University of Porto
Faculty of Engineering
Department of Computing and Electrical Engineering

Thesis presented for the fulfilment of the requirements necessary to complete the degree of PhD in Computing and Electrical Engineering

Ontology-based Services for Agents Interoperability

by

Andreia Malucelli

Supervisor: Professor Eugénio da Costa Oliveira
Co-supervisor: Professor Marcos Augusto Hochuli Schmeil

Porto, 2006

To my parents Olímpio and Glória Maria

Resumo

Esta tese propõe o desenvolvimento de serviços baseados em ontologias para promover a interoperabilidade entre agentes heterogéneos pertencentes a sistemas multiagente (SMA) dedicados ao comércio electrónico entre empresas (B2B)[1]. Estes serviços destinam-se a integrar uma Instituição Electrónica, modelada através de um SMA, que suportará a negociação automática necessária à formação de Empresas Virtuais.

Os serviços baseados em ontologias desenvolvidos são prestados através do Agente de Serviços Ontológicos, o qual é responsável por fornecer aos restantes agentes do SMA, que representam as empresas clientes e fornecedoras, apoio sobre como interpretar e conduzir a negociação de produtos específicos, facilitando o entendimento e propiciando a realização de negócios.

Os problemas de interoperabilidade tratados nesta tese são de dois tipos: (*i*) de índole terminológica e (*ii*) provocados pela falta de padronização ao nível das unidades. Neste domínio, as ontologias, enquanto forma de representação de conhecimento, desempenham um papel importante. Enquanto que, no caso dos SMA homogéneos, a adopção de uma ontologia comum é condição suficiente para uma comunicação eficaz, no caso dos SMA abertos, a heterogeneidade intrínseca dos agentes, onde cada agente possui um vocabulário próprio definido na sua ontologia privada, gera conflitos terminológicos. A utilização, em simultâneo, de sistemas de unidades distintos para representar os mesmos atributos de um dado produto provoca igualmente o desentendimento entre agentes.

A abordagem adoptada tem como objectivo criar uma metodologia que permita determinar as eventuais correspondências entre as descrições de produtos de dois agentes, *i.e.*, que detecte as similaridades léxica e semântica entre duas descrições representadas através de ontologias diferentes. O algoritmo que estabelece a correspondência léxica examina as características (atributos), as relações e as descrições dos produtos. Os atributos são classificados segundo o tipo de

[1] *Business-to-business* (B2B) *e-commerce*.

dados e a relação *tem-parte*. Os nomes dos produtos (conceitos) são comparados através do algoritmo de similaridade semântica baseado em WordNet de Leacock-Chodrow. A resolução dos problemas provocados pela falta de padronização é efectuada através de um conjunto de serviços de conversão de unidades que tomam a forma de serviços *Web*.

O processo de negociação concebido combina dois protocolos de interacção: o protocolo de Redes de Contrato e o protocolo de Interacção Ontológica. O protocolo de Redes de Contrato, proposto pela Foundation for Intelligent Physical Agents (FIPA), representa o enquadramento geral da negociação de produtos entre agentes. O protocolo de Interacção Ontológica é o protocolo adoptado para a resolução dos problemas de interoperabilidade, *i.e.*, que os agentes utilizam para interagir com o Agente de Serviços Ontológicos sempre que desconhecem o significado de parte ou totalidade do conteúdo de alguma mensagem.

Por último, para permitir a criação de agentes com ontologias diferentes de uma forma integrada e automática, desenvolveu-se uma metodologia que integra diversas ferramentas e que permite acompanhar todo o ciclo de desenvolvimento, depuração e actualização do protótipo desenvolvido.

As metodologias propostas, os sucessivos passos da implementação, os testes realizados, assim como os resultados obtidos são apresentados e discutidos neste documento.

Abstract

In this thesis we propose the development of ontology-based services to facilitate the interoperability among heterogeneous agents that belong to a multi-agent system (MAS) dedicated to business-to-business (B2B) electronic commerce. These services will be integrated in an Electronic Institution to support the automatic negotiation required to the formation of Virtual Enterprises (VE).

Our ontology-based services are provided through an Ontology-based Services Agent which is responsible for providing the agents within the MAS, representing customer and supplier enterprises, with suggestions of how to interpret and negotiate specific products, leading to meaningful conversations and making agreements possible.

