Describe difficulties of ontology reuse.</td>
</tr>
<tr>
<td></td>
<td>• Development: Ontology capture</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Development: Coding</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Development: Integrating existing ontology</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Development: Evaluation and documentation</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TOVE (Gruninger and Fox 1995)</td>
<td>• Development: Capture motivating scenarios and formalize informal verification questions</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>• Development: Specify terminology of the ontology within a formal language</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Development: Formulate the verification questions</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Development: Specify axioms and definitions for the terms in the ontology</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Development: Establish conditions for characterizing the completeness of the ontology</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ontology Development 101 (Noy and McGuinness 2001)</td>
<td>• Development: Determine the domain and scope</td>
<td>No</td>
<td>Yes. Provide reason of reusing existing ontologies. Describe where ontology can be found.</td>
</tr>
<tr>
<td></td>
<td>• Development: Consider using existing ontologies</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Development: Enumerate key terms</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Development: Define classes and class hierarchy</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Development: Define properties of classes</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Development: Define facets of slots</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Development: Create instances</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Scenario-based Ontology Development (Lee 2006)</td>
<td>Followed the framework of Methonology but the focus is on the role of scenarios played in each activity</td>
<td>No</td>
<td>Yes. Discuss benefits of using scenarios in ontology reuse activities.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Collaborative Design Approach (Holsapple and Joshi 2002)</td>
<td>• Pre-development: Preparation</td>
<td>No</td>
<td>Yes. Briefly describe how existing ontologies can be reused to develop an initial ontology. State two benefits of reuse ontology.</td>
</tr>
<tr>
<td></td>
<td>• Development: Specify initial ontology</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Development: Identify diverse panel of participants</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Development: Elicit critiques and comments on the ontology from panellist</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Development: Revise the ontology to address panellists’ feedback</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Unified Methodology (Uschold 1996)</td>
<td>• Development: Identify the purpose</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>• Development: Decide level of formality</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Development: Identify the scope</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Choose one approach to build ontology (Development): o Build an ontology without going through above steps,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>o Go through the above steps and begin formal encoding</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>o Go through the above steps and complete an intermediate document and an informal ontology</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>o Go through the above steps and complete a</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Among the ten significant ontology development methodologies studied, the majority of these methodologies have involved integrating or reusing existing ontologies as one of their development activities rather than just building an ontology from scratch (Fernandez-Lopez et al. 1997; Uschold and King 1995; Noy and McGuinness 2001; Lee 2006; Holsapple and Joshi 2002; Staab et al. 2001). Those methodologies make recommendation to examine a set of existing ontologies so that the resulting ontologies can be partially or fully constructed by reusing one or more existing ontologies in their development process.

In fact, most of the ontology development methodologies only provide very little information and guidance on how to reuse existing ontologies in their development process. In the Enterprise Ontology Approach, Uschold and King (1995) use one short paragraph to describe the difficulties of reusing existing ontologies but no substantial guidelines is provided. In Ontology Development 101, Noy and McGuinness (2001) describe the reason of reusing existing ontologies is when a system needs to interact with other applications that have already committed to particular ontologies using one sentence. Besides, Noy and McGuinness (2001) also...
use another paragraph to depict where existing ontologies can be found but there is no description about the reuse process.

In Scenario-based Ontology Development, Lee (2006) provides no information about ontology reuse, except for a very general discussion on the benefits of using scenarios in ontology reuse activities. In the Collaborative Design Approach, Holsapple and Joshi (2002) briefly describe how existing ontologies can be reused to build an initial ontology. The process includes two steps: 1) identification of a base set of ontologies that do not subsume with each other, 2) synthesizing the set of ontologies to a unified ontology by integrating concepts, eliminating sketchier characterizations and reconciling different terminologies. In addition, Holsapple and Joshi (2002) provide a short conclusion regarding the benefits of synthesizing existing ontologies which include prone to adoption and interact coherently by adherents. In Ontology-based KM Systems Methodology, Staab et al. (2001) state that ontology developer should look for potentially reusable ontologies in the kick-off phase and provide a few benefits of using ontologies.

Among the ten development methodologies, Methontology provides relatively more detailed information on the topic of ontology reuse. Fernandez-Lopez et al. (1997) describe the process required to ensure the semantic, definition and implementation of candidate ontology must coherent with concepts identified during the ontology conceptualization activity. However, the downsides are the process only covers two steps (inspect meta-ontologies and search for appropriate terms from existing ontologies) and the description is too general that fails to give sufficient insight. Thus, a detailed description on how to reuse potential ontologies in an ontology development process is required.

As described in Chapter 2, ontology mediations can be used to resolve mismatches between heterogeneous ontologies. Organizations can reuse inter-organizational ontologies and their associated knowledge if ontology mismatches are reconciled. Mapping, merging and integration are three major kinds of ontology mediations. However, not all of these methods can be applied in a development process. Mapping is not an appropriate way to reuse inter-organizational ontologies as mapping only creates alignments between two ontologies by identifying semantic overlaps in
concepts, relationships, ontological axioms and logical statements. The goal of any ontology development methodology is to produce an ontology or a set of ontologies, not alignments between two ontologies.

Both merging and integration can be used to build a new ontology although each of these methods has its own distinctive characteristics. In merging, the resulting ontology usually contains a set of more general knowledge because the knowledge of a particular subject from two or more source ontologies is generalized, extended and combined together. In integration, an ontology is selected to integrate either directly into an ontology development process if no other suitable ontology is available or into an ontology development process with other chosen ontology(ies). For example, if an IT support domain help desk wants to build an ontology related to technical problems in the area of hardware, software, network and other IT administrative issues, it only needs to identify and reuse suitable modules from ontologies that contain the above knowledge. Figure 4.1 shows that knowledge related to hardware and IT administrative issues is available in Ontology 1 and 2 respectively, thus the help desk can reuse these particular modules from Ontology 1 and 2 in its own ontology building process for the development of technical knowledge specific ontology. Figure 4.1 also shows the resulting ontology after the integration of modules from two ontologies. However, the ontology development process needs to be continued as there is no suitable knowledge in the area of software and network available. As a result, the help desk is required to develop its own knowledge in those areas.
Although both integration and merging methods are capable of developing a new ontology, there is very limited information on how to select merging or integration method in an ontology development process. Whether to apply integration or merging in the development process should be considered carefully because selecting an incorrect method may require the development team to use an alternative method to develop the ontology again. Hence, the development cost may increase. To address this gap, this research proposes a set of guidelines for the selection of ontology integration and merging in the ontology development process.

First, both merging and integration can be performed when more than one ontologies are available. However, merging should not take place if only one suitable ontology is available. This ontology should be used to integrate into the ontology development process. Second, merging is a suitable candidate reuse method if the majority of the concepts and relations from source ontologies can be used to rebuild a new ontology. Alternatively if only one or a few portions of the ontologies can be reused from each
source ontology, then integration is more appropriate. Third, merging is a more appropriate method to develop top-level ontology; the resulting top-level ontology can be used to provide root terms that can be linked in existing ontologies. Merging is also an appropriate method to develop an ontology that aims to include every possible concept and relation from multiple source ontologies. In contrast, integration should be used to develop an ontology that contains a specific type of domain knowledge by taking the required knowledge modules from source ontologies. A summary of selection guidelines is shown in Table 4.2.

Table 4.2 Summary of Definition and Selection Guidelines of Ontology Integration and Merging

<table>
<thead>
<tr>
<th>Definition</th>
<th>Merging</th>
<th>Integration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Build an ontology in one subject by unifying two or more different ontologies on that subject</td>
<td>Build an ontology in one subject by reusing one or more ontologies in different subjects</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Is it Possible to Identify Source Ontology from the Resulting Ontology?</th>
<th>Not possible</th>
<th>Possible</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Selection Guidelines</th>
<th>Merging</th>
<th>Integration</th>
</tr>
</thead>
<tbody>
<tr>
<td>• More than one source ontologies are available</td>
<td>• One or more source ontology is available</td>
<td></td>
</tr>
<tr>
<td>• Majority of concepts and relations from each source ontology can be reused</td>
<td>• Only one or a few portions from each source ontology can be reused</td>
<td></td>
</tr>
<tr>
<td>• Ideal method to develop top-level ontologies</td>
<td>• Ideal method to develop ontologies that contain domain specific knowledge</td>
<td></td>
</tr>
<tr>
<td>• Ideal method to develop an ontology that includes every possible concepts and relations from source ontologies</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The above discussions demonstrate ontology integration is a more suitable mediation method to be used in the development of a domain specific ontology. The reason is integration can be used to develop an ontology that contains a specific type of domain knowledge by taking the required knowledge modules from source ontologies. However, little research has been conducted in ontology integration after the emergence of an integration methodology developed by Pinto and Martins (2001a). Pinto and Martins’s methodology comprises several activities to perform ontology integration which include feasibility study, domain identification, candidate ontologies identification, analysis and selection, application of integration operations and resulting ontology analysis. This integration methodology has two major limitations in its design: 1) the methodology only focuses on how to find, select and integrate existing ontologies but fails to recognize integration itself is part of an
ontology development process, and 2) the methodology also fails to be supported by any form of semi- or automatic tools especially in finding suitable knowledge modules from a group of candidate ontologies.

Besides, the integration methodology fails to keep up with the rapid changing external environment. For example, the methodology includes an activity about how to find available candidate ontologies because formalized ontologies had been difficult to locate at that time. Nowadays, a lot of ontologies of various domains are publicly available in ontology libraries (d’Aquinn and Noy 2011). Another example is the emphasis of translation to convert ontologies to an appropriate ontology language in the methodology due to the diversity of ontology languages at the time when the methodology was developed. The adoption of ontology language translator has not been required after OWL became the W3C recommendation ontology language in 2004. Hence, this research proposes a new methodology called Methodology of Integration-oriented Ontology Development (MIOD) that reuses existing ontology as part of its development process to build a domain specific ontology. The methodology not only includes a detailed description of ontology integration but also a semi-automatic tool to support identification of suitable knowledge modules from various candidate ontologies. MIOD includes a list of activities that can help to streamline the ontology development process in a more effective and efficient manner.

4.2 A Proposed Methodology of Integration-oriented Ontology Development (MIOD)

In MIOD, the development process is divided into five phases: preparation, analysis, integration, implementation and evaluation, and maintenance as shown in Figure 4.2.
4.2.1 Preparation Phase

The purpose of this phase is to identify purpose, scope, potential users and domain experts for resulting ontology, to evaluate the feasibility of the ontology development project and to generate project schedule. Four activities are required to conduct in the preparation phase:

1. Identify purpose.
2. Identify domain experts and potential users.
3. Feasibility analysis.
4. Generate project schedule.

The phase begins by defining the purpose of building an ontology including identification of problems that the resulting ontology will be used. The development team also needs to identify a brief scope of the ontology to indicate what must or must not be included in the resulting ontology. Other than having a project manager, the development team should also consist of system analyst, ontology engineer and knowledge engineer. A list of potential users and domain experts who possess
knowledge and experience for knowledge contribution as well as ontology implementation and evaluation, and maintenance should be included too.

Feasibility analysis is an important activity for this phase. Three types of feasibility analysis are to be performed: technical, resources and financial feasibilities. The technical feasibility measures whether the proposed ontology is a practical solution among a list of other possible solutions. As ontology is always used as a component of an ontology-based application, the technical feasibility of the proposed ontology should be evaluated with the application in which the ontology will be integrated with. This feasibility can be evaluated based on performance, ease of use and efficiency of the system. Besides, the technical feasibility also evaluates time, cost and risk required to integrate the proposed ontology with the ontology-based application.

The resources feasibility evaluates if the current human and technical resources can support such a development. The financial feasibility determines if it is cost effective to continue develop the ontology by conducting an analysis of development costs and anticipated benefits. The last activity in this phase is to generate a project schedule which includes a set of activities and their tasks, estimated duration and number of human resources needed to complete each task.

### 4.2.2 Analysis Phase

The purpose of this phase is to generate and categorize key terms identified from a set of inspiration scenarios. The key terms are used to find suitable source ontologies for integration. There are seven phases in the analysis phase:

1. Develop a set of inspiration scenarios.
2. Conceptualization.
3. Evaluate validity and sufficiency of key terms.
4. Categorize key terms.
5. Identify candidate ontology(ies).
6. Evaluate concepts of each candidate ontology.
7. Identify source ontology and its knowledge module.

The first activity of this phase is to develop a set of inspiration scenarios and a set of verification questions initialized by a group of domain experts. The scenarios are used to find out the type of information the ontology needs to provide. Each scenario identifies a problem that needs to be addressed by the resulting ontology. At the same time, each scenario also provides a set of possible solutions to the problem and a list of verification questions that is used to justify if the resulting ontology consists of sufficient terminologies to answer the questions. The inspiration scenarios and verification questions can be generated through discussion among members of developments, and sampling organizational and external documents. Figure 4.3 shows a sample template that can be used to record verification scenarios. The template includes scenario name, actors (participants) and description as well as a list of verification questions.

![Figure 4.3 Sample Template of Inspiration Scenario](image)

The second activity is conceptualization and the major contribution should come from potential users and domain experts. They should be encouraged to discuss in an open and friendly environment without worrying about terms may overlap, relationships among the terms and properties the terms may possess in order to capture all possible key terms related to the inspiration scenarios.
The third activity requires the development team to verify validity and sufficiency of key terms generated from the conceptualization activity. The verification can be achieved by justifying scope and level of details of the key terms to ensure enough terminologies are provided to answer all verification questions. Verification and conceptualization will iterate until all key terms are identified within the domain of interest. The forth activity categorizes key terms into subdomains. For example, key terms of IT knowledge can be categorized into software, hardware, network and other IT-related knowledge.

The fifth activity requires the development team to identify candidate ontologies in the domain of interest. Candidate ontologies can be located in ontology libraries, for instance, Protégé Ontology Library (www.protegewiki.stanford.edu) and DAML Ontology Library (www.daml.org/ontologies). Alternatively they can be searched using semantic search engines such as Swoogle. Very often, some of the partnerships or strategic alliances are willing to share their organizational knowledge as well as ontologies. At this stage, any possible ontologies should be considered as candidates and only those that have no relationship with the domain of interested are eliminated.

The sixth activity is to evaluate concepts of each candidate ontology with the categorized key terms. To achieve this, evaluations are required to conduct in all relevant candidate ontologies for each category. For each key term in a category, the development team needs to conduct a matching test to evaluate every concept from each candidate ontology in order to find semantically identical concepts. A suitable source ontology for each category can be found by comparing the number of semantically identical concepts among all candidate ontologies.

Moreover, the level of distribution among semantically identical concepts in a candidate ontology is another important consideration when choosing source ontologies. Consider O1 in Figure 4.4 and O2 in Figure 4.5 which are two ontologies used to demonstrate how distribution of semantically identical concepts can determine the selection of a source ontology. Each node in the diagrams represents a concept where the unshaded node and shaded node are used to represent a non-
identical and semantically identical concept respectively. Both O1 and O2 have fourteen concepts in which five of them are semantically identical concepts. The only difference between O1 and O2 is the level of distribution of semantically identical terms in which identical concepts in O1 are distributed more loosely than O2. In this case, rather than selecting O1 as the source ontology, O2 with the semantically identical concepts that converge closely with each other is easier to partition and modify, thus making it easier to reuse in the development process.

Figure 4.4 Ontology (O1) Used to Demonstrate Distribution Level of Semantically Identical Concepts

Figure 4.5 Ontology (O2) Used to Demonstrate Distribution Level of Semantically Identical Concepts
Figure 4.6 Proposed Integration-oriented Candidate Ontology Evaluation System (ICOES)

Although the sixth activity can be done manually, such an activity can be performed semi-automatically to reduce human resources expenses. Hence, an Integration-oriented Candidate Ontology Evaluation System (ICOES) is proposed to allow the sixth activity to be carried out semi-automatically. As illustrated in Figure 4.6, the proposed ICOES takes a set of key terms for a particular category and relevant candidate ontologies as input. The ICOES is designed to compare each key term of a category with every possible concept in each relevant candidate ontology to check for semantically identical representation using a single or a combination of ontology matching techniques.

The proposed DISCOMT as presented in Chapter 3 provides a valuable reference point for choosing a single or combination of ontology matching techniques. Since the techniques described in the DISCOMT are designed to perform matching between two ontologies, they are required to undergo modifications in order to fit in the proposed ICOES. In general, the techniques need to be modified to take key terms (identified in conceptualization activity) and concepts from a candidate ontology as input for similarity checking, instead of taking concepts extracted from two ontologies. For instance, if the ICOES uses string-based technique to find identical concepts from candidate ontologies, the technique needs to be modified in order to take key terms directly from each category. In this way, the string-based technique will be able to find semantically identical concepts from the candidate ontologies based on these key terms.
Let us demonstrate the application of the ontology matching techniques in the ICOES using string-based and linguistic source matching techniques. String-based matching technique is used in example 1 and 2 as shown in Figure 4.7 and 4.8 respectively. In example 1, the string-based matching technique embedded in ICOES finds a concept (Network) in Ontology A that is identical to the key term “Network” from a particular list. As a result, an arbitrary matching score of one point is assigned to the pair of key term (Network) and the concept (Network).

<table>
<thead>
<tr>
<th>Example</th>
<th>Key Terms</th>
<th>Concepts</th>
<th>Top-Level Ontology</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Network</td>
<td><img src="image" alt="Network" /></td>
<td>Ontology A</td>
<td>1</td>
</tr>
</tbody>
</table>

Figure 4.7 Demonstration of Ontology Matching Techniques in ICOES (Example 1)

In addition to identical representation of a key term and a concept, string-based matching technique also computes similarity if a prefix or suffix is found in an ontology that matches a particular key term. In example 2 (see Figure 4.8), the string-based matching technique finds a concept (Broadband Network) in Ontology B that contains the suffix of “Network”. Thus, an arbitrary matching score of half a point is assigned to the pair of key term (Network) and the concept (Broadband Network).

Score can be assigned based on the level of similarity. The level of similarity is higher when an identical concept is found rather than a prefix or suffix is found. Therefore, the score assigned in example 2 is lower than the score assigned in
example 1 due to the fact that “Broadband Network” is not identical to but more specific than “Network”

<table>
<thead>
<tr>
<th>Example</th>
<th>Key Terms</th>
<th>Concepts</th>
<th>Top-Level Ontology</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>Computer</td>
<td>PC</td>
<td>Computer</td>
<td>0.5</td>
</tr>
</tbody>
</table>

Figure 4.9 Demonstration of Ontology Matching Techniques in ICOES (Example 3)

In examples 3 (see Figure 4.9), the ICOES adopts linguistic resources matching technique to determine if a key term and a concept are semantically identical. Here, the technique uses domain specific thesauruses or top-level ontologies to derive a list of synonyms, hyponyms and hypernyms for every key term in a category. A score is assigned if a concept has linguistic relationship with a key term. In this example, an arbitrary score of 0.5 is assigned to the pair of concept (Computer) and key term (PC) even though “PC” is not the same as “Computer”. The score is given based on the fact that PC is a hyponym of Computer deduced from the top-level ontology.

After the matching process is completed, the proposed ICOES will generate an overall evaluation result for each pair of category and candidate ontology. The evaluation result should contain maximum, minimum and average score for each key term. Minimum and maximum scores are the minimum and maximum matching score attained for a key term against all concepts in the candidate ontology. Scores of each key term are aggregated to calculate an average matching score for a key term by dividing the aggregated matching score using the number of concepts that are semantically identical to the key term. In addition, the ICOES will also include a coverage score to indicate the average maximum similarity of all key terms.

The proposed ICOES also consists of a concept distribution counter that is used to measure the distribution level of matching concepts in the ontology. The counter views each candidate ontology as a graph with nodes (concepts) and edges (relationships) to represent the taxonomy of the ontology. For each candidate ontology, the counter has to find the shortest path (that is minimum number of nodes) for each pair of concepts that have been identified as semantically identical.
representations to at least one key term. To achieve this, the counter has to traverse all of the possible paths between the pair of concepts to find the total number of nodes (concepts) that are not semantically identical to any key terms in the matching result. By doing so, the counter is able to show the level of distribution for each candidate ontology by calculating the average number of unmatched nodes for each path. At the end of the matching process, the proposed ICOES also generates a distribution score to indicate the level of distribution for each pair of category and candidate ontology. Thus, the development team is able to find a suitable source ontology that can be reused in the category based on the matching scores, concept distributing scores and other evaluation criteria such as quality of documentation of the candidate ontologies and language used to implement the ontologies.