We address two types of interoperability problems: (*i*) problems caused by the use of distinct terminology; and (*ii*) problems generated by the lack of standardization. In this context, ontologies play an important role since they provide the vocabularies used by the agents. Whereas, in homogeneous MAS, the adoption of a common ontology guarantees the intelligibility of the communication between agents, in open MAS, where the agents are intrinsically heterogeneous and have different private ontologies, the interagent communication is affected by unknown and, thus, incomprehensible terminology, as well as, by the *ad-hoc* use of disparate unit systems to represent the attributes of products.

The approach presented in this thesis aims at creating a methodology that matches two products represented in different ontologies through a lexical and semantic similarity evaluation. The lexical measure algorithm compares the characteristics (attributes), relations and descriptions of both products. The attributes are classified according to their data types and to the *has-part* relation. The concepts (names) of both products are compared using the Leacock-Chodrow WordNet-based semantic similarity measure algorithm. To handle the question of the lack of standardization, a set of units conversion Web services are provided.

The implementation of our negotiation process combines the Foundation for Physical Intelligent Agents (FIPA) Contract Net Interaction Protocol with an additional protocol called Ontology Interaction Protocol (OIP). The former represents the general scenario of agents trading goods proposed by FIPA. The latter implements the protocol necessary for solving the interoperability problems, when agents are interacting and requesting some of the provided services. Finally, to implement our MAS prototype, which is made of a set of agents with different ontologies, using an automated and integrated approach, we adopt a new development methodology that integrates several tools and allows ontology updating.

The proposed methodologies, the successive steps of the implementation, the tests carried out and the results obtained are presented and discussed throughout this document.

Résumé

Cette thèse décrit le développement de services ayant comme base des ontologies. Ces services ont l'objectif de faciliter l'interopérabilité des agents dans le contexte du commerce électronique entre entreprises (B2B)[2]. Les services proposés seront intégrés dans une Institution Électronique au système multi-agent de façon à soutenir la négociation automatique nécessaire à la formation d'Entreprises Virtuelles (EV). La formation d'une EV demande la négociation intensive parmi les différentes entreprises en jeu, à savoir, l'entreprise qui cherche à engager un service (modelée comme agent client) et les entreprises capables d'accomplir ce service (modelées comme agents fournisseurs). Le groupe de tous les agents constituera, finalement, l'EV. Dans notre cas particulier, une fois que le domaine d'application est celui du secteur automobile, l'EV finale sera une EV d'assemblage automobile.

Les agents d'entreprise (agents clients et agents fournisseurs) sont des entités autonomes, potentiellement hétérogènes, d'origines diverses, étant libres d'entrer et sortir du système quand ils le souhaitent. Dans un tel scénario on trouve fréquemment des problèmes d'interopérabilité demandant des techniques spécifiques de résolution. Notre effort s'est concentré sur la résolution de ce genre de problèmes qui prennent place quand des agents aux ontologies différentes s'engagent en négociation.

Les méthodologies développées sont mises à disposition a travers le « Ontology-based Services Agent » sous la forme d'un ensemble de services. Ces services offrent des fonctions soutenant l'activité de négociation entre agents acheteurs et agents fournisseurs, possédant des ontologies de domaine différentes.

Les ontologies des agents ont un rôle très important dans l'interaction concernant les systèmes multi-agent car elles stipulent le vocabulaire de communication des agents. Dans un milieu ouvert, comme celui du domaine B2B, il est improbable de trouver deux agents utilisant la même ontologie. Par conséquent,

[2] « Business to Business (B2B) e-commerce ».

des agents ayant des vocabulaires particuliers différents, vont très probablement parler d'un même concept avec des termes différents, ou bien, représenter les caractéristiques des produits en monnaies et/ou unités différentes.

Notre approche veut créer une méthodologie permettant de accoupler des produits originaires d'ontologies différentes en mesurant leur similarité lexicale et sémantique.