The seventh activity is to choose a set of source ontologies from the group of candidate ontologies that can be reused in the development process based on the results generated from the sixth activity. The source ontologies should be the ones that exceed the predefined minimum number of semantically identical concepts (known as threshold) and consist of the highest number of identical semantically concepts in a particular category. In addition, the source ontologies should also be the ones consisting of semantically identical concepts that converge closely with each other. In other words, the source ontology should be the one that has the highest coverage and distribution scores. The seventh activity also requires to identify the appropriate knowledge modules from each source ontology. Using key terms, the development team is able to find semantically identical concepts in the source ontology and partition the most appropriate knowledge modules for each category.

4.2.3 Integration Phase

The purpose of this phase is to evaluate various knowledge modules from source ontologies and integrate them into one integrated ontology. There are two activities in the integration phase:

1. Evaluate the quality of knowledge modules.
2. Integrate knowledge modules into one ontology.
The first activity requires the development team to ensure any knowledge modules from the source ontologies are accurate, precise, simple, clear, concise, correct and complete. To achieve, each knowledge module is required to measure its completeness and way of structuring concepts using the set of categorized key terms, inspiration scenarios and verification questions. Any insufficiency may require the module to undergo some kinds of modification in names and definitions of concepts and relations, and documentation such as:

- Removal of a concept, relation, name/definition from a concept/relation or part of documentation.
- Introduction of a new concept, relation, name/definition for a concept/relation or additional content for documentation.
- Specialization of more specific concepts from a general concept.
- Generalization of a more general concept from a group of specific concepts for classification purpose.
- Standardization of a name for a concept/relation.

Modifications should only be used parsimoniously because a simple modification may cause ripple effect across an ontology. In some cases, it may be more cost effective for the development team to choose the next best ontology from a group of candidate ontologies or to build its own knowledge module. Once the quality of knowledge modules is confirmed, second activity of this phase can be performed. This includes finding a connection point for each module and building a basic ontology to connect with knowledge modules through these connection points.

### 4.2.4 Development

The development team is responsible for using key terms in unused categories to build knowledge modules as part of the integrated ontology. Unused categories are these categories identified in the analysis phase but fail to find suitable source ontologies for integration. There are three activities in the development phase:
1. Use top-down or bottom-up technique to classify key terms in unused category.
2. Build knowledge module for unused category.
3. Assemble all unused key terms to the ontology

In the first activity, the development team is required to use top-down or bottom-up technique to classify key terms for each unused category. Top-down technique identifies the most general key terms followed by the more specific ones, so the most general terms are placed in the first level, the more specific in the second level and so on. In contrast, bottom-up technique locates the most specific key terms followed by the general ones. After classification, each category possesses the most specific terms in the first level and the least specific in the last level.

In the second activity, knowledge modules for unused categories are built by adding relevant relations among key terms that are semantically related. The third activity requires the development team to assemble all knowledge modules for unused categories back to the integrated ontology.

### 4.2.5 Implementation and Evaluation Phase

The purpose of this phase is to formalize the integrated ontology and to ensure the integrated ontology satisfies all evaluation requirements. The implementation and evaluation phase consists of two activities:

1. Encode integrated ontology using a formal representation language.
2. Evaluate integrated ontology.

The first activity requires the ontology completed in the development phase be implemented using a formal representation language ranging from RDF to OWL, depending on the level of expressiveness.

The second activity requires the integrated ontology to undergo two types of evaluation. Firstly, the ontology must be able to answer all verification questions
recorded in every inspiration scenario. Finally, the ontology must be evaluated in an application environment either in an actual system or a prototype and the potential users are required to provide feedbacks based on the usability of the ontology. Deficiencies or feedbacks should be reviewed and the development team may need to return to the analysis, integration, development, and implementation and evaluation phases until the ontology meets all evaluation requirements.

4.2.6 Maintenance Phase

An ontology needs to be updated regularly in order to maintain its validity. Maintenance phase consists of two activities:

1. Ontology Pruning.
2. Ontology Refining.

Ontology pruning is used to remove any invalid components of the ontology when components cannot be refined or are no longer needed. When the components cannot be refined, ontology refinement is needed. Ontology refinement includes extending existing components and adding new components to the ontology. Ontology refinement also includes the renewal of invalid components of the ontology including concepts, properties, relations and axioms. Level of consistency and accuracy, and content richness can be improved in the resulting ontology after ontology pruning and refinement.

Ontology may be required to modify based on changes in an ontology-based system. Ontology may also be required to change due to changes in internal and external business and political environment. Very often, the maintenance activity is performed by users who possess extensive knowledge in the domain of interest and use the ontology-based systems regularly to perform their daily task (but they may not possess high level of technical skill). When designing mechanism of ontology maintenance, it is very important to design a user interface that is appropriate for users as unfriendly user interface may hinder users from performing regular ontology maintenance.
4.3 Conclusion

Ontologies are the building block of the Semantic Web. Organizations and individuals have built ontologies of different domains that are sharable. Thus it is not impractical to build an ontology by reusing existing ontologies that are available publicly. This research has proposed and developed guidelines on how to select ontology merging or integration in a development process.

The analysis of ten significant ontology development methodologies has shown that the majority of methodologies have involved integrating or reusing ontology as one of their development activities rather than building an ontology from scratch. Literatures also show that building ontologies by reusing existing ontologies is more cost effective. However, these methodologies only provide a very limited discussion about how to perform integration in their ontology development processes.

Although Pinto and Martins (2001a) have developed an integration methodology consisting of several activities to perform ontology integration, the methodology only focuses on how to find, select and integrate existing ontologies but fails to recognize integration itself is part of an ontology development process. The integration methodology also fails to keep up with current environment as well as incorporate any forms of technology in its building process.

This research has addressed the above limitations by proposing the methodology called MIOD that includes preparation, analysis, integration, development, implementation and evaluation, and maintenance phases in its development process. MIOD provides a detailed description on how to incorporate and perform integration within an ontology development process including elicitation of key terms, identification of source ontologies and their knowledge modules, and application of ontology integration.

Following the MIOD, an ontology development team is required to identify purpose, scope, potential users and domain experts for resulting ontology, to evaluate
feasibility of the ontology development project and to generate project schedule. In the analysis phase, the team has to generate and categorize key terms identified from a set of inspiration scenarios. The key terms are then used to find suitable source ontologies for integration.

In integration phase, the team is required to evaluate various knowledge modules from source ontologies and integrate into one ontology. In the development phase, the team has to use key terms in unused categories to build knowledge modules. These knowledge modules are assembled to the integrated ontology. In implementation and evaluation phase, the team is responsible for formalizing the integrated ontology and ensuring that the integrated ontology satisfies all evaluation requirements. In the maintenance phase, the team has to update the ontology regularly in order to maintain its validity.

In addition, this research also proposes a semi-automatic tool called ICOES that can be used to find suitable source ontologies from a group of candidate ontologies using as a single or a combination of matching techniques as well as concept distribution counter. The proposed DISCOMT as presented in Chapter 3 provides detail guidelines for choosing a single or combination of ontology matching techniques. However, the techniques described in the DISCOMT are required to undergo modifications in order to fit in the proposed ICOES as they are originally designed to perform matching between two ontologies. The concept distribution counter is designed to measure the distribution level of matching concepts in the ontology. Based on the matching results generated by the ICOES, the team can select the most appropriate candidate ontologies as source ontologies.

In conclusion, MIOD provides tools and mechanisms that enables organizations to select appropriate sources ontology(ies) and integrate modules of the source ontology(ies) in an ontology development process.
As discussed in Chapter 1 and 2, knowledge created from external source plays a very important role in supporting organizational activities because employees often need to utilize organizational knowledge in their daily work. However, literature review shows that a majority of KM frameworks, practices and KMS were designed primarily to manage intra-organizational knowledge.

Let us consider the following scenario. Peter, who is an employee of University A, does not know how to install Microsoft Windows 7 on his MacBook Air. To learn how to do it, he accesses the KMS managed by the IT help desk of the university. The help desk KMS presents a self-help option for Peter to search the relevant knowledge on how to install the operating system. If knowledge regarding Windows 7 installation is available, then it will be retrieved. Otherwise he has the option to conduct a wider search such as searching the knowledge bases offered by the vendors. This process may be repeated using a number of knowledge bases until he finds the desired knowledge.

In the above scenario, Peter seeks help by visiting various KMS and knowledge bases until he finds the knowledge related to “installation of Windows 7 on a MacBook Air”. The organizational-based KM approach generally creates independent “knowledge islands” of the same domain without collaboration with other knowledge bases. The non-collaborative nature of “knowledge islands” presents several disadvantages for both knowledge workers and knowledge engineers. Knowledge workers need to spend time and efforts when searching for relevant knowledge from different KMS; knowledge engineers spend efforts and resources in creating and updating organizational knowledge stored in KMS. In some cases, the same knowledge is available in several KMS.

The non-collaboration nature of “knowledge islands” needs to be addressed. Very often, interoperation of heterogeneous KMS is difficult to achieve; this is due to business and knowledge management requirement vary among organizations.
Middleware has been proposed to improve interoperability of KMS. However, this is difficult to achieve due to the dynamic nature of KMS. In this research, ontology mediation methods are proposed to overcome the limitation of “knowledge islands”. Ontology incorporated in the KMS enables explicit knowledge to be annotated in the form of machine processable metadata. In this way, different organizations may possess their individual ontologies, however mediation methods can be applied to reconcile the underlying heterogeneity of ontologies. Thus, ontology and mediation approach make it possible for inter-organizational KMS to collaborate and be machine processable.

This research proposes to develop an ontology-based Collaborative Inter-organizational KM Network (CIK-Net) to allow KMS of various organizations to collaborate. The proposed CIK-Net is designed to provide a platform for organizations to access and retrieve inter-organizational knowledge of common domain. Ontology and its mediation methods are incorporated in the network to allow organizations to reuse the retrieved inter-organizational knowledge in supporting organizational KM process. In this research, inter-organizational knowledge is defined as a set of explicit knowledge formalized and created by other organizations.

According to Anderson et al. (1994) and Bradford et al. (2004), a network is defined as a set of two or more organizations that have an interdependent relationship without hierarchical control. The organizations in the network have relationships that are characterised by cooperation as well as competition (McLoughlin and Horan 2002). As such, each CIK-Net includes at least two organizations as long as they possess knowledge in the domain of the network. The participant organizations may already be engaged in partnership, not involved in any partnership or are direct or indirect competitors, however all organizations must agree to share all or a portion of formalized knowledge in the domain. Hence, the shared inter-organizational knowledge is reusable in a way that it can be retrieved by any participant organizations to support their own KM processes in terms of knowledge creating and dissemination.
When an organization recognizes the need for shared-knowledge management, it can invite other organizations to establish a domain specific KM network. When a network for a particular knowledge becomes mature, other organizations may choose to join the network instead of establishing a new network. Within the network, organizations must agree to annotate their knowledge using ontology. For organizations that do not possess ontologies or wish to join with their individual ontologies, the MIOD presented at Chapter 4 can be used. Each organization in the network must commit to mutual agreement to allow other participants to access an agreeable portion of ontology and its associated knowledge in its knowledge base. In addition, an organization can commit to more than one KM networks. Consider the following example as an illustration to the establishment of a KM network. A hospital can invite other hospitals and clinics to establish a CIK-Net on cancer prevention, diagnosis and treatment. It is worth noting that individual organization can still use their individual KMS and knowledge bases to manage the intra- and inter-organizational medical knowledge.

This chapter is organized as follows. Section 1 proposes an ontology mediation framework to assist organizations to select appropriate options of ontology mediation during the establishment of a KM network. An ontology-based CIK-Net is proposed in Section 3. Section 4 concludes the chapter.

5.1 **KM-network-based Ontology Mediation Framework**

Before we describe the proposed CIK-Net, the participant organizations need to make a few selections proposed in the KM-network-based Ontology Mediation Framework (KOM-Frame) regarding ontology mediation. The proposed KOM-Frame consists of four important decisions:

1. Whether to adopt top-level ontology or one-to-one as mapping approach for a KM network?
2. Whether to adopt merging, mapping and/or integration as the chosen mediation method?
3. Whether to perform mediation automatically or semi-automatically?
4. Whether to adopt single or multiple matching techniques for the chosen mediation method(s)?

These four decisions can be further distinguished as KM network-level decision and organizational-level decisions in which the first decision belongs to KM network-level and the remaining three decisions belong to organizational-level. The KM network-level decision must be decided by all participant organizations as a whole based on mutual benefits of the network, whereas organizational-level decision is determined by individual organization based on its own need.

The first decision is whether to adopt top-level ontology or one-to-one mapping as the KM network-level mapping approach. As this decision is on the KM network-level, organizations as a whole must select the most appropriate mapping approach for the benefit of the network through mutual agreement. The decision process should include a thorough assessment and discussion with respect to resources, expertise and frequency of modification among all organizations in the network.

The top-level ontology approach can only be applied to an environment where maintenance effort is minimal. Whenever a minor modification is performed in one of the ontologies in the network, the shared ontology used in the top-level ontology approach needs to be updated to include the additional entities and relationships. It is more appropriate to use one-to-one approach if frequent update is required in a dynamic business environment. Here, dynamic business environment refers to an environment where organizations need to modify their ontologies frequently. For instance, top-level ontology in the network is updated frequently to include additional concepts from ontologies of newly joined organization. Another instance of a dynamic business environment is shown in the current adoption of mobile apps. A recent research shows that the number of new Android apps increased from about 125000 to 350000 in 2011 (App Brain 2011). Thus, Android Apps Market will have to add relevant terminologies of these Android apps to its ontology frequently. Hence, any top-level ontologies that include concepts of ontology of Android Apps Market are to be updated frequently.
Organizations must ensure that they have resources (such as time and human resources) to build the shared ontology and maintain it. In addition, organizations must evaluate if they possess expertise to develop the ontology using ontology mediation. It is recommended to use one-to-one approach if resources and expertise are limited. Figure 5.1 shows the flow chart for the selection of one-to-one and top-level mapping approach.

Figure 5.1 One-to-one/Top-Level Mapping Approach Selection Flow Chart

The second decision is whether to adopt merging, mapping and/or integration as the chosen mediation method for each organization. Each organization can choose one or more methods based on its own needs. Ontology mapping is chosen as the mediation method if organizations want to retrieve inter-organizational knowledge using their KMS. Mapping allows semantically identical components of ontologies to be related at the same time allowing individual KMS to maintain its own characteristics.

Both ontology merging and integration can be used to build a combined ontology. However, merging should be selected if the resulting ontology were to include all possible views from multiple source ontologies. The resulting ontology should be
either a single ontology used to substitute individual source ontologies or a shared ontology used in the top-level ontology mapping approach. Merging can only be performed if two or more source ontologies are available. Ontology integration should be selected if the resulting ontology is built by reusing one or a few appropriate modules from each source ontology. Integration can only be performed if more than one source ontology is available, otherwise the resulting ontology should be built from scratch. Figure 5.2 shows the flow chart for the selection of mapping, merging and integration.

The third decision is whether to perform mediation automatically or semi-automatically. This decision has to be made for each mediation method selected in the second decision. Automatic mediation tools can be performed without human
intervention and are capable of support mediation on-the-fly. Unfortunately automatic tools are unable to detect and interpret concepts that do not have a close correlation. Moreover, it may also fail to handle any unforeseeable situations since the tool is designed to perform mediation under certain pre-defined circumstances. For example, a fully-automatic mediation tool may fail to find matching candidates if the matching condition is not pre-defined in its matching algorithm.

Mediation can also be performed semi-automatically which requires the support of automatic tools as well as human intervention. In general, semi-automatic mediation tools suggest a list of possible actions for ontology merging based on computational results of similarity but the final decision on choosing the most suitable action is left to users. Other forms of support provided by automatic tools include post-mediation verification, validation, evaluation as well as conflicts recognition and resolution. Figure 5.3 shows the flow chart for the selection of semi- and fully-automatic mediation.

Figure 5.3 Fully-/Semi-Automatic Mediation Selection Flow Chart
The final decision is whether to adopt single or multiple matching techniques for the selected mediation method(s). The DISCOMT presented in Chapter 3 provides a valuable reference point for choosing a single or combination of matching technique. In general, single matching technique can only handle elementary lightweight ontology that contains concepts and simple relationships.

When the majority of ontologies are heavyweight ontologies in the network, it is essential to choose multiple matching strategies. In the decision process, organization must also consider the level of time and human resources available for implementation. Multiple strategies generate more accurate result than single matching technique but it still depends on aggregation algorithm and cut-off point. The choice of aggregation algorithm and cut-off point are important in determining the level of matching accuracy. When choosing multiple strategies as its matching technique, organization must conduct a series of experiments with the purpose of finding a combination of multiple strategies, aggregation algorithm and cut-off point that can produce the most accurate result which resulting in more time and resources required to design and implement multiple matching techniques. Compare to single matching technique, multiple strategies are relatively more difficult to design and implement. Figure 5.4 shows the flow chart for the selection of single and multiple matching techniques.
5.2 The Proposed Collaborative Inter-organizational KM Network (CIK-Net)

In the proposed CIK-Net, organizations are required to use concepts and relationships in their organizational ontologies to explicitly represent the formalised knowledge. The reconcilability of ontology mediation allows participant organizations to reuse inter-organizational knowledge within the network. Under mutual agreement, organizations are permitted to retrieve inter-organization knowledge and the retrieved knowledge can be reused to support knowledge creation and dissemination of KM process.
Approaches of ontology mediation provide a practical way to create explicit knowledge in the CIK-Net using the method of combination. As described by Nonaka et al. (2001), combination is a method to create knowledge by merging and editing “explicit knowledge from multiple sources” into a new set of more comprehensive and systematic explicit knowledge. Combination can be performed to create explicit knowledge on both network and organizational level in the proposed CIK-Net. On the KM network level, ontology merging allows explicit knowledge of all inter-organizational ontologies in the network to be combined. This can be achieved by using semi- or fully-automatic merging tools to combine common views of every inter-organizational ontology in the network to create a shared ontology. By selecting top-level mapping approach as the designated mapping approach, the shared ontology enables inter-organizational ontologies to resolve conflicts and ambiguities based on the primitives of the shared ontology.

On the organizational level, organizations can also use the combination method to create a set of new explicit inter-organizational knowledge. When organizations do not possess any ontologies, ontology integration allows these organizations to reuse inter-organizational ontologies to build individual organizational ontologies. Another benefit of adopting integration is organizations can reuse inter-organizational knowledge associated with the source ontologies. Instead of creating their own knowledge from scratch, the organizations can store and reuse the inter-organizational knowledge directly in their individual knowledge repositories.

It is recommended to use the proposed MIOD presented in Chapter 4 to develop integrated ontologies in the CIK-Net. This is because MIOD adopts ontology integration as part of its development process in which appropriate tools and mechanisms are provided to support the process. In MIOD, ICEOS is designed to find appropriate source ontologies developed by participant organizations in the network. Mechanisms are provided to integrated relevant modules from the source ontologies build an integrated ontology. The MIOD can be applied to organizations that want to join the CIK-Net with a new ontology.
In the CIK-Net, ontology mediation can be used to support organizational and inter-organizational knowledge dissemination. To achieve this, the organizations need to use mapping tool to create mappings across two ontologies through identical concepts, relationships, ontological axioms or logical sentences. With the created mappings, organizational users can use search mechanisms to retrieve both organizational and inter-organizational knowledge. If users cannot find suitable organization knowledge in the organizational knowledge base, they can seek from external sources in the network as the established mappings allow one KMS to access KMS of other organizations to search for relevant knowledge.

Besides, it is also practical for real-time mapping to be performed. Whenever required knowledge is not available in the organizational KMS and inter-organizational knowledge bases through established mappings, the fully-automatic mapping tool is activated to look for and select semantically identical concepts and properties from other ontologies in the network. The automatic mapping tool is also responsible to establish mappings with the identical concepts and properties without human’s intervention. In this way, the organizational KMS is able to retrieve and deliver inter-organizational knowledge through mappings created real-time.

### 5.3 Conclusion

Traditionally, KM approaches have limitation of non-collaborative problem in which individual organization is not able to reuse inter-organizational knowledge. The ontology-based CIK-Net is proposed to address this limitation. Ontology approach allows participant organizations to represent organizational knowledge explicitly using machine processable representation in the network. The ontologies possessed by various participant organizations can cause ontology mismatches but the mismatches can be reconciled using ontology mediation methods. Thus, the participant organizations can access, retrieve and reuse domain specific inter-organizational knowledge by applying appropriate ontology mediation methods. The major contribution of the proposed CIK-Net is the retrieved inter-organizational knowledge can be reused to support organizational KM process in the area of knowledge creating and dissemination.
The application of ontology merging in the proposed CIK-Net enables explicit knowledge to be created through the combination method. The use of combination method to create explicit knowledge can be achieved to develop a shared ontology to be used for top-level mapping approach. The combination method can also be achieved by the application of ontology integration to create an integrated ontology for organization that wants to join a CIK-Net. The application of ontology mapping enables users to retrieve inter-organizational knowledge from external knowledge sources, thus making it possible for knowledge to be disseminated and reused throughout the network.

To establish a CIK-Net, KOM-Frame is proposed to assist organizations choosing appropriate ontology mediation options. The first selection is whether to adopt top-level ontology or one-to-one as mapping approach for a KM network. The second decision is whether to adopt merging, mapping and/or integration as the chosen mediation method, this is followed by considering whether to perform mediation automatically or semi-automatically. The final decision is whether to adopt single or multiple matching techniques for the selected mediation method(s).
Chapter 6: APPLICATION SCENARIO AND PROTOTYPE DEVELOPMENT

This chapter describes an application scenario for the ontology-based CIK-Net in an IT help desk environment. The application scenario shows how the KOM-Frame can be used to select appropriate ontology mediation options during the establishment of the CIK-Net. The application scenario will also be used to illustrate how explicit inter-organizational knowledge can be disseminated in individual organizational KM process through the establishment of mappings with inter-organizational ontologies.

The application scenario is used to illustrate how explicit inter-organizational knowledge can be created in individual organizational KM process. This is achieved by the application of MIOD to develop an IT knowledge ontology reusing existing inter-organizational ontologies. Furthermore, a prototype is developed to illustrate how users can retrieve the organizational and inter-organizational knowledge in CIK-Net and shows how ICOES can be used to find the appropriate source ontologies semi-automatically in MIOD.

This chapter is organized as follows. An application scenario of the ontology-based CIK-Net is described in Section 1. The discussion of the prototype design and development is presented in Section 2. Four illustrations will be presented to demonstrate the adoption of KOM-Frame, knowledge dissemination, mapping establishment and knowledge creation in Section 3. Section 4 concludes the chapter.

6.1 An Application Scenario

To demonstrate the application of the ontology-based CIK-Net, let us consider the following scenario. The IT help desk of University A (ITA) requires high level of knowledge on implementation issues of new software due to the delivery and implementation of new IT platform. ITA invites IT service provider and IT help desks of other organizations to establish a KM network of the new IT platform. Let us assume that IT help desk of University B (ITB) and University C (ITC), and IT
Solution Provider E (ITE) agree to join the network. After careful consideration, the four organizations have agreed to allow other participants to access an agreeable portion of ontology and its associated knowledge in the knowledge base.

The news of setting up the inter-organizational KM network in the domain of IT knowledge is widely spread among higher education institutes which include a newly established private university known as University D. University D is keen to provide excellent education and support to students but this university has limited IT budget. Although IT help desk of University D (ITD) wants to develop a KMS that contains IT knowledge, ITD itself has insufficient human and financial resources to support the KMS development, in particular in the process of knowledge creation and storage. ITD shows strong desire to join the network because it is able to reuse other participants’ ontologies and their associated knowledge base within the network. Through the participation of the KM network, ITD believes that the burden of KMS development can be eased particularly in the knowledge creation and storing phase. After several discussions with ITD, all existing participants of the KM network agree to accept ITD as a new member in the network. Table 6.1 shows a summary of characteristics of KMS and ontologies possessed by the KM network participants.
Table 6.1 Summary of KMS and Ontology Possessed by the KM Network

<table>
<thead>
<tr>
<th>Participants</th>
<th>KMS Characteristics</th>
<th>Ontology Characteristics</th>
</tr>
</thead>
</table>
| ITA          | Possess an IT knowledge KMS with three major components:  
               1) Web-based user interface  
               2) Knowledge base  
               3) Ontology | Possess a simple, lightweight ontology (Ontology1) that represents categories of technical knowledge and their characteristics in hardware, IT admin, software and other relevant areas. |
| ITB          | Possess an IT knowledge KMS with three major components:  
               1) Web-based user interface  
               2) Knowledge base  
               3) Ontology | Possess a simple, lightweight ontology (Extontology2) that represents categories of technical knowledge and their characteristics in hardware area. |
| ITC          | Possess an IT knowledge KMS with three major components:  
               1) Web-based user interface  
               2) Knowledge base  
               3) Ontology | Possess a simple, lightweight ontology (Extontology3) that represents categories of technical knowledge and their characteristics in web application area. |
| ITD          | Do not possess any KMS. | Do not possess any ontology. |
| ITE          | Possess an IT knowledge KMS with three major components:  
               1) Web-based user interface  
               2) Knowledge base  
               3) Ontology | Possess a simple, lightweight ontology (Extonology1) that represents categories of technical knowledge and their characteristics in Microsoft software area. |

6.2 Design and Development of the Prototype

We have developed a prototype to demonstrate the above application scenario based on the Ontology-based CIK-Net framework presented in Chapter 5. The prototype is developed using a web-based programming language called Java. Java is chosen because of its platform independent capability that allows the same Java program to execute on other platforms without recompilation and modification. The logical design of the prototype is shown in Figure 6.1. The prototype contains five major components to allow the functionalities to be carried out. These include a web browser, ontology engineering tool, knowledge browsing tool, ontology evaluation tool and ontology repository. The first two components are located in users’ desktops (clients) to provide a medium for users to interact with the prototype and functions for ontology matching and mapping respectively. The rest of the components are implemented in the server where repository is used for storing ontologies and their associated knowledge, ontology browsing tool is used for traversing organizational and inter-organizational ontology and ontology evaluation tool is used to evaluate
candidate ontologies in the ontology integration process. Source codes of the prototype are included in Appendix A

![Figure 6.1 Logical Design of the Prototype](image)

6.2.1 **Prototype Development Environment**

Figure 6.2 depicts the physical design of the prototype. Users can access the user interface by entering the URL of the prototype ([http://182.50.129.42/SemanticHelpDesk/settings](http://182.50.129.42/SemanticHelpDesk/settings)) on their browsers (preferably using Mozilla Firefox) which leads to all functionalities of the prototype except the functionality provided by NeOn Toolkit (and its alignment plugin). NeOn Toolkit is a state-of-the-art, open source multi-platform ontology engineering environment that designed to provide comprehensive support for the ontology engineering life-cycle (Haase et al. 2008). As NeOn Toolkit is not a web-based application, it cannot be installed on the server, thus making it impossible to be accessed through the web user interface. As a result, the NeOn Toolkit has to be installed in local client computers and any mappings created must be uploaded to the server of the prototype through a web user interface manually.
The NeOn toolkit and its alignment plugin are responsible for identifying matching concepts from two ontologies, creating mappings in the form of OWL axioms and trim mappings by applying threshold to the matching results semi-automatically (Chan and Euzenat 2010). Users can choose any one of the eight matchers provided by the alignment plugin to perform ontology matching. The description of the matchers are summarised in Table 6.2.

Table 6.2 Summary of Matchers in Alignment Plugin of NeOn Toolkit (Fu 2011)

<table>
<thead>
<tr>
<th>Matcher</th>
<th>Description of the Matcher</th>
</tr>
</thead>
<tbody>
<tr>
<td>NameAndPropertyAlignment</td>
<td>Comparison of resources based on their names and properties</td>
</tr>
<tr>
<td>StrucSubsDistAlignment</td>
<td>Comparison of resources based on substring distance of their name and aggregated these distances with property differences</td>
</tr>
<tr>
<td>ClassStructAlignment</td>
<td>Comparison of resources and their structures</td>
</tr>
<tr>
<td>NameEqAlignment</td>
<td>Comparison of the equality of resources names</td>
</tr>
<tr>
<td>SMOANameAlignment</td>
<td>Comparison of resources using edit distance</td>
</tr>
<tr>
<td>SubsDistNameAlignment</td>
<td>Comparison of resources based on substring distance of their names and properties</td>
</tr>
<tr>
<td>EditDistNameAlignment</td>
<td>Comparison of resources names using edit distance</td>
</tr>
<tr>
<td>StringDistAlignment</td>
<td>Comparison of resource names regardless of their types</td>
</tr>
</tbody>
</table>

Servlets and JavaServer Pages (JSP) are used to allow the user interface to be displayed dynamically on the web browser. Servlets are portable between servers and operating systems and are designed to handle Hypertext Transfer Protocol (HTTP) requests (Hall 2000; Hunter and Crawford 2001). JSP extends servlet technology and allows Java codes to be embedded in special tags so that they can combine with
regular HTML on the same page. While HTML is used to handle static content, JSP is responsible for the display of dynamic content on the web user interface. In this prototype, Apache Tomcat server is used as a servlet server to provide facilities for running servlets. During the runtime, the Apache Tomcat server turns the JSP into a Java servlet using a Jasper compiler and the resulting servlet is further converted into byte code and run on the server.

Other than acting as a servlet container, the Apache Tomcat server also serves as an application server for OpenRDF Sesame and Semantic Help Desk (the prototype). An extended view of the server is shown in Figure 6.3. Within the prototype, OpenRDF Sesame is used as a knowledge repository for ontologies and its associated knowledge. OpenRDF Sesame, an open source Java framework for storing and queries of RDF data, is fully extensible and configurable for storage mechanisms, inferencers, RDF file formats, query result formats and query languages (Aduna 2011).
The OpenRDF Sesame can possess more than one ontology repository and each repository must have two storages, namely static storage and deducted storage. The static storage is designed to keep the uploaded ontologies whereas the deducted storage is for storing ontologies and their deducted statements. The deducted statements stored in deducted storage play an important role in inferring indirect relations during the knowledge retrieval process. For example, if A is a subclass of B and B is a subclass of C, then the deducted statements infer that A is also a subclass of C. When an ontology is added to the prototype through the web user interface, the ontology is loaded into the static storage. In order to keep two storages synchronized, the data in the static storage is transformed into deducted models to replace the original data housed in the deducted storage after each modification in the static storage.

The prototype supplies concepts for the selections in the drop boxes on the user interface by traversing in the intra-organizational ontology stored in the deducted
storage. When the prototype needs to retrieve concepts from a set of more complicated relationships involving inter-organizational ontologies, it needs to rely on the ontologies and their deducted statements stored in the deducted storage. Besides, OpenRDF Workbench is also included in the server in which users can query, update and explore repositories of the OpenRDF Sesame via a web browser.

The server also contains components that implement user interface and server-side logic of ontology browsing and evaluation tools as well as for uploading ontologies into OpenRDF Sesame (see Figure 6.3). The above three functionalities cannot be performed if appropriate Application Programming Interface (API) is not included to enable various components to communicate with each other. As illustrated in Figure 6.3, this prototype makes use of three APIs: 1) Align API is a set of abstractions for expressing, accessing and sharing ontology alignments in which the implementation consists of a library of matchers (David et al. 2010), 2) Jena API is a language independent interface that provides a consistent programming interface for ontology application development (Dickinson 2009), and 3) OpenRDF Repository API is an access point for Sesame repository offering a variety of data querying and updating methods (Aduna 2011).

Figure 6.4 shows the components of the prototype “Semantic Help Desk” interacting with the APIs. The “Add Ontology” function uses OpenRDF API to access repository of OpenRDF Sesame and it also uses Jena API to transform uploaded ontology to deducted model using OWL reasoner. The “Browse Ontology” function uses OpenRDF API to access repository of OpenRDF Sesame; this API is also adopted to retrieve knowledge from organizational and inter-organizational ontology housed in deducted storage.
The “Evaluate Ontology” function uses Align API to find identical concepts between candidate ontology and a set of user-defined keywords. Align API supports a number of matchers to locate identical concepts. The matcher implemented in the prototype is the EditDistNameAlignment matcher that exploits edit distance to find identical concepts. Edit distance calculates the amount of difference or similarity between two concepts by measuring the number of steps required to transfer a concept to another. To measure the distribution level of matching concepts, this function uses findShortestPath method of OntTools class provided in Jena API to calculate the shortest path between two concepts in ontology. The findShortestPath method calculates the shortest path by comparing the number of nodes of all possible paths between a starting and an end node.

In the prototype, the ontologies are developed using OWL to represent various categories of technical knowledge types and their characteristics. The knowledge types and characteristics are used to support the user interface in which users can choose to describe and identify the relevant IT knowledge. OWL builds on RDF and RDF Schema and adds more vocabulary for describing properties and classes: among others, relations between classes, cardinality, equality, richer typing of properties, characteristics of properties and enumerated classes (McGuinness and Harmelen 2004). The RDF uses XML as interchange syntax to provide a lightweight ontology.
system to support the exchange of knowledge on the Web (Antoniou and Harmelen 2004, Klyne and Carroll 2004).

### 6.2.2 Ontology Design

In this example of application scenario, IT organizational knowledge repositied within the KMS of each participant organization must be categorised using concepts of its own organizational ontology in terms of knowledge type and characteristic. For instance, a password reset guideline is categorised using “Help Desk Enquiry”, “IT Admin Issue”, “Password Issue” and “Reset Password Guidance”. To demonstrate the functionalities of inter-organizational knowledge retrieval and candidate ontology evaluation in the scenario, four ontologies are developed; these include Ontology1 for ITA, Extontology1 for ITE, Extontology2 for ITB and Extontology3 for ITC. These ontologies are created using inspiration design approach in which the resulting ontologies are built using individual imagination, creativity and personal views about the domain of interest to meet the recognized need (Holsapple and Joshi 2002).

![Figure 6.5 Four Categories of Help Desk Enquiry in Ontology1](image)

Figure 6.5 depicts the first two levels in Ontology1 that represents an organizational ontology for ITA. The ontology includes “Help Desk Enquiry” as its superclass and “Hardware Problem”, “IT Admin Issue”, “Software Problem” and “Other Problem” as the subclasses of “Help Desk Enquiry”. These four subclasses are extended to represent the four main categories of ITA as shown in Figures 6.6 (Hardware Problem), 6.7 (IT Admin Issue), 6.8 (Software Problem) and 6.9 (Other Problem) respectively. For instance, “Hardware Problem” in Figure 6.6 is further developed into “Non-Standard Hardware Problem” and “Standard Hardware Problem”
subclasses. The expansion of subclasses is continued until a complete set of concepts is developed to describe knowledge types and characteristics in ITA.

Figure 6.6 Extended View of Ontology1 in Hardware Problem Class
Figure 6.7 Extended View of Ontology1 in IT Admin Issue Class

Figure 6.8 Extended View of Ontology1 in Software Problem Class
Extontology1 is designed to classify Microsoft Software knowledge for ITE (see Figure 6.10). There are two subclasses: “Office” and “Windows” under the superclass “Microsoft Knowledge”. Extontology2 is developed to describe hardware knowledge for ITB (see Figure 6.11). Extontology2 has “Product” as its superclass and “Mobile Device” and “Visual Device” as subclasses of “Product”. Extontology3 is used to classify knowledge regarding web application errors for ITC (see Figure 6.12). The superclass “Web Application Error” has “HTTP Error”, “SMTP Error”, “POP3 Error” and “FTP Error” as its subclasses. Further expansion of the above subclasses is required until complete sets of concepts are developed to describe knowledge types and characteristics in ITE, ITB and ITC. Ontology1, Extontology1, Extontology2 and Extontology3 in OWL format are included in Appendix B.
6.2.3 Design of the Prototype

There are two major design issues in the development of the prototype. First, the prototype is required to support inter-organizational knowledge reuse in the KM process of knowledge dissemination. In practice, the prototype should be able to
allow users from ITA, ITB, ITC, ITD and ITE to retrieve inter-organizational IT knowledge from each other within the network if the required knowledge is not available in their own knowledge base.

As summarized in Table 6.1, each participant organization possesses a simple hierarchical lightweight ontology so that each piece of organizational knowledge is categorised by concepts of the organizational ontology. Ontology mismatches occur as the ontologies were developed by different organizations. According to KOM-Frame, ontology mapping is an appropriate method to reconcile ontology mismatches if the reconciled ontology is used to retrieve inter-organizational knowledge. Therefore, the prototype needs to possess mapping facility to reconcile ontology mismatches. The prototype also needs to be designed to allow users to retrieve organizational knowledge from its own knowledge base as well inter-organizational knowledge from external knowledge base through the established mappings. To demonstrate this functionality, the prototype is set up only for ITA to retrieve inter-organization knowledge from other participants; that is ontology of ITA is used as organizational ontology and ITB, ITC and ITE are the inter-organizational ontologies.

Secondly, the prototype is also used to demonstrate how it supports inter-organizational knowledge reuse in the knowledge creation phase of the KM process. Assume that ITD decides to use MIOD to integrate explicit inter-organizational knowledge in terms of ontologies and its associated knowledge in its ontology development process. Therefore, the prototype needs to provide functionality that is identical to ICOES. The prototype needs to include facilities to evaluate a group of candidate ontologies within the network using concept distribution counter as well as a single or a combination of matching techniques.

While the matching techniques is designed to compare key terms with every possible concept in each candidate ontology to check for identical representation, the distribution counter is used to measure the distribution level of matching concepts in the candidate ontologies. The prototype takes a set of keywords identified in the analysis activities as input and produces evaluation results that include the matching
and distribution results for each candidate ontology. The functionalities of the prototype are summarised in Figure 6.13.

![Figure 6.13 Functionalities of the Prototype](image)

### 6.3 Illustration of the CIK-Net in the Application Scenario

In this section, four illustrations are conducted to demonstrate the application of CIK-Net in the scenario:

1. Illustration of KOM-Frame.
2. Illustration of Knowledge Dissemination.
3. Illustration of Mapping Establishment.
4. Illustration of Knowledge Creation.

#### 6.3.1 Illustration of KOM-Frame

Before the establishment of the CIK-Net, the four organizations need to make four important selections based on the KOM-Frame, as presented in Chapter 5, in which a decision is made at the KM network-level and the remaining three decisions are made at the organizational-level.

The first decision is whether to adopt top-level ontology or one-to-one mapping as the KM network-level mapping approach. In this example, the four organizations
reach a mutual agreement not to adopt top-level ontology as the network-wide mapping approach. This decision is based on the fact that there will be more organizations which may wish to join the network in the future, therefore a shared ontology built for the top-level ontology mapping approach may need to undergo a series of reconstructions. In addition, ontologies of the four organizations will need to be updated whenever new IT applications, services and products are deployed in the future and the updates may be fairly regular due to the rapid product life cycle in current business environment. It is also found that these organizations do not have sufficient expertise and resources to build and maintain a shared ontology. Thus, the organizations decide to use one-to-one mapping approach. They have agreed to review the mapping approach at a later date when the KM network is stabilised.

As the rest of the decisions are at the organizational-level, thus individual organizations can make their own decisions based on individual organizational environment. In this example, assume that ITA chooses to adopt ontology mapping as its mediation method because ITA wants to make use of mappings established among inter-organizational ontologies to allow users to retrieve inter-organizational knowledge though its KMS. Ontology mapping is an appropriate ontology mediation method if the reconciled ontologies are used for retrieving inter-organizational knowledge. Besides, ITA possesses its own ontology and it does not need to build a new ontology using merging or integration method.