L'algorithme de mesure lexicale prend une paire de concepts (un produit de chaque ontologie) et fait la comparaison de leurs caractéristiques (attributs), relations et descriptions. Pour les attributs, cette comparaison s'effectue selon la nature des donnés. Dans le cas des relations, la comparaison prend en considérération la relation « has part ». L'algorithme de détermination de similarité sémantique - l'algorithme de Leacock-Chodrow basé sur WordNet - établi une comparaison entre les descriptions des deux produits. Finalement, on a aussi développé, sous la forme de services Web, des fonctions de conversion d'unités.

Le système multi-agent a été développé en utilisant la plate-forme intégrée de développement d'agents « Java Agent DEvelopment Framework » (JADE). Pour parvenir à implementer notre système multi-agent il nous a fallu répondre a trois questions: (*i*) comment est-il possible de créer des agents aux ontologies différentes de façon systématique, automatique et intégrée? (*ii*) comment peut-on assurer une compréhension nette des intentions commerciales (de négociation) entre agents ayant des vocabulaires distincts? (*iii*) quels protocoles d'interaction devra-t-on utiliser afin de négocier et de résoudre les problèmes d'interopérabilité?

Pour répondre à la première question, nous avons développé une nouvelle méthodologie. Cette méthodologie est intégrée dans la plate forme JADE et permet la création et l'usage, dans un même système multi-agent, d'ontologies différentes. Pour y parvenir, nous avons, d'abord, intégré dans la plate-forme JADE l'éditeur d'ontologies Protégé. En suite, on génère et sauve les ontologies en fichiers « Web Ontology Language » (OWL). Ainsi, au moment de la création du système multi-agent, l'utilisateur peut choisir le fichier d'ontologie OWL qui correspond à l'ontologie désirée.

Pour résoudre la deuxième question on a défini une ontologie de plus - l'ontologie de commerce électronique - partagée par tous les agents. Par conséquent, chaque agent possède deux ontologies: l'ontologie de commerce électronique partagée et sa propre ontologie de la connaissance du domaine. Finalement, nous avons adopté la combinaison de deux protocoles pour assurer la réalisation des négociations. Le premier protocole, « Foundation for Physical Intelligent Agents (FIPA) Contract Net Interaction Protocol », représente le scénario général des agents échangeant des biens, proposé par FIPA. Le deuxième, met en marche le protocole nécessaire pour résoudre les problèmes d'interopérabilité qui se présen-

tent pendant les négociations. Comme résultat, tous les agents échangent leurs messages en utilisant le « Agent Communication Language » (FIPA-ACL).

Les méthodologies proposées, les pas successifs du développement du prototype, les essais menés et les résultats obtenus sont présentés et discutés au long de ce document.

Contents

List of Figures

List of Tables

List of Abbreviations

Abbreviation	Description	Definition
KR	Knowledge Representation	page 1
MAS	Multi-Agent Systems	page 1
B2B	Business-to-Business	page 2
EI	Electronic Institution	page 2
VO	Virtual Organisation	page 3
CEAg	Customer Enterprise Agent	page 4
SEAg	Supplier Enterprise Agent	page 4
FAg	Facilitator Agent	page 4
OSAg	Ontology-based Services Agent	page 5
E-ComDO	E-Commerce Domain Ontology	page 5
AADO	Automobile Assembling Domain Ontology	page 5
UCS	Units Conversion Service	page 5
MTS	Matching Terms Service	page 5
LCH	Leacock and Chodorow	page 6
BLM	Basic Learning Mechanism	page 6
JADE	Java Agent DEvelopment Framework	page 6
IDE	Integrated Development Environment	page 6
JENA	Semantic Web Framework for Java	page 6
OWL-S	OWL-based Web Service Ontology	page 6
FIPA-CNP	FIPA Contract Net Interaction Protocol	page 7
OIP	Ontology Interaction Protocol	page 7
FIPA	Foundation for Physical Intelligent Agents	page 7
B2C	Business-to-Consumer	page 7
AI	Artificial Intelligence	page 7
VE	Virtual Enterprise	page 11
KSL	Knowledge Systems Laboratory	page 28
KIF	Knowledge Interchange Format	page 28
CML	Chemical Markup Language	page 28
CORBA IDL	Common Object Request Broker Architecture - Interface Definition Language	page 28
Prolog	Programming in Logic	page 28

Abbreviation	Description	Definition
OCML	Operational Conceptual Modelling Language	page 28
RDF(S)	Resource Description Framework Schema	page 28
WebODE	Ontological Engineering Workbench	page 28
UPM	Universidad Politécnica de Madrid	page 28
ODEKM	Ontology-based Knowledge Management System	page 28
ODESeW	Automatic Semantic Web Portal Generator	page 28
ODEAnnotate	Web Resources Annotation Tool & ODEAnnotate	page 29
ODESWS	Semantic Web Services Editing Tools	page 29
API	Application Programming Interface	page 29
XML	Extensible Markup Language	page 29
OIL	Ontology Inference Layer	page 29
DAML+OIL	DARPA Agent Markup Language + Ontology Inference Layer	page 29
XCARIN	XMLization of the description logic language CARIN	page 29
FLogic	Frame Logic	page 29
OntoEdit	Ontology Engineering Environment	page 29
AIFB	Institute of Applied Informatics and Formal Description Methods	page 29
OXML	OntoEdit Extensible Markup Language	page 29
SMI	Stanford Medical Informatics	page 29
GUI	Graphical User Interface	page 29
Cycl	Cyc Representation Language	page 31
SHOE	Simple HTML Ontology Extensions	page 31
XOL	XML-based Ontology-exchange Language	page 31
MCC	Microelectronics and Computer Technology Corporation	page 31
SGML	Standard Generalized Markup Language	page 32
HTML	HyperText Markup Language	page 32
RDF	Resource Description Framework	page 32
W3C	World Wide Web Consortium	page 32
OWL	Web Ontology Language	page 33
SRI	Scientific Research Institute	page 33
DAML	DARPA Agent Markup Language	page 33
NAICS	North American Industry Classification System	page 34
SCTG	Standard Classification of Transported Goods	page 34
AOL	America Online	page 35
BT	British Telecom	page 35
UNSPSC	United Standard Products and Services Codes	page 35
RN	RosettaNet	page 36
KQML	Knowledge Query and Manipulation Language	page 39
FIPA-ACL	FIPA Agent Communication Language	page 39
FIPA-SL	FIPA Content Language Specification	page 39
UML	Unified Modeling Language	page 44
OA	Ontology Agent	page 46

Abbreviation	Description	Definition
ONP	Ontology Negotiation Process	page 47
LDOCE	Longman Dictionary of Contemporary English	page 59
HSO	Hirst and St-Onge	page 61
WUP	Wu and Palmer	page 62
lcs	lowest common subsumer	page 62
RES	Resnik	page 63
IC	Information Content	page 63
JCN	Jiang and Conrath	page 63
NLP	Natural Language Processing	page 65
ARTEMIS	Analysis of Requirements: Tool Environment for Multiple Information Systems	page 67
COMA	COmbination of Matching Algorithms	page 67
FCA-merge	Formal Concept Analysis	page 68
IF-Map	Ontology-Mapping Method based on Information-Flow Theory	page 68
NOM	Naive Ontology Mapping	page 68
OLA	OWL-Lite Aligner	page 69
QOM	Quick Ontology Mapping	page 69
SF	Similarity Flooding	page 69
S-Match	Schema-based Matching	page 69
SAT	SATisfiability	page 69
MAFRA	Ontology MApping FRAmework Toolkit	page 70
OBSERVER	Ontology Based System Enhanced with Relationships for Vocabulary heterogeneity Resolution	page 71
IRM	Inter-ontology Relationship Manager	page 71
MOMIS	Mediator envirOnment for Multiple Information Sources	page 71
CRM	Content Reference Model	page 81
USD	Dollar	page 79
EUR	Euro	page 79
DF	Directory Facilitator	page 80
CLS	Ontological Classes	page 83
FIPA-CNP	FIPA Contract Net Interaction Protocol	page 7
CFP	Call-For-Proposals	page 85
CPAN	Comprehensive Perl Archive Network	page 100
CGI	Common Gateway Interface	page 121
MIME	Multipurpose Internet Mail Extensions	page 122
URL	Universal Resource Locator	page 122
IP	Internet Protocol	page 122
HTTP	HyperText Transfer Protocol	page 122
URI	Uniform Resource Identifier	page 125
UDDI	Universal Description Discovery and Integration	page 125
WSDL	Web Services Description Language	page 125
WSDL2Java	WSDL to Java Remote Interface RMI Compiler	page 125
JVM	Java Virtual Machine	page 129