The third decision ITA makes is to perform ontology mapping semi-automatically as ITA does not need to execute real-time mapping. Another consideration is inter-organizational ontologies may become inconsistent as a result of frequent updating. Hence, this may create invalid mappings if automatic mapping is performed without human intervention. IT equipment or IS can be used inappropriately if users follow wrong instructions retrieved through invalid mappings created by a fully-automatic tool.

For the final decisions, ITA decides to adopt a single matching technique used to find identical concepts. There are two main reasons behind such a decision: 1) ITA has no time and human resource available to develop and test multiple matching strategies, aggregation algorithm and cut-off point, and 2) ITA has examined ontology of other
participating organizations and found that all the organizations possess simple and lightweight ontologies, so a single matching technique is capable of accurately identifying matching concepts within the network. However, ITA has agreed to constantly review this decision to ensure the matching technique is performed in a correct manner.

Consider the case of ITD which will make three important organizational-level decisions using the KOM-Frame. Firstly, ITD decides to use ontology integration method to build an integrated ontology as it currently does not possess any ontology. Ontology integration method is selected because ITD does not require its resulting ontology to include all possible concepts and relationships from source ontologies. Instead ITD only wants to be able to include appropriate knowledge modules from source ontologies. An important factor to consider is whether appropriate source ontologies can be found during the integration process. ITD also decides to adopt ontology mapping as the other mediation method. The decision is made due to the fact that ITD users will be able to retrieve inter-organizational knowledge through mappings established between its ontology and other inter-organizational ontologies.

Secondly, ITD decides to perform ontology mapping semi-automatically. The decision is based on two reasons: 1) ITD does not need to perform real-time mapping, and 2) inappropriate mappings may be created by fully-automatic mapping method due to frequent update of inter-organizational ontologies. The inappropriate mappings may lead to misuse of IT equipment and application programs if users follow inappropriate instructions retrieved through inappropriate mappings. ITD also decides to perform ontology integration semi-automatically. There are two reasons behind this decision: 1) ITD does not need to perform real-time integration, and 2) unforeseeable situations may happen due to complexity of integration process in which each candidate ontology has its own language, structure, scope and granularity. Human effort is required to intervene to make appropriate decisions in the integration process.

Thirdly, ITD also decides to adopt a single matching technique used to find identical concepts as it does not have time and human resource to develop and evaluate multiple matching strategies, aggregation algorithm and cut-off point. Besides, it is
appropriate to use a single matching technique to identify matching concepts as the inter-organizational ontologies that the participant organizations possess are all simple and lightweight ontologies. A summary of organizational-level decision of ITA and ITD is presented in Table 6.3.

Table 6.3 Summary of Organizational-level Decision of ITA and ITD

<table>
<thead>
<tr>
<th>Decision</th>
<th>ITA</th>
<th>ITD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mediation Method</td>
<td>1) Mapping is selected because:</td>
<td>1) Mapping is selected because:</td>
</tr>
<tr>
<td></td>
<td>• Reconciled ontologies can be used to retrieve inter-</td>
<td>• Reconciled ontologies can be used to retrieve inter-</td>
</tr>
<tr>
<td></td>
<td>organizational knowledge.</td>
<td>organizational knowledge.</td>
</tr>
<tr>
<td></td>
<td>2) Integration is selected because:</td>
<td>2) Integration is selected because:</td>
</tr>
<tr>
<td></td>
<td>• ITD wants to build an ontology for its KMS reusing one or</td>
<td>• ITD wants to build an ontology for its KMS reusing one or</td>
</tr>
<tr>
<td></td>
<td>more portions from each source ontology.</td>
<td>more portions from each source ontology.</td>
</tr>
<tr>
<td>Fully-/Semi-automatic</td>
<td>Semi-automatic is selected for mapping because:</td>
<td>1) Semi-automatic is selected for mapping</td>
</tr>
<tr>
<td>Mediation</td>
<td>• Not required to perform real-time mapping.</td>
<td>• Not required to perform real-time mapping.</td>
</tr>
<tr>
<td></td>
<td>• Unforeseeable situation may occur due to frequent update of</td>
<td>• Unforeseeable situation may occur due to frequent update of</td>
</tr>
<tr>
<td></td>
<td>ontologies.</td>
<td>ontologies.</td>
</tr>
<tr>
<td></td>
<td>2) Semi-automatic is selected for integration</td>
<td>2) Semi-automatic is selected for integration</td>
</tr>
<tr>
<td></td>
<td>• Not required to perform real-time mapping.</td>
<td>• Not required to perform real-time mapping.</td>
</tr>
<tr>
<td></td>
<td>• Unforeseeable situations may occur due to the complexity of</td>
<td>• Unforeseeable situations may occur due to the complexity of</td>
</tr>
<tr>
<td></td>
<td>integration process.</td>
<td>integration process.</td>
</tr>
<tr>
<td>Single/Multiple</td>
<td>Single matching technique is selected because:</td>
<td>Single matching technique for mapping and integration because:</td>
</tr>
<tr>
<td>Matching Technique(s)</td>
<td>• No time and human resource available to develop and test</td>
<td>• No time and human resources available to develop and test</td>
</tr>
<tr>
<td></td>
<td>multiple matching strategies, aggregation algorithm and cut-</td>
<td>multiple matching strategies, aggregation algorithm and cut-</td>
</tr>
<tr>
<td></td>
<td>off point.</td>
<td>off point.</td>
</tr>
<tr>
<td></td>
<td>• All ontologies in the KM network are simple and lightweight.</td>
<td>• All inter-organizational ontologies are simple and lightweight.</td>
</tr>
</tbody>
</table>

6.3.2 Illustration of Knowledge Dissemination

The prototype can be used to illustrate how explicit knowledge can be disseminated in ITA. To retrieve organizational and inter-organizational knowledge, consider the case when users of ITA select “Retrieve Knowledge” function in the navigation
menu (see Figure 6.14) of the web user interface of the prototype (http://182.50.129.42/SemanticHelpDesk/). The knowledge retrieval function on the web user interface of the prototype enables general ITA users to obtain the most appropriate IT knowledge based on their needs.

![Sample Screen of Navigation Menu](image)

To view the knowledge, users from ITA have to describe the knowledge types and their characteristics by choosing the related vocabularies in which the vocabularies are defined in the organizational ontology. If the required knowledge is available in the organizational knowledge repository, it will be displayed. If knowledge is unavailable in the organizational knowledge repository, the system will retrieve the knowledge from other organizations through the mappings established in the inter-organizational ontologies.

1 Users are required to login using “admin” and “1234” as ID and password
Let us consider two examples to illustrate the knowledge retrieval function. In the first example, Peter from ITA wants to obtain information about how to reset password. He describes the knowledge by selecting “IT Admin Issue”, “Password Issue” and “Reset Password Guidance” in three selections (see Figure 6.15). The system retrieves and displays the knowledge as it is available in the organizational repository (see Figure 6.16).
In the second example, Peter wants to retrieve knowledge related to permanent failure in FTP server. Thus, he describes the knowledge by choosing “Other Problem”, “Website Problem”, “Non-Enterprise Website Problem” and “FTP Problem” supplied by the organizational ontology (see Figure 6.17). Since the required knowledge is not available in ITA’s knowledge repository, the system examines the established mappings to ontologies of ITB, ITC and ITE. After examination, it is found that there is a mapping between “FTP Problem” in Ontology1 and “FTP Error” in Extontology3 of ITC. In this case, the system continues with the selection that includes “Persistent Transient Failure” and “Permanent Failure” obtained from Extontology3 (see Figure 6.18). After Peter has selected “Permanent Failure” in the next selection, the inter-organizational knowledge is retrieved as it is available in the repository of University C (see Figure 6.19).
Figure 6.17 First Sample Screen of Retrieving Inter-organizational Knowledge in accordance with ITC’s Ontology (Concepts Retrieved from Organizational Ontology)
Figure 6.18 Second Sample Screen of Retrieving Inter-organizational Knowledge based on ITC’s Ontology (Mappings between Ontology1 and Extonology3 as well as Concepts Retrieval from Inter-organizational Ontology)

Figure 6.19 Sample Screen of Inter-organizational Knowledge Retrieved from ITC
6.3.3 Illustration of Mapping Establishment

Mappings must be created between each pair of semantically identical concepts from two different ontologies in order to allow general users to retrieve inter-organizational knowledge. To identify matching concepts and to establish and upload mappings, the ontology management team in ITA uses NeOn Toolkit and its alignment plugin in the local computer.

Consider the following example as an illustration to the establishment of mappings between ITA and ITC. Firstly, the ontology management team of ITA needs to select Ontology1 of ITA and Extontology3 of ITC on the alignment user interface of NeOn Toolkit (see Figure 6.20). Before clicking on “Match” button, the team must first select a matching method. The team chooses StringDistAlignment because based on past experiences it has been confirmed that the selected matcher has the best performance compared to the other seven matchers in terms of precision, recall and f-measure (Fu 2011). A score is issued to every pair of concepts between Ontology1 and Extontology3.

Figure 6.20 Sample Screen of User Interface of NeOn Toolkit’s Alignment Plugin
The matching results are shown in Figure 6.21 for pairs that have scored higher than zero. As there are 26 results in this matching, it is appropriate to remove some of the irrelevant pairs with low matching score by setting a threshold of 0.5 (as illustrated in Figure 6.22). Figure 6.23 shows the matching pairs with a threshold equal or higher than 0.5. These pairs can then be saved by clicking the Import button in the Toolkit (see Figure 6.24) and upload to the system by clicking “Add Ontology” on the navigation menu of the web user interface (see Figure 6.14).
Apart from adding mappings, the “Add Ontology” function can also be used to upload ontologies as well. Uploaded ontologies and alignments can be viewed by clicking the “Uploaded Ontologies” function. In addition, the “Settings” function allows the ontology management team to specify the repository address and ID as well as the master ontology and its root class. If the organization is involved in more
than one KM networks, the “Setting” functionality enables the ontology management team to switch to the appropriate repository, master ontology and its root class. The “Manage Repository” function on the navigation menu allows members of the management team to access the OpenRDF Sesame through OpenRDF Workbench to create, query, update and explore repositories.

### 6.3.4 Illustration of Knowledge Creation

To build its own ontology, ITD decides to adopt MIOD as the designated development methodology because MIOD incorporates ontology integration mechanism and a semi-automatic tool (ICOES) in the development methodology. There are six phases in MIOD, namely preparation, analysis, integration, development, implementation and evaluation, and maintenance. In the preparation phase, the ontology development team of ITD identifies the purpose and scope of the ontology as follows.

- **Purpose**: To provide a meta-knowledge layer to categorise IT help desk knowledge reposited in ITD’s knowledge base.
- **Scope**: Categories of IT admin issues, hardware and software problems and website errors

Apart from the project manager, system analyst, ontology engineer and knowledge engineer, the ontology development team also includes four domain experts who possess extensive IT knowledge. The ontology development team has also conducted a thorough analysis to assess technical, resource and financial feasibilities. In the feasibility analysis, the development team finds that ontology-based KMS is the best solution among all other alternatives in terms of performance, ease of use and efficiency. The team also confirms there are sufficient human, technical and time resources as well as cost-effectiveness to conduct the ontology development project. The preparation phase is concluded by generating a project schedule.

In the analysis phase, the ontology development team is required to develop a set of inspiration scenarios that include “finding knowledge of IT admin issues”, “finding
hardware knowledge”, “finding software knowledge” and “finding solutions to web application errors”. A sample of details of inspiration scenario including scenario name, actors, scenario descriptions and verification questions are recorded in a template of inspiration scenario as shown in Figure 6.25

![Figure 6.25 Inspiration Scenario of “Finding Solutions to Web Application Errors”](image)

Based on the inspiration scenarios, the development team can identify all possible key terms related to the scenarios. The key terms are verified and categorised into four categories: IT admin, hardware, software and web. Table 6.4 shows the set of categorised key terms identified by ITD. Moreover, the team has also identified a set of candidate ontologies from ITA (Ontology1), ITB (Extontology2), ITC (Extontology3) and ITE (Extontology1).
Table 6.4 Set of Categorised Keywords Identified by University D

<table>
<thead>
<tr>
<th>IT Admin</th>
<th>Hardware</th>
<th>Software</th>
<th>Web</th>
</tr>
</thead>
<tbody>
<tr>
<td>PC Account Problem</td>
<td>Mobile Device</td>
<td>Microsoft Application</td>
<td>Web Application</td>
</tr>
<tr>
<td>PC Account Termination</td>
<td>Tablet Computer</td>
<td>Window XP</td>
<td>HTTP</td>
</tr>
<tr>
<td>PC Account Setup</td>
<td>Smart Phone</td>
<td>Window Vista</td>
<td>Client</td>
</tr>
<tr>
<td>PC Account Suspension</td>
<td>TV</td>
<td>Window 7</td>
<td>Server</td>
</tr>
<tr>
<td>Login Problem</td>
<td>DVD Player</td>
<td>Customise Taskbar</td>
<td>SMTP</td>
</tr>
<tr>
<td>Email Account Problem</td>
<td>Projector</td>
<td>Customise Start Menu</td>
<td>POP3</td>
</tr>
<tr>
<td>Email Account Termination</td>
<td></td>
<td>FTP</td>
<td></td>
</tr>
<tr>
<td>Email Account Setup</td>
<td></td>
<td>Persistent Transient Failure</td>
<td></td>
</tr>
<tr>
<td>Email Account Suspension</td>
<td></td>
<td>Permanent Failure</td>
<td></td>
</tr>
<tr>
<td>Email Account Maintenance</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hardware Installation Guidance</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hardware Disposal Guidance</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

To evaluate candidate ontologies, the ontology development team in ITD can make use of the ontology evaluation function of the prototype to determine if the candidate ontologies are suitable to be integrated as source ontologies in MIOD. This function can be activated by selecting the “Evaluate Ontology” function in the navigation menu of the web user interface of the prototype (see Figure 6.14). As shown in Figure 6.26, the team has to enter a list of keywords and select a candidate ontology before evaluation can take place. After selecting the “Evaluate” function, the evaluation results are displayed.
The prototype is designed to generate evaluation results and the results can be divided into two types. The first result is for each key term that is entered for evaluation and it includes min score, max score, average score, number of matches and score variance (see Figure 6.27). Min and max scores are the minimum and maximum similarity score attained for a key term against all concepts in the candidate ontology and the value of the scores can be influenced by specifying a threshold. Average score is calculated by dividing the total similarity score of a key term by the number of matching concepts (number of matches shown in Figure 6.7) for which similarity score is greater than zero whereas score variance is the variance of similarity measure for the key terms.

Figure 6.27 Sample Screen of Evaluation Results of IT Admin Key Terms and Ontology1
The second result is the overall results for all key terms against the candidate ontology that includes coverage and distribution (see Figure 6.27). In the output, coverage is the average maximum similarity (max score) of all key terms and distribution is the average number of unmatched concepts between a pair of matched concepts using the shortest path. The team can choose to display all of the above results or to include only the coverage, distribution and max score for each key term. A summary of evaluation results conducted by ITD is illustrated in Table 6.5. Based on the results in this instance, it is appropriate to select Ontology1, Extontology1, Extontology2 and Extontology3 for IT admin, software, hardware and website respectively as source ontologies as their coverage scores are the highest in the categories. In some cases, if the coverage scores are very close, then the distribution scores will be used to determine the most appropriate source ontology.

<table>
<thead>
<tr>
<th>Ontology</th>
<th>IT Admin</th>
<th>Hardware</th>
<th>Software</th>
<th>Website</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Coverage: 0.674</td>
<td>Coverage: 0.132</td>
<td>Coverage: 0</td>
<td>Coverage: 0.309</td>
</tr>
<tr>
<td></td>
<td>Distribution: 3.0</td>
<td>Distribution: 0</td>
<td>Distribution: 0</td>
<td>Distribution: 1.7</td>
</tr>
<tr>
<td>Extontology1</td>
<td>Coverage: 0</td>
<td>Coverage: 0</td>
<td>Coverage: 0.917</td>
<td>Coverage: 0</td>
</tr>
<tr>
<td></td>
<td>Distribution: 0</td>
<td>Distribution: 0</td>
<td>Distribution: 0.8</td>
<td>Distribution: 0</td>
</tr>
<tr>
<td>Extontology2</td>
<td>Coverage: 0</td>
<td>Coverage: 0</td>
<td>Coverage: 0</td>
<td>Coverage: 0</td>
</tr>
<tr>
<td></td>
<td>Distribution: 0</td>
<td>Distribution: 0</td>
<td>Distribution: 0</td>
<td>Distribution: 0</td>
</tr>
<tr>
<td>Extontology3</td>
<td>Coverage: 0</td>
<td>Coverage: 0</td>
<td>Coverage: 0.095</td>
<td>Coverage: 0.677</td>
</tr>
<tr>
<td></td>
<td>Distribution: 0</td>
<td>Distribution: 0</td>
<td>Distribution: 0</td>
<td>Distribution: 0.786</td>
</tr>
</tbody>
</table>

Based on the evaluation results generated by the ICOES, the development team has selected all four candidate ontologies as the source ontologies. The development team has also identified the appropriate knowledge module from each source ontology. Figure 6.28, 6.29, 6.30 and 6.31 show the knowledge module partition from Ontology1, Extontology2, Extontology3 and Extontology1 respectively.
In the integration phase, the team is required to evaluate the quality of the knowledge modules from the source ontologies using the set of categorised key terms, inspiration scenarios and verification questions. The development team has renamed some of the concepts to match with the key terms identified in the analysis phase.
Additional concepts are also added to ensure completeness of the knowledge modules. This phase is concluded by integrating the knowledge modules into an integrated ontology as shown in Figure 6.32.

![Figure 6.32 Sample of the Integrated Ontology of ITD](image)

As there is no unused key terms left, it is not necessary to conduct the development phase. In the implementation phase, the development team encodes the integrated ontology into OWL. The formalized ontology is then evaluated in the application environment by a group of general users. When the users are satisfied with the performance of the KMS and the ontology, the ontology will be signed off and
deployed. From this point onward, the ontology enters the maintenance phase and regular update is required to maintain its validity.

### 6.4 Conclusion

An application scenario has developed in this chapter to illustrate how participant organization such as ITA make uses of KOM-Frame to select from a range of ontology mediation options on both KM network- and organizational-levels during the establishment of IT knowledge CIK-Net. The scenario presented is also used to illustrate the reuse of inter-organizational to support explicit knowledge dissemination in the CIK-Net. This is achieved by the prototype developed for ITA to retrieve organizational and inter-organizational knowledge.

The prototype provides a web user interface to allow ITA users to retrieve organizational and inter-organizational knowledge through established mappings among various ontologies. Users describe the knowledge types and their characteristics by selecting the related vocabularies in which the vocabularies are supplied from the organizational ontology. The required knowledge can be retrieved if it is available in the knowledge repository of ITA. Otherwise, the system will retrieve the knowledge from knowledge repository of other participating organizations through mappings established in the KM network. The system also provides functionality to enable ITA to find identical concepts and create mappings between ontologies of ITA and other organizations in the network.

In the example presented, reuse of inter-organizational knowledge to support explicit knowledge creation in the CIK-Net is also described. In the example, ITD adopts MIOD to develop its organizational ontology by reusing inter-organizational ontologies through the CIK-Net. The prototype also demonstrates functionalities of the proposed ICOES to allow the ITD ontology development team to find suitable source ontologies using a single matching technique and a concept distribution counter.
The illustration of the prototype shows that: 1) KMS of ITA can retrieve inter-organizational knowledge through the established mappings in the CIK-Net, and 2) MIOD can be used to support explicit knowledge creation for ITD in the CIK-Net. The application scenario also shows the effectiveness of using KOM-Net, MIOD and ICOES in the CIK-Net. In conclusion, the proposed CIK-Net enables inter-organizational knowledge to be reused and to support knowledge creation and dissemination in participant organization’s individual KM process.
This chapter concludes the presentation of the thesis. A summary of research results, contributions and future research directions are presented.