Abbreviation	Description	Definition
RMI	Java Remote Method Invocation	page 129
IIOP	Internet Inter-ORB Protocol	page 129
CORBA	Common Object Request Broker Architecture	page 129
GNU	General Public License	page 129
LGPL	Lesser General Public License	page 129
JRE	Java Runtime Environment	page 129
JDK	Java Development Kit	page 129
AMS	Agent Management System	page 130
ACC	Agent Communication Channel	page 130
RMA	Remote Monitoring Agent	page 130
BDI	Belief-Desire-Intention	page 131
CBR	Case-Based Reasoning	page 136

Acknowledgements

First I would like to thank my family because without their love and support I would not be able to go ahead. All my grateful to my wonderful parents Olímpio and Glória Maria, my lovely sisters Luciane and Thays, my brother-in-low Sandro, my sweet grandmother Azize, and to my little angel Isadora.

I would like also to thank my supervisor, Professor Eugénio Oliveira, for the opportunity to work and learn with him. Thank you for your technical advices, comments and suggestions. Special thanks for the support when I arrived in Portugal, being alone and far from our country is always difficult and it is important to know we have someone to count with.

I am grateful to Professor Marcos Shmeil for all the emails with motivate words during these years. Thank you specially for trusting me and for the help and incentive you gave me to come to Portugal.

I am also grateful to Professor H. Sofia Pinto for being always so nice answering my emails about ontologies, even without knowing me. Thank you for helping me with the first steps about ontologies, and for the good ideas for my work.

Many thanks to Benedita Malheiro for being so careful with me since I arrived in Porto. Special thanks for the time you spent reading carefully this thesis. Thank you for the detailed English revision and all the relevant comments and suggestions.

Thanks also to Professors Rui Camacho and Luís Paulo Reis for the comments and suggestions about some Chapters.

Thanks also to Daniel Palzer, Frédéric Hustinx and João Pinto for the technological support. Special thanks to Daniel for the months we worked together in a really fruitful cooperation.

Thanks to the Mechanical Engineering Department and colleague Roberto Hernandéz Garcia for his patience of teaching me about automobile assembling.

Thanks also to Ana Paula Rocha, António Castro, Célia Martins, Daniel Moura, Francisco Reinaldo, Henrique Lopes Cardoso, Luís Sarmento, Luís Paulo Reis, and Nuno Sousa for the moments I spent with them at LIACC.

Thanks to Rosário Rebelo, José António Nogueira and Nina Amaral for helping me with the administrative tasks whenever it was necessary.

I am very grateful to my adoptive Portuguese family Pinto Ribeiro (Dr. João, Dra. Maria Fernanda, Joana and Miguel) who received me as a member of their family and made me feel at home several times. I will be forever grateful.

"I could endure, although not without pain, if all my loved ones had died, but I would go crazy if all my friends died! Some of them I do not look for, it is enough to know that they exist. This condition encourages me to follow ahead with life... but it is delicious to know and feel that I love them, although I do not always declare or look for them ..." (Vinícius de Moraes). My special thanks to my lovely friends, without them this period had not been so good: Adilson de Ângelo, Ana Paula Gonçalves, Benedita Malheiro, Dulce Mota, Elaine Souto, Fabíola Bezerra, Fernanda Campos, Francisco Reinaldo, Joana Pinto Ribeiro, Karin Chvatal, José António Nogueira, Raquel Kolitski Stasiu, Ricardo Moraes, Rosário Rebelo, and Solange Nogueira.

I am grateful to Faculdade de Engenharia da Universidade do Porto for the resources provided for the development of this work.

I am also grateful for the financial support given by: Fundação Araucária for paying a part of my first faculty fee, the Pontifícia Universidade Católica do Paraná and Fundação para a Ciência e Tecnologia for giving me the chance to focus exclusively in this PhD, and to LIACC for their financial support to my participation in international conferences.