This chapter is organized as follows. Section 1 discusses research results. Section 2 outlines research contributions. Future research direction is given in Section 3.

7.1 Research Results

As outlined in Chapter 1, this research aims to investigate a theoretical ontology mediation framework for developing an integrated ontology approach by reusing inter-organizational ontologies. In addition, this research also examines an ontology-based KM approach to enable the interoperation of heterogeneous KMS in the domain of reusing inter-organizational knowledge. This research has addressed the four objectives outlined in Chapter 1.

The three tiers DISCOMT has been proposed as ontology matching techniques are crucial in finding suitable source ontologies in MIOD and reconciling ontology mismatches in the ontology-based CIK-Net. The purpose of DISCOMT aims to provide a clear guideline on designing new mediation tool and a method to identify the type of the matching technique and its related executive approach. We have demonstrated that DISCOMT can provide a valuable reference point when choosing appropriate matching techniques in ICOES and CIK-Net.

MIOD has been proposed to address the lack of details and insights in ontology integration in the current literature. MIOD is an ontology development methodology that consists of six phases: preparation, analysis, integration, development, implementation and evaluation, and maintenance. The methodology provides a detailed description on how to incorporate and perform integration in its ontology development process. The illustration of MIOD in the application scenario shows
that MIOD is capable of developing an IT knowledge ontology by reusing existing inter-organizational ontologies

A semi-automatic tool, ICOES, is included as a component of MIOD to assist finding suitable source ontologies from a group of candidate ontologies using concept distribution counter as well as a single or a combination of matching techniques. A prototype of ICOES has been implemented and illustrated in the application scenario in Chapter 6. The illustration shows that ICOES is capable of finding candidate ontologies in the development of organizational ontology using MIOD.

An ontology-based CIK-Net has been proposed to address limitations of current KM research that only focuses on managing organizational knowledge. CIK-Net aims to resolve current collaboration problem in which organizations are unable to reuse inter-organizational knowledge. Organizations joining the CIK-Net are required to annotate their knowledge explicitly in their ontologies.

In the CIK-Net, merging is used to build a common ontology to be used in the top-level ontology mapping approach. For organizations that wish to join the KM network but currently do not possess ontologies of their own, integration method can help these organizations to reuse the inter-organizational ontologies to build their own ontologies. Through integration, the associated inter-organizational knowledge can also be reused in knowledge creating and storing phases of the KM process in the organizations. Mappings created by ontology mapping approach enable users to retrieve and use inter-organizational knowledge to support knowledge dissemination and use within the participant organizations. To establish the CIK-Net, KOM-Frame has been proposed to assist organizations in choosing suitable ontology mediation options.

The illustration of CIK-Net in the application scenario and the prototype shows that inter-organizational knowledge can be reused to support explicit knowledge creation. This is achieved by adopting MIOD and ICOES to build an integrated ontology by reusing inter-organizational ontologies. The illustration of CIK-Net also shows that inter-organizational knowledge can be reused to support explicit knowledge dissemination. The illustration of CIK-Net also shows that KOM-Frame has provided
comprehensive guidelines to choose appropriate ontology mediation options. A summary of features and contributions of DISCOMT, MIOD and CIK-Net is shown in Table 7.1

Table 7.1 Summary of Features and Contributions of DISCOMT, MIOD and CIK-Net

<table>
<thead>
<tr>
<th>Framework</th>
<th>Main Features</th>
<th>Main Contribution</th>
<th>Chapter</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISCOMT</td>
<td>DISCOMT is a three tiers classification of ontology matching techniques in which: • T1 consists of 3 executive approaches • T2 consists of 7 matching techniques • T3 consists of 8 input types</td>
<td>Provide a foundation for the development of ontology matching techniques used in mediation tools.</td>
<td>3</td>
</tr>
<tr>
<td>MIOD</td>
<td>MIOD provides a detailed description on how to perform integration within an ontology development process that includes six phases: preparation, analysis, integration, development, implementation and evaluation, and maintenance. MIOD incorporates ICOES to assist finding suitable source ontologies semi-automatically using distribution counter and ontology matching techniques.</td>
<td>Allow ontology integration to be applied in the ontology development process.</td>
<td>4</td>
</tr>
<tr>
<td>CIK-Net</td>
<td>CIK-Net is designed to address: • Current research only focuses on organizational knowledge management • Collaborative problem in which organizations are unable to reuse inter-organizational knowledge KOM-Frame is developed to assist organizations in choosing suitable ontology mediation options.</td>
<td>Provide framework of reusing inter-organizational knowledge to support knowledge creation and dissemination.</td>
<td>5</td>
</tr>
</tbody>
</table>

7.2 Research Contributions

This research has made original contributions as follows.
• A classification framework of ontology matching techniques is formalised to provide a foundation for the development of ontology matching techniques used in mediation tools.

• An integration-oriented ontology development methodology is proposed to develop an ontology by reusing existing ontology. An original contribution is made to allow ontology integration to be applied in the ontology development process.

• An ontology-based KM framework is proposed to reuse inter-organizational knowledge to create and disseminate organizational knowledge. An original contribution is made in providing framework of reusing inter-organizational knowledge to support knowledge creation and dissemination.

7.3 Research Limitations

There are two major limitations in this research and they are listed as follows.

1. NeOn Toolkit (and its alignment plugin) is one of the major components of the prototype developed in Chapter 6 to demonstrate the application scenario. Unfortunately, NeOn Toolkit can reduce the reusability of the prototype as the Toolkit is not a web-based application and cannot be installed on the server. Users can access the prototype to retrieve knowledge and evaluate candidate ontologies through the designated URL. However, if users want to identify matching concepts and establish mappings between each pair of semantically identical concepts from two different ontologies, the Toolkit must be installed on users’ computers.

2. The CIK-Net was evaluated by one application scenario in this research. More application scenarios can be developed to provide additional depth to evaluation findings. This includes establishing scenarios that specifies domain knowledge other than IT help desk knowledge, organizations adopting different types of ontologies such as heavyweight ontologies and
using multiple ontology matching techniques to identify semantically identical concepts or to evaluate candidate ontologies.

7.4 Further Research

A business ecosystem is an economic community that produces goods and services to customers (Moore 1996). Each business ecosystem is supported by a group of suppliers, lead producers, competitors and other stakeholders that share the same vision to align their investments and to find mutually supportive roles. Members of a business ecosystem tend to align themselves with direction set by one or more dominant members holding the leadership. For example, Google is the leader of Android ecosystem in which Android-based smartphones occupied 43% of market share in the USA smartphone market in the third quarter of 2011 (Yin 2011). In the Android ecosystem, there will be a large number of companies (such as app development companies and processor manufacturers) that rely on direction of Google. The concept of CIK-Net can be applied to individual business ecosystem to enables inter-organizational knowledge to be reused to support organizational knowledge creation and dissemination.

Globalization has made the world smaller and flatter and has created many opportunities for various business ecosystems. One way to stay competitive is to combine various ecosystems into one ecosystem. For instance, Nokia ecosystem has combined with Window Mobile ecosystem to develop Window-based Nokia smartphones in order to compete with iPhone and Android-based smartphones. However, the combination of multiple ecosystems also requires their individual CIK-Nets to combine to allow inter-organizational knowledge to be accessed, retrieved and reused. It is not a simple task to combine various CIK-Nets as each CIK-Net adopts different mapping approach and each participant organization adopts different kind of ontology and mediation methods. Thus, further research on approaches that enable CIK-Nets to be combined will be pursued in future research to investigate practical applications of CIK-Net in the business ecosystem.
REFERENCES


160


Muslea, I. (2002). Active Learning with Multiple Views, PhD thesis, Department of Computer Science, University of Southern California.


APPENDIX A – SOURCE CODES FOR PROTOTYPE

AddExtOntologyAction.java

package org.semantichelpdesk.struts2;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.util.LinkedHashMap;
import javax.servlet.http.HttpServletRequest;
import org.apache.struts2.interceptor.ServletRequestAware;
import org.openrdf.OpenRDFException;
import org.openrdf.model.Resource;
import org.openrdf.model.Statement;
import org.openrdf.model.URI;
import org.openrdf.model.ValueFactory;
import org.openrdf.model.vocabulary.RDFS;
import org.openrdf.repository.Repository;
import org.openrdf.repository.RepositoryConnection;
import org.openrdf.repository.RepositoryResult;
import org.openrdf.repository.http.HTTPRepository;
import org.openrdf.rio.RDFFormat;
import org.openrdf.rio.rdfxml.RDFXMLWriter;
import com.hp.hpl.jena.rdf.model.InfModel;
import com.hp.hpl.jena.rdf.model.Model;
import com.hp.hpl.jena.reasoner.ReasonerRegistry;
import com.hp.hpl.jena.util.FileManager;
import com.hp.hpl.jena.vocabulary.OWL;
import com.hp.hpl.jena.vocabulary.RDF;
import com.opensymphony.xwork2.ActionSupport;

public class AddExtOntologyAction extends ActionSupport implements ServletRequestAware {
    private File addOntology;
    private String addOntologyContentType;
    private String addOntologyFileName;
    private HttpServletRequest servletRequest;
    private SettingsAction settings = new SettingsAction();
    private LinkedHashMap<String, String> ontologies = new LinkedHashMap<String, String>();

    public LinkedHashMap<String, String> getOntologies() {
        return ontologies;
    }

    public void setOntologies(LinkedHashMap<String, String> ontologies) {

    }
this.ontologies = ontologies;
}

public String execute() {
    try {
        String filePath =
    servletRequest.getSession().getServletContext().getRealPath("/");
    System.out.println("Server path:" + filePath);
    File tmpfile = new File(this.addOntologyFileName);
    File dtmpfile = new File("d"+this.addOntologyFileName);
    // reasoning
    Model data =
    FileManager.get().loadModel(this.addOntology.getAbsolutePath());
    // add existing data in master rep to deducted rep
    String baseURI = settings.getMs_ont_base();
    String rootClass = settings.getMs_ont_root();
    String sesameServer = settings.getRep_address() + "/openrdf-sesame";
    String masterRepositoryID = settings.getRep_id();

    Repository masterRepository = new HTTPRepository(sesameServer,
    masterRepositoryID);
    masterRepository.initialize();
    ValueFactory f = masterRepository.getValueFactory();
    URI staticContext =
    f.createURI("http://www.helpdesk.org/static");
    URI deductedContext =
    f.createURI("http://www.helpdesk.org/deducted");

    try {
        RepositoryConnection con =
        masterRepository.getConnection();
        try {
            con.add(this.addOntology, null, RDFFormat.RDFXML,
            staticContext);
            con.export(new RDFXMLWriter(new
            FileOutputStream(tmpfile)),staticContext);
            data.read(new FileInputStream(tmpfile), null);
            com.hp.hpl.jena.reasoner.Reasoner reasoner =
            ReasonerRegistry.getOWLReasoner();
            InfModel infmodel =
            ModelFactory.createInfModel(reasoner, data);
            FileOutputOutputStream fos = new
            FileOutputOutputStream(dttmpfile);
            Model deductedModel =
            infmodel.getDeductionsModel();
            infmodel.write(fos, "RDF/XML-ABBREV");
            con.clear(deductedContext);
            con.add(dttmpfile, null,
            RDFFormat.RDFXML,deductedContext);
        } finally {
            con.close();
        }
    } catch (OpenRDFException e) {
        // handle exception
    } catch (Exception e) {
e.printStackTrace();
addActionError(e.getMessage());

return INPUT;
}
return SUCCESS;
}

public File getAddOntology() {
    return addOntology;
}

public void setAddOntology(File addOntology) {
    this.addOntology = addOntology;
}

public String getAddOntologyContentType() {
    return addOntologyContentType;
}

public void setAddOntologyContentType(String addOntologyContentType) {
    this.addOntologyContentType = addOntologyContentType;
}

public String getAddOntologyFileName() {
    return addOntologyFileName;
}

public void setAddOntologyFileName(String addOntologyFileName) {
    this.addOntologyFileName = addOntologyFileName;
}

@Override
public void setServletRequest(HttpServletRequest servletRequest) {
    this.servletRequest = servletRequest;
}
package org.semantichelpdesk.struts2;

import java.io.File;

import javax.servlet.http.HttpServletRequest;

import org.apache.struts2.interceptor.ServletRequestAware;
import org.openrdf.OpenRDFException;
import org.openrdf.repository.Repository;
import org.openrdf.repository.RepositoryConnection;
import org.openrdf.repository.http.HTTPRepository;
import org.openrdf.rio.RDFFormat;
import com.opensymphony.xwork2.ActionSupport;

public class AddOntologyAction extends ActionSupport implements ServletRequestAware {
    private File addOntology;
    private String addOntologyContentType;
    private String addOntologyFileName;
    private HttpServletRequest servletRequest;

    public String execute() {
        try {
            String filePath = servletRequest.getSession().getServletContext().getRealPath("/");
            System.out.println("Server path:" + filePath);
            // storing uploaded ontology to repository
            String baseURI = getText("master.ontology.base.uri");
            String sesameServer = getText("menu.sesame.server.url") + "/openrdf-sesame";
            String repositoryID = getText("sesame.server.master.rep");

            Repository myRepository = new HTTPRepository(sesameServer, repositoryID);
            myRepository.initialize();
            try {
                RepositoryConnection con = myRepository.getConnection();
                try {
                    con.add(this.addOntology, baseURI, RDFFormat.RDFXML);
                }
            } finally {
                con.close();
            }
        } catch (OpenRDFException e) {
            // handle exception
            // FileUtils.copyFile(this.addOntology, fileToCreate);
        } catch (Exception e) {
            e.printStackTrace();
            addActionError(e.getMessage());
        }
    }
}

AddOntologyAction.java
return INPUT;
}
return SUCCESS;
}

public File getAddOntology() {
    return addOntology;
}

public void setAddOntology(File addOntology) {
    this.addOntology = addOntology;
}

public String getAddOntologyContentType() {
    return addOntologyContentType;
}

public void setAddOntologyContentType(String addOntologyContentType) {
    this.addOntologyContentType = addOntologyContentType;
}

public String getAddOntologyFileName() {
    return addOntologyFileName;
}

public void setAddOntologyFileName(String addOntologyFileName) {
    this.addOntologyFileName = addOntologyFileName;
}

@Override
public void setServletRequest(HttpServletRequest servletRequest) {
    this.servletRequest = servletRequest;
}
package org.semantichelpdesk.struts2;

import java.util.ArrayList;
import java.util.LinkedHashMap;
import java.util.List;
import javax.servlet.http.HttpServletRequest;
import org.apache.struts2.interceptor.ServletRequestAware;
import org.openrdf.OpenRDFException;
import org.openrdf.model.Resource;
import org.openrdf.model.Statement;
import org.openrdf.model.URI;
import org.openrdf.model.ValueFactory;
import org.openrdf.model.vocabulary.OWL;
import org.openrdf.model.vocabulary.RDFS;
import org.openrdf.repository.Repository;
import org.openrdf.repository.RepositoryConnection;
import org.openrdf.repository.RepositoryException;
import org.openrdf.repository.http.HTTPRepository;
import com.opensymphony.xwork2.ActionSupport;

public class BrowseKnowledgeAction extends ActionSupport implements ServletRequestAware {
    private HttpServletRequest servletRequest;
    private LinkedHashMap<String, LinkedHashMap<String, String>> classTax;
    private Boolean showText = false;
    private String inputText = null;
    private List<String> externalTexts;
    private String leafMasterClass;

    public String getLeafMasterClass() {
        return leafMasterClass;
    }

    public void setLeafMasterClass(String leafMasterClass) {
        this.leafMasterClass = leafMasterClass;
    }

    private SettingsAction settings = new SettingsAction();

    public String getInputText() {
        return inputText;
    }

    public void setInputText(String inputText) {
        this.inputText = inputText;
    }

    public Boolean getShowText() {
        return showText;
    }
}
public void setShowText(Boolean showText) {
    this.showText = showText;
}

getClassTax() {
    return classTax;
}

public void setClassTax(LinkedHashMap<String, LinkedHashMap<String, String>> classTax) {
    this.classTax = classTax;
}

private LinkedHashMap<String, String> subclassKeys = new LinkedHashMap<String, String>();

private List<String> rootClasses;

public List<String> getRootClasses() {
    return rootClasses;
}

public void setRootClasses(List<String> rootClasses) {
    this.rootClasses = rootClasses;
}

public LinkedHashMap<String, String> getSubclassKeys() {
    return subclassKeys;
}

public void setSubclassKeys(LinkedHashMap<String, String> selectedSubclass) {
    this.subclassKeys = selectedSubclass;
}

public String saveKnowledge() {
    return this.execute();
}

public String execute() {
    try {
        String sesameServer = settings.getRep_address() + "/openrdf-sesame";
        String repositoryID = settings.getRep_id();
        Repository myRepository = new HTTPRepository(sesameServer, repositoryID);
        myRepository.initialize();
        // storing uploaded ontology to repository
        String baseURI = settings.getMs_ont_base();
        String rootClass = settings.getMs_ont_root();
        ValueFactory f = myRepository.getValueFactory();
        URI staticContext = f.createURI("http://www.helpdesk.org/static");
        URI deductedContext = f.createURI("http://www.helpdesk.org/deducted");
if (rootClasses == null)
rootClasses = new ArrayList<String>();
rootClasses.add(0, baseURI + "#" + rootClass);

classTax = new LinkedHashMap<String,
LinkedHashMap<String, String>>();
subclasses = null;
try {
    RepositoryConnection con =
myRepository.getConnection();
    try {
        String prevSuperClass = "";
        boolean textProcessed = false;
        for (java.util.ListIterator<String> iter =
rootClasses
            .listIterator();
        iter.hasNext();)
            String superClass = iter.next();
        if (superClass.endsWith("http://example.com"))
            break;
        //check if redraw the down list
        with textarea
        if (subclasses != null
            && !subclasses.containsKey(superClass))
            break;
    }
    subclasses = new
LinkedHashMap<String, String>();
    statements = con
        .getStatements((Resource) null, RDFS.SUBCLASSOF,
            f.createURI(superClass), true, staticContext);
    //find subclasses of any equivalent
class
    eqSubclassListStatements = null;
    eqStatements = con
        .getStatements((Resource) null, OWL.EQUIVALENTCLASS,
            f.createURI(superClass), true, staticContext);
    boolean hasEqSubclasses = false;
    try{
        eqStatements.next();
        Resource subject =
            st.getSubject();
        eqSubclassListStatements = con
            .getStatements((Resource) null, RDFS.SUBCLASSOF,
            subject, true,
            staticContext);
while (eqSubclassStatements.hasNext()) {
    hasEqSubclasses = true;
    stSub = eqSubclassStatements.next();
    subjectSub = stSub.getSubject();
    subclassName = subjectSub.stringValue().substring(subjectSub.stringValue().indexOf("#") + 1);
    subclasses.put(subjectSub.stringValue(), subclassName);
}

if ((statements == null || !statements.hasNext()) && !textProcessed && !hasEqSubclasses //uncomment for only leaf master knowledge
    ) {// output knowledge
    edit
    // form in the form of an instance
    this.showText = true;
    textProcessed = true;
    //get knowledge instances
    from master ontology
    instText = getDirectKnowledge(f, con, superClass, staticContext);
    // provide text to output
    if (inputText == null) inputText = instText;
    superClass;
    // get knowledge instances
    from external ontologies
    if(!inputText.equals(instText)){
        // update instance...}
setDirectKnowledge(f, con, superClass, inputText, staticContext);

if (statements.hasNext() || (hasEqSubclasses)) {
    try {
        while (statements.hasNext()) {
            Statement st = statements.next();
            Resource subject = st.getSubject();
            String subclassName = subject.stringValue().substring(subject.stringValue().indexOf("#") + 1);
            subclasses.put(subject.stringValue(), subclassName);
            // add selected class and its subclasses
            classTax.put(superClass, subclasses);
        }
    } finally {
        statements.close(); // make sure the result // object // is closed properly
    }
}

if (!superClass.startsWith(baseURI)) {
    // output external knowledge
    externalTexts = getExtDirectKnowledge(f, con, superClass, staticContext);
    // clean up double comment from static context in deducted
    externalTexts.remove("\" + inputText + "\");
}