Chapter 1

Introduction

Organisations, namely enterprises, are increasingly rationalising their dimension[1]*, internal operations*[2] *and external processes*[3]*. While the advent of the Internet and the associated technologies provide a basic infrastructure to support knowledge representation (KR), knowledge management and the interaction between different nodes (internal and external to the organisation), agent and Multi-Agent Systems (MAS) technology give computational support to the modelling of the new inter-organisational relations, leading to a new type of economy - the electronic commerce economy (e-economy). However, new sets of techniques and methods need to be investigated and enhanced in order to allow, in a trustful and transparent way, the automatic, flexible and secure functioning of this type of economy. This broad research and development area, in which the work here presented is included, involves knowledge representation methods, automatic interaction processes and distributed platforms.*

1.1 Motivation

E-commerce is currently facing revolutionary changes: electronic marketplaces are enabling new kinds of services and interactions between suppliers and customers [Fensel, 2001]. It is an application domain where thousands of heterogeneous enterprises, acting both as suppliers and customers and using different formats to represent the traded products, may be involved. Moreover, different companies tend to use knowledge representations (ontologies) that differ signif-

[1] By downsizing and outsourcing.

[2] By having their characteristics, products and *modus operandi* electronically available.

[3] By automating interactions between privileged partners.

icantly either syntactically or semantically. This heterogeneity in the representation of the comercial products is a critical impediment to an efficient business information exchange and to the automation of the business-to-business (B2B) transactions.

Furthermore, in a decentralized and distributed approach, computational agents may negotiate with other agents geographically distributed. The agents may represent enterprises that negotiate using their own product descriptions as well as their local currency and measurement units. If each agent uses local units, either for measurements or currency, and they differ from the ones used by other agents then an interoperability problem occurs. These issues make the negotiation process, as well as the selection of partners to form an alliance of enterprises (Virtual Organisation), hard to achieve.

In a B2B context, multi-agent systems (MAS) that implement virtual marketplaces are intended to be open systems. The involved agents, representing enterprises, use ontologies to model their knowledge. Therefore, each agent defines the concepts and the characteristics they are aware of using different ontology tools and ontology languages. Even when a group of agents models the same knowledge domain, it is foreseeable that their ontologies will differ significantly, either syntactically or semantically. As a result, the use of different ontologies in shared application domains affects the interoperability and compromises the openness of the MAS. Additionally, basic differences, as the use of different measurement and currency units, generate also undesirable effects.

The need to regulate the interaction of heterogeneous agents in open environments has led to the concept of Electronic Institution (EI) [Dignum and Dignum, 2001]. The EI concept is particularly important when attempting to apply MAS technology to real-world scenarios, such as e-business, where trust is a major concern. One of the main roles of an EI is to support agent interaction within a coordination framework, making the establishment of business agreements more efficient. To achieve this goal, a set of institutional services are provided, among which are those devoted to the resolution of ontological discrepancies, which must be dealt with to make the establishment of business agreements possible [Malucelli *et al.*, 2005b].

The EI concept represents the virtual counterpart of real world institutions. One of its roles is to provide institutional services. Besides enforcing norms, institutional services should assist in the coordination efforts between agents which, representing real world entities, interact with the aim of establishing business relationships.

Our scenario is the domain of e-business automation, comprising not only information gathering and filtering agents but also the establishment and operation of business relationships. Furthermore, we are interested in the process of Virtual

Organisation (VO) formation and operation through which agents, representing different business units or enterprises, come together to explore new market opportunities by combining skills and resources that no single partner can alone fulfill. In our open and distributed environment, where agents representing different enterprises meet to search for new market opportunities, problems related to interoperability and trust are expected to occur.

Ontological discrepancies, *i.e.*, the existence of different representations and terminologies regarding the same concepts within the MAS, occur when agents with different ontologies interact. To make matters worst, no formal ontology mapping procedures are available. [Uschold, 2001] enumerates some barriers to effective agent communication:

1. There are many different ontology languages:
   - Ontology languages may be based on different underlying paradigms;
   - The level of expressiveness is variable;
   - Some ontology languages have formally defined semantics;
   - Some ontology languages provide inference support.
2. Even when the exact same ontology language is used, ontologies can be incompatible in various ways:
   - People or agents may use different terms for the same item;
   - People or agents may use the same term for different items;
   - A given notion or concept may be modeled using different primitives of the language.