// specify which option should be selected
if (prevSuperClass != ")
    this.subclassKeys.put(prevSuperClass, superClass);
    prevSuperClass = superClass;
}
private List<String> getKnowledgeFromSubclasses(ValueFactory f, RepositoryConnection con, String superClass, Resource context) throws RepositoryException {
    List<String> extTexts = new ArrayList<String>();
    RepositoryResult<Statement> subclasses = con.getStatements(null, RDFS.SUBCLASSOF, f.createURI(superClass), true, context);
    RepositoryResult<Statement> comments = null;
    try{
        while(subclasses.hasNext()){
            Statement statement = subclasses.next();
            comments = con.getStatements(statement.getSubject(), RDFS.COMMENT, null, true, context);
            while(comments.hasNext())
                extTexts.add(comments.next().getObject().toString());
        }
    }finally{
        if(comments != null) comments.close();
        subclasses.close();
    }
    return extTexts;
}

private void setDirectKnowledge(ValueFactory f, RepositoryConnection con, String superClass, String text, Resource context) throws RepositoryException {
    // get this labels or comments
    RepositoryResult<Statement> instLabels = con.getStatements(f.createURI(superClass),

    } finally {
        con.close();
    }

    } catch (OpenRDFException e) {
        // handle exception
    }

    } catch (Exception e) {
        e.printStackTrace();
        addActionError(e.getMessage());
        return INPUT;
    }
    return SUCCESS;
}
try {
    con.setAutoCommit(false);
    while (instLabels.hasNext()) {
        Statement labelSt = instLabels.next();
        con.remove(labelSt);
    }
    con.add(f.createURI(superClass), RDFS.COMMENT, 
            f.createLiteral(text),context);
    con.commit();
} catch (RepositoryException e) {
    // Something went wrong during the transaction, so we roll it back
    con.rollback();
} finally {
    instLabels.close(); // is closed properly
    con.setAutoCommit(true);
}

private String getDirectKnowledge(ValueFactory f, 
        RepositoryConnection con, 
        String superClass, Resource context) throws 
        RepositoryException {
    // get this instance's label or comment
    RepositoryResult<Statement> instLabels = con.getStatements(
        f.createURI(superClass), 
        RDFS.COMMENT, null, true, context);
    try {
        if (instLabels.hasNext()) {
            Statement labelSt = instLabels.next();
            return labelSt.getObject().stringValue();
        } else {
            return "";
        }
    } finally {
        instLabels.close(); // is closed properly
    }
}

private List<String> getExtDirectKnowledge(ValueFactory f, 
        RepositoryConnection con, 
        String superClass, Resource context) throws 
        RepositoryException {
    // get this instance's label or comment
    List<String> extTexts = new ArrayList<String>();
    RepositoryResult<Statement> instLabels = con.getStatements(
        f.createURI(superClass), 
        RDFS.COMMENT, null, true, context);
    try{
        while(instLabels.hasNext()){
            extTexts.add(instLabels.next().getObject().toString());
        }
    }finally{
        instLabels.close();
    }
    return extTexts;
@Override
public void setServletRequest(HttpServletRequest servletRequest) {
    this.servletRequest = servletRequest;
}
CustomerAction.java

package org.semantichelpdesk.struts2;

import com.opensymphony.xwork2.ActionSupport;

public class CustomerAction extends ActionSupport{
    private String name;
    private Integer age;
    private String email;
    private String telephone;

    public String addCustomer() {
        return SUCCESS;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public Integer getAge() {
        return age;
    }

    public void setAge(Integer age) {
        this.age = age;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public String getTelephone() {
        return telephone;
    }

    public void setTelephone(String telephone) {
        this.telephone = telephone;
    }
}
public class EvaluateOntologyAction extends ActionSupport implements ServletRequestAware {
    private File addOntology;
    private String keywords;
    private String trim;
    private Double coverage;
    private Double distribution;
    private boolean simpleScore;
    private LinkedHashMap<String, ArrayList<String>> shortestPaths;

    public LinkedHashMap<String, ArrayList<String>> getShortestPaths() {
        return shortestPaths;
    }
}

import java.io.BufferedReader;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileWriter;
import java.io.IOException;
import java.net.URL;
import java.util.ArrayList;
import java.util.Enumeration;
import java.util.LinkedHashMap;
import java.util.Properties;
import javax.servlet.http.HttpServletRequest;
import org.apache.struts2.interceptor.ServletRequestAware;
import org.openrdf.model.vocabulary.RDFS;
import org.semantichelpdesk.semweb.JenaOntToolsEx;
import org.semanticweb.owl.align.Alignment;
import org.semanticweb.owl.align.AlignmentException;
import org.semanticweb.owlapi.model.OWLClass;
import com.hp.hpl.jena.ontology.OntClass;
import com.hp.hpl.jena.ontology.OntModel;
import com.hp.hpl.jena.ontology.OntModelSpec;
import com.hp.hpl.jena.ontology.OntTools.Path;
import com.hp.hpl.jena.rdf.model.Model;
import com.hp.hpl.jena.rdf.model.ModelFactory;
import com.hp.hpl.jena.util.FileManager;
import com.opensymphony.xwork2.ActionSupport;
import fr.inrialpes.exmo.align.impl.BasicParameters;
import fr.inrialpes.exmo.align.impl.DistanceAlignment;
import fr.inrialpes.exmo.align.impl.method.SubsDistNameAlignment;
import fr.inrialpes.exmo.align.impl.renderer.OWLAxiomsRendererVisitor;
import fr.inrialpes.exmo.align.impl.renderer.RDFRendererVisitor;
import fr.inrialpes.exmo.align.ling.JWNLAlignment;
public void setShortestPaths(LinkedHashMap<String, ArrayList<String>> shortestPaths) {
    this.shortestPaths = shortestPaths;
}

private ArrayList<String> matchedConceptsNames = new ArrayList<String>();

public ArrayList<String> getMatchedConceptsNames() {
    return matchedConceptsNames;
}

public void setMatchedConceptsNames(ArrayList<String> matchedConceptsNames) {
    this.matchedConceptsNames = matchedConceptsNames;
}

private ArrayList<URI> matchedConcepts;

public ArrayList<URI> getMatchedConcepts() {
    return matchedConcepts;
}

public void setMatchedConcepts(ArrayList<URI> matchedConcepts) {
    this.matchedConcepts = matchedConcepts;
}

public boolean isSimpleScore() {
    return simpleScore;
}

public void setSimpleScore(boolean simpleScore) {
    this.simpleScore = simpleScore;
}

public String getTrim() {
    return trim;
}

public boolean getHidePaths() {
    SettingsAction settings = new SettingsAction();
    return settings.isHide_paths();
}

public void setTrim(String trim) {
    this.trim = trim;
}

private ArrayList<OntClass> keywordClasses;

private LinkedHashMap<String, InnerStat> keywordsStat;

public String getKeywords() {
    return keywords;
}

public void setKeywords(String keywords) {
    this.keywords = keywords;
}

private String addOntologyContentType;
private String addOntologyFileName;

private HttpServletRequest servletRequest;

@SuppressWarnings({"deprecation", "unchecked"})
public String execute() {
    if (this.keywords == null || this.keywords.isEmpty()) {
        this.keywords = "pop3, problem, access";
        this.trim = "0.5";
        return INPUT;
    }
    if (this.trim == null || this.trim.isEmpty()) this.trim = "0.5";
    try {
        String filePath =
        servletRequest.getSession().getServletContext().getRealPath("/");
        System.out.println("Server path:" + filePath);
        String[] keywordsArray = this.keywords.split(",");
        String baseURI = getText("keyword.ontology.base.uri");
        Properties params = new BasicParameters();
        this.keywordsStat = new LinkedHashMap<String, InnerStat>();
        this.keywordClasses = new ArrayList<OntClass>();
        Double scoreSum = 0.0;
        int matchedKeywords = 0;
        matchedConcepts = new ArrayList<URI>();
        for(int i = 0; i < keywordsArray.length;i++) {
            String keyword = keywordsArray[i].trim();
            File dtmpfile = prepareKeywordOntology(filePath,
            baseURI,
            keyword);
            DistanceAlignment a = doAlignment(params, dtmpfile);
            trimAlignment(a);
            InnerStat stat = calculateStat(a);
            this.keywordsStat.put(keyword, stat);
            //calculate sum of max scores for matched keywords
            scoreSum += stat.nbCells==0?0:stat.maxConfidence;
            if (stat.nbCells != 0) {
                matchedKeywords++;
                //collect matched concepts
                collectMatchedConcepts(matchedConcepts, a);
            }
            //printToSystemOut(as, i);
            printAlignmentToOwl( a, filePath+"keywordsAlignment_"+
            keyword + ".owl");
            //Path path = OntTools.findShortestPath(m,
            keywordClasses.get(0), keywordClasses.get(1), Filter.any);
        }
        this.coverage = keywordsArray.length == 0?
        0:scoreSum/keywordsArray.length;
        //calculate distance
        //get ontology model
        Model data =
        FileManager.get().loadModel(this.addOntology.getAbsolutePath());
        //get unique pairs between concepts
        int numberOfPairs = 0;
    }
int sumUnmatched = 0;
this.shortestPaths = new LinkedHashMap<String,
ArrayList<String>>();
for (int i=0;i<matchedConcepts.size()-1;i++)
    for (int j = i+1; j<matchedConcepts.size();j++){
        numberOfPairs++;
        //for each pair calculate shortest path
        com.hp.hpl.jena.rdf.model.Resource start =
data.getResource(matchedConcepts.get(i).toString());
        com.hp.hpl.jena.rdf.model.Resource end =
data.getResource(matchedConcepts.get(j).toString());
        Path shortestPath =
            JenaOntToolsEx.findShortestPath(data, start, end,
            new com.hp.hpl.jena.ontology.OntTools.PredicatesFilter(data.getProperty(RDFS.SUBCLASSOF.toString())));
        System.out.println(shortestPath.toString());
        //calculate unmatched nodes in each path
        int nbUnmatched = 0;
        ArrayList<String> pathNodes = new ArrayList<String>();
        for (int k = 0; k < shortestPath.size();k++){
            com.hp.hpl.jena.rdf.model.Statement st =
                shortestPath.get(k);
            if (st.getObject().isURIResource() &&
                !matchedConcepts.contains(new URI(st.getObject().asResource().getURI())))
                nbUnmatched++;
            if (st.getSubject().isURIResource() &&
                !matchedConcepts.contains(new URI(st.getSubject().asResource().getURI())))
                nbUnmatched++;
            if (st.getSubject().isURIResource() &&
                !pathNodes.contains(st.getSubject().toString()))
                pathNodes.add(st.getSubject().toString());
            if (st.getObject().isURIResource() &&
                !pathNodes.contains(st.getObject().toString()))
                pathNodes.add(st.getObject().toString());
        }
        this.shortestPaths.put(shortestPath.toString(),pathNodes);
        sumUnmatched += nbUnmatched / 2; //we calculated each not twice as subject and object
    }
    //calculate average number of unmatched nodes per path
    this.distribution =
        (numberOfPairs!=0)?(double)sumUnmatched/(double)numberOfPairs:0.0;
}

} catch (Exception e) {
e.printStackTrace();
addActionError(e.getMessage());
return INPUT;
} return SUCCESS;
private void collectMatchedConcepts(ArrayList<URI> matchedConcepts, DistanceAlignment a) throws AlignmentException {
    // a.cut(a.maxConfidence()); // let us not cut the alignment as several keywords might match the same element
    Enumeration<Cell> elements = a.getElements();
    while(elements.hasMoreElements()) {
        Cell nextElement = elements.nextElement();
        OWLClass object1 = (OWLClass) nextElement.getObject1();
        OWLClass object2 = (OWLClass) nextElement.getObject2();
        URI uri1 = object1.getIRI().toURI();
        URI uri2 = object2.getIRI().toURI();
        String nameId1 = object1.toStringID();
        String nameId2 = object2.toStringID();
        if(this.simpleScore) {
            if (matchedConcepts.contains(uri2)) continue;
            matchedConcepts.add(uri2);
            this.matchedConceptsNames.add(uri2.toString());
        } else {
            if (matchedConcepts.contains(uri1))
                continue;// already matched by another keyword
            matchedConcepts.add(uri1);
            this.matchedConceptsNames.add(uri1.toString());
        }
        break;
    }
}

private InnerStat calculateStat(DistanceAlignment a) {
    InnerStat stat;
    stat = new InnerStat();
    if (a.nbCells() > 0) {
        stat.minConfidence = a.minConfidence();
        stat.maxConfidence = a.maxConfidence();
        stat.avgConfidence = a.avgConfidence();
        stat.nbCells = a.nbCells();
        stat.varianceConfidence = a.varianceConfidence();
    }
    return stat;
}

private void trimAlignment(DistanceAlignment a) throws AlignmentException {
    Double threshould = null;
    try {
        threshould = Double.parseDouble(this.trim);
        if (threshould > 1) threshould = 1.0;
        if (threshould < 0) threshould = 0.0;
    } catch (Exception ex) {
        if (threshould != null) a.cut(threshould);
    }
}

private DistanceAlignment doAlignment(Properties params, File dtmpfile) throws AlignmentException {
    DistanceAlignment a = getAlignment(params, false);
    java.net.URI onto1 = this.addOntology.toURI();
    java.net.URI onto2 = dtmpfile.toURI();
}
if (simpleScore){
    onto2 = this.addOntology.toURI();
    onto1 = dtmpfile.toURI();
}

a.init ( onto1, onto2 );
a.align( (Alignment)null, params );
return a;
}

private File prepareKeywordOntology(String filePath, String baseURI,
String keyword) throws FileNotFoundException, 
IOException {
    File dtmpfile = new File(filePath+"keywords_"+ keyword + 
".owl");
    OntModel m =
    ModelFactory.createOntologyModel(OntModelSpec.OWL_LITE_MEM);
    m.createOntology(baseURI+"_"+keyword);
    String classUri = baseURI+ "_"+keyword + ">#" + keyword;
    keywordClasses.add(m.createClass(classUri));
    FileOutputStream fos = new FileOutputStream(dtmpfile);
    m.write(new FileOutputStream(dtmpfile), "RDF/XML-ABBREV");
    fos.flush();
    fos.close();
    return dtmpfile;
}

private void printAlignmentToOwl(Alignment al, String 
tempOntoFileName) throws IOException, AlignmentException{
    // (Sol1) generate a merged ontology between the ontologies
    PrintWriter writer = null;
    File merged = new File( tempOntoFileName );
    writer = new PrintWriter ( new FileWriter( merged, false ), true );
    AlignmentVisitor renderer = new OWLAxiomsRendererVisitor(writer);
    al.render(renderer);
    writer.flush();
    writer.close();
}

private DistanceAlignment getAlignment(Properties params, boolean 
withWordNet) {
    DistanceAlignment alig;
    if (withWordNet){
        params.setProperty("wnvers", "2.1");
        params.setProperty("wndict", "C:\Program Files
(x86)\WordNet\2.1\dict\\"");
        alig = new JWNLAlignment();
    }else
        alig = new SubsDistNameAlignment();
    return alig;
}

public Double getCoverage() {
    return coverage;
}

public void setCoverage(Double coverage) {
    this.coverage = coverage;
}
public Double getDistribution() {
    return distribution;
}

public void setDistribution(Double distribution) {
    this.distribution = distribution;
}

private void printToSystemOut(SubsDistNameAlignment[] as, int i) throws UnsupportedEncodingException, AlignmentException {
    // Outputing
    PrintWriter writer = new PrintWriter(new BufferedWriter(new OutputStreamWriter(System.out, "UTF-8")), true);
    AlignmentVisitor renderer = new RDFRendererVisitor(writer);
    as[i].render(renderer);
    writer.flush();
    writer.close();
}

public ArrayList<OntClass> getKeywordClasses() {
    return keywordClasses;
}

public void setKeywordClasses(ArrayList<OntClass> keywordClasses) {
    this.keywordClasses = keywordClasses;
}

public LinkedHashMap<String, InnerStat> getKeywordsStat() {
    return keywordsStat;
}

public void setKeywordsStat(LinkedHashMap<String, InnerStat> keywordsStat) {
    this.keywordsStat = keywordsStat;
}

public File getAddOntology() {
    return addOntology;
}

public void setAddOntology(File addOntology) {
    this.addOntology = addOntology;
}

public String getAddOntologyContentType() {
    return addOntologyContentType;
}

public void setAddOntologyContentType(String addOntologyContentType) {
    this.addOntologyContentType = addOntologyContentType;
}

public String getAddOntologyFileName() {
    return addOntologyFileName;
}
public void setAddOntologyFileName(String addOntologyFileName) {
    this.addOntologyFileName = addOntologyFileName;
}

@Override
public void setServletRequest(HttpServletRequest servletRequest) {
    this.servletRequest = servletRequest;
}

public class InnerStat {
    public double minConfidence;
    public double maxConfidence;
    public double avgConfidence;
    public int nbCells;
    public double varianceConfidence;
}
import java.io.File;
import java.io.FileOutputStream;
import java.util.ArrayList;
import java.util.List;
import javax.servlet.http.HttpServletRequest;
import org.apache.struts2.interceptor.ServletRequestAware;
import org.openrdf.OpenRDFException;
import org.openrdf.repository.Repository;
import org.openrdf.repository.RepositoryConnection;
import org.openrdf.repository.http.HTTPRepository;
import org.openrdf.rio.RDFFormat;
import org.semanticweb.owlapi.apibinding.OWLManager;
import org.semanticweb.owlapi.model.IRI;
import org.semanticweb.owlapi.model.OWLAxiom;
import org.semanticweb.owlapi.model.OWLOntology;
import org.semanticweb.owlapi.model.OWLOntologyCreationException;
import org.semanticweb.owlapi.model.OWLOntologyFormat;
import org.semanticweb.owlapi.reasoner.ConsoleProgressMonitor;
import org.semanticweb.owlapi.reasoner.InferenceType;
import org.semanticweb.owlapi.reasoner.OWLReasoner;
import org.semanticweb.owlapi.reasoner.OWLReasonerConfiguration;
import org.semanticweb.owlapi.reasoner.OWLReasonerFactory;
import org.semanticweb.owlapi.reasoner.SimpleConfiguration;
import org.semanticweb.owlapi.util.InferredAxiomGenerator;
import org.semanticweb.owlapi.util.InferredClassAssertionAxiomGenerator;
import org.semanticweb.owlapi.util.InferredDisjointClassesAxiomGenerator;
import org.semanticweb.owlapi.util.InferredEquivalentClassAxiomGenerator;
import org.semanticweb.owlapi.util.InferredOntologyGenerator;
import org.semanticweb.owlapi.util.InferredPropertyAssertionGenerator;
import org.semanticweb.owlapi.util.InferredSubClassAxiomGenerator;

public class FileUploadAction extends ActionSupport implements ServletRequestAware {
    private File userImage;
    private String userImageContentType;
    private String userImageFileName;
    private HttpServletRequest servletRequest;

    public String execute() {
        try {
            String filePath = servletRequest.getSession().getServletContext().getRealPath("/");
            System.out.println("Server path:" + filePath);
        } catch (Exception e) {
            e.printStackTrace();
        }
        return null;
    }
}

import com.hp.hpl.jena.rdf.model.InfModel;
import com.hp.hpl.jena.rdf.model.Model;
import com.hp.hpl.jena.rdf.model.ModelFactory;
import com.hp.hpl.jena.reasoner.ReasonerRegistry;
import com.hp.hpl.jena.util.FileManager;
import com.opensymphony.xwork2.ActionSupport;
File fileToCreate = new File("D:/temp/", this.userImageFileName);

// reasoning
// doOwlApiReasoning();
Model data = FileManager.get().loadModel(this.userImage.getAbsolutePath());
com.hp.hpl.jena.reasoner.Reasoner reasoner = ReasonerRegistry.getOWLReasoner();
InfModel infmodel = ModelFactory.createInfModel(reasoner, data);
FileOutputStream fos = new FileOutputStream(fileToCreate);
infmodel.write(fos, "RDF/XML-ABBREV");

// storing uploaded ontology to repository
String baseURI = "http://www.helpdesk.org/ontontology2";
String sesameServer = "http://localhost:8080/openrdf-sesame";
String repositoryID = "1";
Repository myRepository = new HTTPRepository(sesameServer, repositoryID);
myRepository.initialize();
try {
    RepositoryConnection con = myRepository.getConnection();
    try {
        con.add(this.userImage, baseURI, RDFFormat.RDFXML);
    }
    finally {
        con.close();
    }
} catch (OpenRDFException e) {
    // handle exception
}
// FileUtils.copyFile(this.userImage, fileToCreate);
} catch (Exception e) {
    e.printStackTrace();
    addActionError(e.getMessage());
    return INPUT;
} else {
    return SUCCESS;
}
*/
private void doOwlApiReasoning() throws OWLOntologyStorageException {
    // perform reasoning
    try {
        OWLOntologyManager manager = OWLManager.createOWLOntologyManager();
        OWLOntology ont = manager.loadOntologyFromOntologyDocument(this.userImage);
        OWLOntologyFormat format = manager.getOntologyFormat(ont);
        System.out.println(" [FORMAT]: " + format);
//OWLReasonerFactory reasonerFactory = new PelletReasonerFactory();
ConsoleProgressMonitor progressMonitor = new ConsoleProgressMonitor();
OWLReasonerConfiguration conf = new SimpleConfiguration(progressMonitor);
OWLReasoner reasoner = reasonerFactory.createReasoner(ont, conf);
reasoner.precomputeInferences(InferenceType.CLASS_ASSERTIONS);
List<InferredAxiomGenerator<? extends OWLAXiom>> gens = new ArrayList<InferredAxiomGenerator<? extends OWLAXiom>>();
gens.add(new InferredSubClassAxiomGenerator());
gens.add(new InferredClassAssertionAxiomGenerator());
gens.add(new InferredPropertyAssertionGenerator());
gens.add(new InferredEquivalentClassAxiomGenerator());
gens.add(new InferredDisjointClassesAxiomGenerator());
OWLOntology infOnt = manager.createOntology();
InferredOntologyGenerator iog = new InferredOntologyGenerator(reasoner, gens);
iog.fillOntology(manager, infOnt);
manager.saveOntology(infOnt, IRI.create("file:///D:/temp/inferredont.owlapi");

boolean consistent = reasoner.isConsistent();
System.out.println("[CONSISTENT]: " + consistent);
System.out.println("\n");

File file = new File("/tmp/local.owl");
manager.saveOntology(ont, IRI.create(file.toURI()));
}
}
*/

public File getUserImage() {
    return userImage;
}

public void setUserImage(File userImage) {
    this.userImage = userImage;
}

public String getUserImageContentType() {
    return userImageContentType;
}

public void setUserImageContentType(String userImageContentType) {
    this.userImageContentType = userImageContentType;
}

public String getUserImageFileName() {
    return userImageFileName;
public void setUserImageFileName(String userImageFileName) {
    this.userImageFileName = userImageFileName;
}

@Override
public void setServletRequest(HttpServletRequest servletRequest) {
    this.servletRequest = servletRequest;
}
package org.semantichelpdesk.semweb;

import java.util.HashSet;
import java.util.Iterator;
import java.util.LinkedList;
import java.util.List;
import java.util.Set;
import com.hp.hpl.jena.ontology.OntTools;
import com.hp.hpl.jena.rdf.model.Model;
import com.hp.hpl.jena.rdf.model.RDFNode;
import com.hp.hpl.jena.rdf.model.Resource;
import com.hp.hpl.jena.rdf.model.Statement;
import com.hp.hpl.jena.util.iterator.Filter;

public class JenaOntToolsEx extends OntTools {
    public static Path findShortestPath( Model m, Resource start, RDFNode end, Filter<Statement> onPath ) {
        List<Path> bfs = new LinkedList<Path>();
        Set<Resource> seen = new HashSet<Resource>();

        // initialise the paths
        for (Iterator<Statement> i = m.listStatements( start, null, (RDFNode) null ).filterKeep( onPath ); i.hasNext(); ) {
            bfs.add( new PathEx().append( i.next() ) );
        }
        for (Iterator<Statement> i = m.listStatements( (Resource) null, null, start ).filterKeep( onPath ); i.hasNext(); ) {
            bfs.add( new PathEx().append( i.next() ) );
        }
        seen.add(start);

        // search
        PathEx solution = null;
        while (solution == null && !bfs.isEmpty()) {
            PathEx candidate = (PathEx) bfs.remove( 0 );

            if (candidate.hasTerminus( end ) || candidate.hasHead( end )) {
                solution = candidate;
            } else {
                Resource terminus = candidate.getTerminalResource();
                if (terminus != null && !seen.contains(terminus)) {
                    seen.add( terminus );
                // get statements where terminus is a subject
                    for (Iterator<Statement> i = terminus.listProperties().filterKeep( onPath ); i.hasNext(); ) {
                        Statement link = i.next();
                        // no looping allowed, so we skip this link if it takes us to a node we've seen
                        if (!seen.contains( link.getObject() )) {
                            bfs.add( candidate.append( link ) );
                        }
                    }
                // get statements where terminus is an object
            }
        }
    }
}
for (Iterator<Statement> i = m.listStatements(
    (Resource) null, null, terminus); i.hasNext(); ) {
    Statement link = i.next();
    // no looping allowed, so we skip this link if it
takes us to a node we've seen
    if (!seen.contains(link.getSubject())) {
        bfs.add(candidate.append(link));
    }
}

Resource head = candidate.getHeadResource();
if (head != null && !seen.contains(head)) {
    seen.add(head);
    // get statements where head is a subject
    for (Iterator<Statement> i = head.listProperties().filterKeep(onPath); i.hasNext(); ) {
        Statement link = i.next();
        // no looping allowed, so we skip this link if it
takes us to a node we've seen
        if (!seen.contains(link.getObject())) {
            bfs.add(candidate.append(link));
        }
    }
    // get statements where head is an object
    for (Iterator<Statement> i = m.listStatements(
        (Resource) null, null, head); i.hasNext(); ) {
        Statement link = i.next();
        // no looping allowed, so we skip this link if it
takes us to a node we've seen
        if (!seen.contains(link.getSubject())) {
            bfs.add(candidate.append(link));
        }
    }
}

return solution;

public static class PathEx extends Path {

    public PathEx() {
        super();
    }

    public PathEx(Path basePath) {
        super(basePath);
    }

    /** Answer a new Path whose elements are this Path with
     * <code>s</code> added at the end */
    public Path append(Statement s) {
        Path newPath = new PathEx(this);
        newPath.add(s);
        return newPath;
    }

    /** Answer true if the last link on the path has subject equal to
     * <code>n</code> */
    public boolean hasHead(RDFNode n) {
        return n != null && n.equals(getHead());
    }

}
/** Answer the subject Resource at the end of the path, if defined, or null */
public RDFNode getHead() {
    return size() > 0 ? get( size() - 1 ).getSubject() : null;
}

/** Answer the resource at the end of the path, if defined, or null */
public Resource getHeadResource() {
    RDFNode n = getHead();
    return (n != null && n.isResource()) ? (Resource) n : null;
}
package org.semantichelpdesk.struts2;

import java.util.ArrayList;
import java.util.LinkedHashMap;
import java.util.List;
import javax.servlet.http.HttpServletRequest;

import org.apache.struts2.interceptor.ServletRequestAware;
import org.openrdf.OpenRDFException;
import org.openrdf.model.Resource;
import org.openrdf.model.Statement;
import org.openrdf.model.URI;
import org.openrdf.model.ValueFactory;
import org.openrdf.model.vocabulary.RDFS;
import org.openrdf.repository.Repository;
import org.openrdf.repository.RepositoryConnection;
import org.openrdf.repository.RepositoryException;
import org.openrdf.repository.http.HTTPRepository;
import com.hp.hpl.jena.vocabulary.OWL;
import com.hp.hpl.jena.vocabulary.RDF;
import com.opensymphony.xwork2.ActionSupport;

public class ListOntologiesAction extends ActionSupport implements ServletRequestAware {
    private HttpServletRequest servletRequest;
    private SettingsAction settings = new SettingsAction();
    private LinkedHashMap<String, String> ontologies = new LinkedHashMap<String, String>();

    public LinkedHashMap<String, String> getOntologies() {
        return ontologies;
    }

    public void setOntologies(LinkedHashMap<String, String> ontologies) {
        this.ontologies = ontologies;
    }

    public String execute() {
        try {
            String sesameServer = settings.getRep_address() + "/openrdf-sesame";
            String repositoryID = settings.getRep_id();

            Repository myRepository = new HTTPRepository(sesameServer, repositoryID);
            myRepository.initialize();
            // storing uploaded ontology to repository
            ValueFactory f = myRepository.getValueFactory();
            URI staticContext = f.createURI("http://www.helpdesk.org/static");
            URI deductedContext =

        }
    }
}

ListOntologiesAction.java
f.createURI("http://www.helpdesk.org/deducted");
try {
    RepositoryConnection con = myRepository.getConnection();
    try {
        RepositoryResult<Statement> statements = con
            .getStatements((Resource)null, org.openrdf.model.vocabulary.RDF.TYPE
            , org.openrdf.model.vocabulary.OWL.ONTOLOGY
            , true, staticContext);
        while (statements.hasNext()) {// output list of ontologies
            Statement st = statements.next();
            String ont_address = "/openrdf-workbench/repositories/" + repositoryID
                               + "/explore?resource=<";
            ontologies.put(st.getSubject().toString(), ont_address);
        }
    } finally {
        con.close();
    }
} catch (OpenRDFException e) {
    // handle exception
}
} catch (Exception e) {
    e.printStackTrace();
    addActionError(e.getMessage());
    return INPUT;
} return SUCCESS;

@Override
public void setServletRequest(HttpServletRequest servletRequest) {
    this.servletRequest = servletRequest;
    }
LoginAction.java

package org.semantichelpdesk.struts2;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpSession;

import org.apache.catalina.Session;
import org.apache.struts2.interceptor.ServletRequestAware;

import com.opensymphony.xwork2.ActionSupport;

public class LoginAction extends ActionSupport implements ServletRequestAware{
    private HttpServletRequest servletRequest;
    private String username = "admin";
    private String password = "";

    public String execute() {
        if (this.username.equals("admin")
            && this.password.equals("1234")) {
            HttpSession session = servletRequest.getSession(true);
            session.setAttribute("authenticated", 1);
            return "success";
        } else {
            addActionError(getText("error.login"));
            return "error";
        }
    }

    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    @Override
    public void setServletRequest(HttpServletRequest servletRequest) {
        this.servletRequest = servletRequest;
    }
}
package org.semantichelpdesk.struts2;

import java.util.prefs.Preferences;
import com.opensymphony.xwork2.ActionSupport;

public class SettingsAction extends ActionSupport{
    // Preference keys for this package
    private static final String REPOSITORY_ADDRESS = "rep_address";
    private static final String REPOSITORY_ID = "rep_id";
    private static final String MASTER_ONTOLOGY_BASE = "ms_ont_base";
    private static final String MASTER_ONTOLOGY_ROOT = "ms_ont_root";
    private static final String HIDE_SHORTEST_PATHS = "hide_paths";

    Preferences prefs = Preferences.systemNodeForPackage(SettingsAction.class);

    private String rep_address;
    private String rep_id;
    private String ms_ont_base;
    private String ms_ont_root;
    private boolean hide_paths;

    public boolean isHide_paths() {
        this.hide_paths = Boolean.parseBoolean(prefs.get(HIDE_SHORTEST_PATHS, "false"));
        return hide_paths;
    }

    public void setHide_paths(boolean hide_paths) {
        prefs.put(HIDE_SHORTEST_PATHS, String.valueOf(hide_paths));
        this.hide_paths = hide_paths;
    }

    public String getRep_address() {
        this.rep_address = prefs.get(REPOSITORY_ADDRESS, getText("menu.sesame.server.url"));
        return rep_address;
    }

    public void setRep_address(String rep_address) {
        prefs.put(REPOSITORY_ADDRESS, rep_address);
        this.rep_address = rep_address;
    }

    public String getRep_id() {
        this.rep_id = prefs.get(REPOSITORY_ID, getText("sesame.server.master.rep"));
        return rep_id;
    }

    public void setRep_id(String rep_id) {
        prefs.put(REPOSITORY_ID, rep_id);
        this.rep_id = rep_id;
    }
}
public String getMs_ont_base() {
    this.ms_ont_base = prefs.get(MASTER_ONTOLOGY_BASE,
    getText("master.ontology.base.uri"));
    return ms_ont_base;
}

public void setMs_ont_base(String ms_ont_base) {
    prefs.put(MASTER_ONTOLOGY_BASE,ms_ont_base);
    this.ms_ont_base = ms_ont_base;
}

public String getMs_ont_root() {
    this.ms_ont_root = prefs.get(MASTER_ONTOLOGY_ROOT,
    getText("master.ontology.root.class"));
    return ms_ont_root;
}

public void setMs_ont_root(String ms_ont_root) {
    prefs.put(MASTER_ONTOLOGY_ROOT,ms_ont_root);
    this.ms_ont_root = ms_ont_root;
}

public String execute() {
    return SUCCESS;
}
}
ontologyn1.owl

<!DOCTYPE rdf:RDF [
  <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
  <!ENTITY swrl "http://www.w3.org/2003/11/swrl#" >
  <!ENTITY owl2 "http://www.w3.org/2006/12/owl2#" >
  <!ENTITY swrlx "http://www.w3.org/2003/11/swrlx#" >
  <!ENTITY swrlb "http://www.w3.org/2003/11/swrlb#" >
  <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
  <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
  <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
]>

<rdf:RDF xmlns="http://www.helpdesk.org/ontology1#"
  xml:base="http://www.helpdesk.org/ontology1"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:swrl="http://www.w3.org/2003/11/swrl#"
  xmlns:swrlx="http://www.w3.org/2003/11/swrlx#"
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
  xmlns:swrlb="http://www.w3.org/2003/11/swrlb#"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:owl2="http://www.w3.org/2006/12/owl2#">
  <owl:Ontology rdf:about=""/>

  <!--
  //////////////////////////////////////////////////////////////////////
  // Classes
  //////////////////////////////////////////////////////////////////////
  -->

  <!-- http://www.helpdesk.org/ontology1#Abnormal_Image -->

  <owl:Class rdf:about="#Abnormal_Image">
    <rdfs:subClassOf rdf:resource="#Monitor_Problem"/>
  </owl:Class>
<!-- http://www.helpdesk.org/ontology1#Account_Suspension_Guidance -->
<owl:Class rdf:about="#Account_Suspension_Guidance">
  <rdfs:subClassOf rdf:resource="#Email_Account_Issue"/>
  <rdfs:subClassOf rdf:resource="#PC_Account_Issue"/>
</owl:Class>

<!-- http://www.helpdesk.org/ontology1#Account_Termination_Guidance -->
<owl:Class rdf:about="#Account_Termination_Guidance">
  <rdfs:subClassOf rdf:resource="#Email_Account_Issue"/>
  <rdfs:subClassOf rdf:resource="#PC_Account_Issue"/>
</owl:Class>

<!-- http://www.helpdesk.org/ontology1#Adobe_PDF_Problem -->
<owl:Class rdf:about="#Adobe_PDF_Problem">
  <rdfs:subClassOf rdf:resource="#Standard_Software_Problem"/>
</owl:Class>

<!-- http://www.helpdesk.org/ontology1#Button_Not_Responding -->
<owl:Class rdf:about="#Button_Not_Responding">
  <rdfs:subClassOf rdf:resource="#Mouse_Problem"/>
</owl:Class>

<!-- http://www.helpdesk.org/ontology1#CD_DVD_ROM_Problem -->
<owl:Class rdf:about="#CD_DVD_ROM_Problem">
  <rdfs:subClassOf rdf:resource="#Standard_Hardware_Problem"/>
</owl:Class>

<!-- http://www.helpdesk.org/ontology1#Camera_Problem -->
<owl:Class rdf:about="#Camera_Problem">
  <rdfs:subClassOf rdf:resource="#Non_Standard_Hardware_Problem"/>
</owl:Class>

<!-- http://www.helpdesk.org/ontology1#Cannot_Close -->
<owl:Class rdf:about="#Cannot_Close">
  <rdfs:subClassOf rdf:resource="#CD_DVD_ROM_Problem"/>
</owl:Class>
<!-- http://www.helpdesk.org/ontology1#Cannot_Completely_Load -->
<owl:Class rdf:about="#Cannot_Completely_Load">
  <rdfs:subClassOf rdf:resource="#Enterprise_Website_Problem"/>
</owl:Class>

<!-- http://www.helpdesk.org/ontology1#Cannot_Login -->
<owl:Class rdf:about="#Cannot_Login">
  <rdfs:subClassOf rdf:resource="#PC_Account_Issue"/>
  <rdfs:subClassOf rdf:resource="#Remote_Server_Problem"/>
</owl:Class>

<!-- http://www.helpdesk.org/ontology1#Cannot_Open -->
<owl:Class rdf:about="#Cannot_Open">
  <rdfs:subClassOf rdf:resource="#CD_DVD_ROM_Problem"/>
</owl:Class>

<!-- http://www.helpdesk.org/ontology1#Cannot_Play -->
<owl:Class rdf:about="#Cannot_Play">
  <rdfs:subClassOf rdf:resource="#CD_DVD_ROM_Problem"/>
</owl:Class>

<!-- http://www.helpdesk.org/ontology1#Cannot_Record -->
<owl:Class rdf:about="#Cannot_Record">
  <rdfs:subClassOf rdf:resource="#CD_DVD_ROM_Problem"/>
</owl:Class>

<!-- http://www.helpdesk.org/ontology1#Cannot_Shut_Down -->
<owl:Class rdf:about="#Cannot_Shut_Down">
  <rdfs:subClassOf rdf:resource="#Hard_Drive_Problem"/>
</owl:Class>

<!-- http://www.helpdesk.org/ontology1#Cannot_Start -->
<owl:Class rdf:about="#Cannot_Start">
  <rdfs:subClassOf rdf:resource="#Hard_Drive_Problem"/>
  <rdfs:subClassOf rdf:resource="#Performance_Problem"/>
</owl:Class>

<!-- http://www.helpdesk.org/ontology1#Change_Password_Guidance -->

220
<owl:Class rdf:about="#Change_Password_Guidance">
  <rdfs:subClassOf rdf:resource="#Password_Issue"/>
</owl:Class>

<!-- http://www.helpdesk.org/ontology1#Corrupted -->
<owl:Class rdf:about="#Corrupted">
  <rdfs:subClassOf rdf:resource="#File_Problem"/>
</owl:Class>

<!-- http://www.helpdesk.org/ontology1#Cursor_Frozen -->
<owl:Class rdf:about="#Cursor_Frozen">
  <rdfs:subClassOf rdf:resource="#Mouse_Problem"/>
</owl:Class>

<!-- http://www.helpdesk.org/ontology1#Email_Account_Issue -->
<owl:Class rdf:about="#Email_Account_Issue">
  <rdfs:subClassOf rdf:resource="#Account_Issue"/>
</owl:Class>

<!-- http://www.helpdesk.org/ontology1#Enterprise_Website_Problem -->
<owl:Class rdf:about="#Enterprise_Website_Problem">
  <rdfs:subClassOf rdf:resource="#Website_Problem"/>
</owl:Class>

<!-- http://www.helpdesk.org/ontology1#Entire_Keyboard_Nor_Responding -->
<owl:Class rdf:about="#Entire_Keyboard_Nor_Responding">
  <rdfs:subClassOf rdf:resource="#Keyboard_Problem"/>
</owl:Class>

<!-- http://www.helpdesk.org/ontology1#Equipment_Moving_Guidance -->
<owl:Class rdf:about="#Equipment_Moving_Guidance">
  <rdfs:subClassOf rdf:resource="#IT_Admin_Issue"/>
</owl:Class>

<!-- http://www.helpdesk.org/ontology1#Error_Message -->
<owl:Class rdf:about="#Error_Message"/>
<owl:Class rdf:about="#Hardware_Problem">
  <rdfs:subClassOf rdf:resource="#Help_Desk_Enquiry"/>
</owl:Class>

<owl:Class rdf:about="#Help_Desk_Enquiry"/>

<owl:Class rdf:about="#Hung">
  <rdfs:subClassOf rdf:resource="#Hard_Drive_Problem"/>
</owl:Class>

<owl:Class rdf:about="#IT_Admin_Issue">
  <rdfs:subClassOf rdf:resource="#Help_Desk_Enquiry"/>
</owl:Class>

<owl:Class rdf:about="#IT_Product_Purchase_Guidance">
  <rdfs:subClassOf rdf:resource="#IT_Admin_Issue"/>
</owl:Class>

<owl:Class rdf:about="#Internet_Explorer_Problem">
  <rdfs:subClassOf rdf:resource="#Standard_Software_Problem"/>
</owl:Class>

<owl:Class rdf:about="#Keyboard_Problem">
  <rdfs:subClassOf rdf:resource="#Standard_Hardware_Problem"/>
</owl:Class>

<owl:Class rdf:about="#Loading_Speed"/>
<owl:Class rdf:about="#Loading_Speed">
<rdfs:subClassOf rdf:resource="#Enterprise_Website_Problem"/>
</owl:Class>

<!-- http://www.helpdesk.org/ontology1#Long_Delay -->
<owl:Class rdf:about="#Long_Delay">
<rdfs:subClassOf rdf:resource="#Keyboard_Problem"/>
</owl:Class>

<!-- http://www.helpdesk.org/ontology1#MS_Access_Problem -->
<owl:Class rdf:about="#MS_Access_Problem">
<rdfs:subClassOf rdf:resource="#MS_Problem"/>
</owl:Class>

<!-- http://www.helpdesk.org/ontology1#MS_Excel_Problem -->
<owl:Class rdf:about="#MS_Excel_Problem">
<rdfs:subClassOf rdf:resource="#MS_Problem"/>
</owl:Class>

<!-- http://www.helpdesk.org/ontology1#MS_Problem -->
<owl:Class rdf:about="#MS_Problem">
<rdfs:subClassOf rdf:resource="#Standard_Software_Problem"/>
</owl:Class>

<!-- http://www.helpdesk.org/ontology1#MS_Outlook_Problem -->
<owl:Class rdf:about="#MS_Outlook_Problem">
<rdfs:subClassOf rdf:resource="#MS_Problem"/>
</owl:Class>

<!-- http://www.helpdesk.org/ontology1#MS_PowerPoint_Problem -->
<owl:Class rdf:about="#MS_PowerPoint_Problem">
<rdfs:subClassOf rdf:resource="#MS_Problem"/>
</owl:Class>

<!-- http://www.helpdesk.org/ontology1#MS_Word_Problem -->
<owl:Class rdf:about="#MS_Word_Problem">
<rdfs:subClassOf rdf:resource="#MS_Problem"/>
</owl:Class>

<!-- http://www.helpdesk.org/ontology1#McAfee_Virus_Scan_Problem -->
<owl:Class rdf:about="#McAfee_Virus_Scan_Problem">
    <rdfs:subClassOf rdf:resource="#Standard_Software_Problem"/>
</owl:Class>

<!-- http://www.helpdesk.org/ontology1#Missing -->
<owl:Class rdf:about="#Missing">
    <rdfs:subClassOf rdf:resource="#File_Problem"/>
</owl:Class>

<!-- http://www.helpdesk.org/ontology1#Missing_Folder -->
<owl:Class rdf:about="#Missing_Folder">
    <rdfs:subClassOf rdf:resource="#Remote_Server_Problem"/>
</owl:Class>

<!-- http://www.helpdesk.org/ontology1#Mobile_Gadget_Problem -->
<owl:Class rdf:about="#Mobile_Gadget_Problem">
    <rdfs:subClassOf rdf:resource="#Non_Standard_Hardware_Problem"/>
</owl:Class>

<!-- http://www.helpdesk.org/ontology1#Monitor_Problem -->
<owl:Class rdf:about="#Monitor_Problem">
    <rdfs:subClassOf rdf:resource="#Standard_Hardware_Problem"/>
</owl:Class>

<!-- http://www.helpdesk.org/ontology1#Mouse_Problem -->
<owl:Class rdf:about="#Mouse_Problem">
    <rdfs:subClassOf rdf:resource="#Standard_Hardware_Problem"/>
</owl:Class>

<!-- http://www.helpdesk.org/ontology1#Movement_Too_Fast -->
<owl:Class rdf:about="#Movement_Too_Fast">
    <rdfs:subClassOf rdf:resource="#Mouse_Problem"/>
</owl:Class>
<!-- http://www.helpdesk.org/ontology1#Movement_Too_Slow -->
<owl:Class rdf:about="#Movement_Too_Slow">
  <rdfs:subClassOf rdf:resource="#Mouse_Problem"/>
</owl:Class>

<!-- http://www.helpdesk.org/ontology1#No_Image -->
<owl:Class rdf:about="#No_Image">
  <rdfs:subClassOf rdf:resource="#Monitor_Problem"/>
</owl:Class>

<!-- http://www.helpdesk.org/ontology1#No_Printout -->
<owl:Class rdf:about="#No_Printout">
  <rdfs:subClassOf rdf:resource="#Printer_Problem"/>
</owl:Class>

<!-- http://www.helpdesk.org/ontology1#Non_Enterprise_Website_Problem -->
<owl:Class rdf:about="#Non_Enterprise_Website_Problem">
  <rdfs:subClassOf rdf:resource="#Website_Problem"/>
</owl:Class>

<!-- http://www.helpdesk.org/ontology1#Non_Standard_Hardware_Installation_Guidance -->
<owl:Class rdf:about="#Non_Standard_Hardware_Installation_Guidance">
  <rdfs:subClassOf rdf:resource="#Hardware_Installation_Guidance"/>
</owl:Class>

<!-- http://www.helpdesk.org/ontology1#Non_Standard_Hardware_Problem -->
<owl:Class rdf:about="#Non_Standard_Hardware_Problem">
  <rdfs:subClassOf rdf:resource="#Hardware_Problem"/>
</owl:Class>

<!-- http://www.helpdesk.org/ontology1#Non_Standard_Software_Installation_Guidance -->
<owl:Class rdf:about="#Non_Standard_Software_Installation_Guidance">
  <rdfs:subClassOf rdf:resource="#Software_Installation_Guidance"/>
</owl:Class>
<!-- http://www.helpdesk.org/ontology1#Password_Syntax_Info -->
<owl:Class rdf:about="#Password_Syntax_Info">
  <rdfs:subClassOf rdf:resource="#Password_Issue"/>
</owl:Class>

<!-- http://www.helpdesk.org/ontology1#Performance_Problem -->
<owl:Class rdf:about="#Performance_Problem">
  <rdfs:subClassOf rdf:resource="#Software_Problem"/>
</owl:Class>

<!-- http://www.helpdesk.org/ontology1#Plugin_Not_Responding -->
<owl:Class rdf:about="#Plugin_Not_Responding">
  <rdfs:subClassOf rdf:resource="#Hard_Drive_Problem"/>
</owl:Class>

<!-- http://www.helpdesk.org/ontology1#Printer_Problem -->
<owl:Class rdf:about="#Printer_Problem">
  <rdfs:subClassOf rdf:resource="#Standard_Hardware_Problem"/>
</owl:Class>

<!-- http://www.helpdesk.org/ontology1#Remote_Server_Problem -->
<owl:Class rdf:about="#Remote_Server_Problem">
  <rdfs:subClassOf rdf:resource="#Other_Problem"/>
</owl:Class>

<!-- http://www.helpdesk.org/ontology1#Reset_Password_Guidance -->
<owl:Class rdf:about="#Reset_Password_Guidance">
  <rdfs:subClassOf rdf:resource="#Password_Issue"/>
</owl:Class>

<!-- http://www.helpdesk.org/ontology1#Retrieve_Password_Guidance -->
<owl:Class rdf:about="#Retrieve_Password_Guidance">
  <rdfs:subClassOf rdf:resource="#Password_Issue"/>
</owl:Class>
<!-- Generated by the OWL API (version 3.0.0) -->

<!-- http://www.helpdesk.org/ontology1#access_master_instance1 -->

<owl:NamedIndividual rdf:about="#access_master_instance1">
    <rdf:type rdf:resource="#MS_Access_Problem"/>
</owl:NamedIndividual>

<!-- Generated by the OWL API (version 3.0.0) -->

http://owlapi.sourceforge.net -->
extontology1.owl

<?xml version="1.0"?>

<!DOCTYPE rdf:RDF [
  <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
  <!ENTITY swrl "http://www.w3.org/2003/11/swrl#" >
  <!ENTITY owl2 "http://www.w3.org/2006/12/owl2#" >
  <!ENTITY swrlx "http://www.w3.org/2003/11/swrlx#" >
  <!ENTITY swrlb "http://www.w3.org/2003/11/swrlb#" >
  <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
  <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
  <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >]

<rdf:RDF xmlns="http://www.helpdesk.org/extontology1#"
  xml:base="http://www.helpdesk.org/extontology1"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:swrl="http://www.w3.org/2003/11/swrl#"
  xmlns:swrlx="http://www.w3.org/2003/11/swrlx#"
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
  xmlns:swrlb="http://www.w3.org/2003/11/swrlb#"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:owl2="http://www.w3.org/2006/12/owl2#">
  <owl:Ontology rdf:about=""/>

<!--

// Annotation properties
//

-->  
  <owl:AnnotationProperty rdf:about="&rdfs;comment"/>

<!--

// Classes
//

-->
<!-- http://www.helpdesk.org/extontology1#Access -->

<owl:Class rdf:about="#Access">
  <rdfs:subClassOf rdf:resource="#Office"/>
</owl:Class>

<!-- http://www.helpdesk.org/extontology1#Change_Animation -->

<owl:Class rdf:about="#Change_Animation">
  <rdfs:subClassOf rdf:resource="#PowerPoint"/>
</owl:Class>

<!-- http://www.helpdesk.org/extontology1#Change_Design -->

<owl:Class rdf:about="#Change_Design">
  <rdfs:subClassOf rdf:resource="#PowerPoint"/>
</owl:Class>

<!-- http://www.helpdesk.org/extontology1#Change_Transition -->

<owl:Class rdf:about="#Change_Transition">
  <rdfs:subClassOf rdf:resource="#PowerPoint"/>
</owl:Class>

<!-- http://www.helpdesk.org/extontology1#Create_Form -->

<owl:Class rdf:about="#Create_Form">
  <rdfs:subClassOf rdf:resource="#Access"/>
  <rdfs:comment rdf:datatype="&rdfs;Literal">Knowledge for Create Form subclass of Access of Office</rdfs:comment>
</owl:Class>

<!-- http://www.helpdesk.org/extontology1#Create_Function -->

<owl:Class rdf:about="#Create_Function">
  <rdfs:subClassOf rdf:resource="#Excel"/>
</owl:Class>

<!-- http://www.helpdesk.org/extontology1#Create_Micro -->

<owl:Class rdf:about="#Create_Micro">
  <rdfs:subClassOf rdf:resource="#Excel"/>
</owl:Class>
<!-- http://www.helpdesk.org/extontology1#Create_Query -->
<owl:Class rdf:about="#Create_Query">
  <rdfs:subClassOf rdf:resource="#Access"/>
</owl:Class>

<!-- http://www.helpdesk.org/extontology1#Create_Report -->
<owl:Class rdf:about="#Create_Report">
  <rdfs:subClassOf rdf:resource="#Access"/>
</owl:Class>

<!-- http://www.helpdesk.org/extontology1#Create_Table -->
<owl:Class rdf:about="#Create_Table">
  <rdfs:subClassOf rdf:resource="#Access"/>
</owl:Class>

<!-- http://www.helpdesk.org/extontology1#Create_Table_Of_Content -->
<owl:Class rdf:about="#Create_Table_Of_Content">
  <rdfs:subClassOf rdf:resource="#Word"/>
</owl:Class>

<!-- http://www.helpdesk.org/extontology1#Create_VLookUp -->
<owl:Class rdf:about="#Create_VLookUp">
  <rdfs:subClassOf rdf:resource="#Excel"/>
</owl:Class>

<!-- http://www.helpdesk.org/extontology1#Customise_Start_Menu -->
<owl:Class rdf:about="#Customise_Start_Menu">
  <rdfs:subClassOf rdf:resource="#Window_7"/>
</owl:Class>

<!-- http://www.helpdesk.org/extontology1#Customise_Taskbar -->
<owl:Class rdf:about="#Customise_Taskbar">
  <rdfs:subClassOf rdf:resource="#Window_7"/>
</owl:Class>
<!-- http://www.helpdesk.org/extontology1#Excel -->
<owl:Class rdf:about="#Excel">
  <rdfs:subClassOf rdf:resource="#Office"/>
</owl:Class>

<!-- http://www.helpdesk.org/extontology1#Microsoft_Knowledge -->
<owl:Class rdf:about="#Microsoft_Knowledge"/>

<!-- http://www.helpdesk.org/extontology1#Office -->
<owl:Class rdf:about="#Office">
  <rdfs:subClassOf rdf:resource="#Microsoft_Knowledge"/>
</owl:Class>

<!-- http://www.helpdesk.org/extontology1#PowerPoint -->
<owl:Class rdf:about="#PowerPoint">
  <rdfs:subClassOf rdf:resource="#Office"/>
</owl:Class>

<!-- http://www.helpdesk.org/extontology1#Use_Mail_Merge -->
<owl:Class rdf:about="#Use_Mail_Merge">
  <rdfs:subClassOf rdf:resource="#Word"/>
</owl:Class>

<!-- http://www.helpdesk.org/extontology1#Use_Track_Changes -->
<owl:Class rdf:about="#Use_Track_Changes">
  <rdfs:subClassOf rdf:resource="#Word"/>
</owl:Class>

<!-- http://www.helpdesk.org/extontology1#Window_7 -->
<owl:Class rdf:about="#Window_7">
  <rdfs:subClassOf rdf:resource="#Windows"/>
</owl:Class>

<!-- http://www.helpdesk.org/extontology1#Window_Vista -->
<owl:Class rdf:about="#Window_Vista">
  <rdfs:subClassOf rdf:resource="#Windows"/>
</owl:Class>
<!-- http://www.helpdesk.org/extontology1#Window_XP -->

<owl:Class rdf:about="#Window_XP">
  <rdfs:subClassOf rdf:resource="#Windows"/>
</owl:Class>

<!-- http://www.helpdesk.org/extontology1#Windows -->

<owl:Class rdf:about="#Windows">
  <rdfs:subClassOf rdf:resource="#Microsoft_Knowledge"/>
</owl:Class>

<!-- http://www.helpdesk.org/extontology1#Word -->

<owl:Class rdf:about="#Word">
  <rdfs:subClassOf rdf:resource="#Office"/>
</owl:Class>

<!-- Generated by the OWL API (version 3.0.0) 
http://owlapi.sourceforge.net -->
<?xml version="1.0"?>

<!DOCTYPE rdf:RDF [  
<!ENTITY owl "http://www.w3.org/2002/07/owl#" >  
<!ENTITY swrl "http://www.w3.org/2003/11/swrl#" >  
<!ENTITY owl2 "http://www.w3.org/2006/12/owl2#" >  
<!ENTITY swrlx "http://www.w3.org/2003/11/swrlx#" >  
<!ENTITY swrlb "http://www.w3.org/2003/11/swrlb#" >  
<!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >  
<!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >  
<!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >  
]
>

<rdf:RDF xmlns="http://www.helpdesk.org/extontology2#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:swrl="http://www.w3.org/2003/11/swrl#">
  <owl:Ontology rdf:about="http://www.helpdesk.org/extontology2#"/>

  <!--
  /////////////////////////////////////////////////////////////////////////
  // Classes
  /////////////////////////////////////////////////////////////////////////
  -->

  <!-- http://www.helpdesk.org/extontology2#DVD_Player -->
  <owl:Class rdf:about="http://www.helpdesk.org/extontology2#DVD_Player">  
    <rdfs:subClassOf rdf:resource="http://www.helpdesk.org/extontology2#Visual_Device"/>
  </owl:Class>

  <!-- http://www.helpdesk.org/extontology2#Mobility_Device -->

<owl:Class rdf:about="http://www.helpdesk.org/extontology2#Mobility_Device">
  <rdfs:subClassOf rdf:resource="http://www.helpdesk.org/extontology2#Product"/>
</owl:Class>

<!-- http://www.helpdesk.org/extontology2#Product -->
<owl:Class rdf:about="http://www.helpdesk.org/extontology2#Product"/>

<!-- http://www.helpdesk.org/extontology2#Smart_Phone -->
<owl:Class rdf:about="http://www.helpdesk.org/extontology2#Smart_Phone">
  <rdfs:subClassOf rdf:resource="http://www.helpdesk.org/extontology2#Mobility_Device"/>
</owl:Class>

<!-- http://www.helpdesk.org/extontology2#TV -->
<owl:Class rdf:about="http://www.helpdesk.org/extontology2#TV">
  <rdfs:subClassOf rdf:resource="http://www.helpdesk.org/extontology2#Visual_Device"/>
</owl:Class>

<!-- http://www.helpdesk.org/extontology2#Tablet_Computer -->
<owl:Class rdf:about="http://www.helpdesk.org/extontology2#Tablet_Computer">
  <rdfs:subClassOf rdf:resource="http://www.helpdesk.org/extontology2#Mobility_Device"/>
</owl:Class>

<!-- http://www.helpdesk.org/extontology2#Visual_Device -->
<owl:Class rdf:about="http://www.helpdesk.org/extontology2#Visual_Device">
  <rdfs:subClassOf rdf:resource="http://www.helpdesk.org/extontology2#Product"/>
</owl:Class>

<owl:Class rdf:about="http://www.helpdesk.org/extontology2#Wifi_Problem">
  <rdfs:subClassOf rdf:resource="http://www.helpdesk.org/extontology2#Tablet_Computer"/>
</owl:Class>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:owl="http://www.w3.org/2002/07/owl#">
  <owl:Class
    rdf:about="http://www.helpdesk.org/extontology2#Setup_Problem">
    <rdfs:subClassOf
      rdf:resource="http://www.helpdesk.org/extontology2#Tablet_Computer"/>
  </owl:Class>
  <owl:Class
    rdf:about="http://www.helpdesk.org/extontology2#Touchscreen_Problem">
    <rdfs:subClassOf
      rdf:resource="http://www.helpdesk.org/extontology2#Smart_Phone"/>
  </owl:Class>
  <owl:Class
    rdf:about="http://www.helpdesk.org/extontology2#App_Installation_Problem">
    <rdfs:subClassOf
      rdf:resource="http://www.helpdesk.org/extontology2#Smart_Phone"/>
  </owl:Class>
</rdf:RDF>
extonology3.owl

<?xml version="1.0"?>

<!-- http://www.helpdesk.org/extontology3#Client_Error -->
<owl:Class rdf:about="http://www.helpdesk.org/extontology3#Client_Error">
  <rdfs:subClassOf rdf:resource="http://www.helpdesk.org/extontology3#HTTP_Error"/>
</owl:Class>

<!-- http://www.helpdesk.org/extontology3#FTP_Error -->
<!-- http://www.helpdesk.org/extontology3#Server_Error -->

<owl:Class
rdf:about="http://www.helpdesk.org/extontology3#Server_Error">
  <rdfs:subClassOf
    rdf:resource="http://www.helpdesk.org/extontology3#HTTP_Error"/>
</owl:Class>

<!-- http://www.helpdesk.org/extontology3#Web_Application_Error -->

<owl:Class
rdf:about="http://www.helpdesk.org/extontology3#Web_Application_Error"/>
</rdf:RDF>

<!-- Generated by the OWL API (version 3.2.0.1502) http://owlapi.sourceforge.net -->