A simple solution would be to define either a common ontology or a shared ontology understood by all enterprise agents participating in the business interactions. One would expect that, by defining a simple common ontology, the agents developed by different participants and located at different places would be able to share that ontology. This would provide a basis to a meaningful conversation, allowing advanced knowledge sharing between computers and humans. Moreover, it would ensure the preciseness and quality of the communication between agents. However, in open environments (where a central design is neither possible nor desirable) populated by heterogeneous agents it is unlikely to have a common ontology. Each agent will typically use a different ontology and the definition of a universal shared ontology is nothing but a *chimera*. The main reason is that no enterprise will consider converting the content of its ontology into a new representation unless the new ontology is a *de facto* standard. And currently, there are no standard e-commerce ontologies available.

1.2 Research Aim

The aim of this thesis is the development of ontology-based services to be integrated in an Electronic Institution capable of supporting the automatic negotiation required to form a VO. These services, provided through an Ontology-based Services Agent, are intended to match concepts from two agents using different ontologies by detecting the existing similarities.

In order to be able to compare and detect similarities between two concepts from distinct ontologies, the ontologies must share some components (a relation, a hierarchy, an attribute, a data type, *etc.*). A good starting point is to compare the characteristics (attributes) of the concepts (products) under evaluation since attributes capture details. In our approach, we use the characteristics (grouped by data type), the relation *has-part* and the descriptions of concepts as the common bridging components between ontologies. As a result, to successfully match two ontologies, there is a set of relations and characteristics that must be known and used in both ontologies.

Our focus is on lightweight ontologies whose specifications include concepts, characteristics, a natural language description explaining the meaning of the concept and a set of relations connecting concepts.

1.3 Main Contributions

Our main contributions are based on the research's aim and they are summarized below:

Development of Agents

The Electronic Institution MAS is made of agents which are deliberative, autonomous, intelligent entities that use different ontologies and apply goal driven reasoning to achieve their objectives. There are at least four types of agents:

1. The **Customer Enterprise Agent (CEAg)** that represents the enterprises interested in buying an item (product/service/good) required to assemble a final product. Several suppliers in the world may have these products/services/goods with different prices and conditions.

2. The **Supplier Enterprise Agent (SEAg)** which models the enterprises interested in providing some kind of product/service/good.

3. The **Facilitator Agent (FAg)** that matches the right agents (potential negotiators) and supports the negotiation process.

4. The **Ontology-based Services Agent (OSAg)** which provides the proposed ontology-based services to the enterprise agents (CEAg and SEAg).

Knowledge Representation

There are two main types of ontologies within our EI MAS:

1. E-Commerce Domain Ontology (E-ComDO)

   The e-commerce ontology defines an e-commerce vocabulary just for the trading. This vocabulary, which contains the terms used in the negotiation process, ensures that all agents will uniformly interpret the negotiation intent of the messages exchanged. This does not imply understanding the specific content of the message, *i.e.*, the requested products/items, since each agent interprets this information based on its own private automobile assembling domain ontology.

2. Automobile Assembling Domain Ontologies (AADO)

   The automobile assembling domain ontologies represent the diverse views that the multiple enterprise agents may have over the automobile assembling domain. As far as we know there are no standard ontologies for the automobile assembling domain. We have contributed by defining lightweight ontologies which, as a result of this work, are now available for the interested community.

Methodologies for the Resolution of Interoperability Problems between Agents using Different Ontologies

- Units Conversion Services (UCS)

  The UCS are invoked whenever the enterprise agents involved in a negotiation use different currency and measurement units. The implemented services calculate the conversion ratio between different currency and measurement units. The currency and measurement units conversion services are provided as Web services.

- Matching Terms Service (MTS)

  The MTS is contacted whenever an enterprise agent is unable to understand the content of a message, *e.g.*, an item (product/service) under negotiation. Our approach aims at creating a methodology that assesses lexical and semantic similarity among concepts represented by different ontologies without the need to build *a priori* a shared ontology. We have integrated the following three similarity matching algorithms:

  - Calculation of the n-grams value [Damashek, 1995] between the attributes and relations of the concepts involved.
  - Calculation of the n-grams value between the descriptions of the concepts involved.
  - Application of the Leacock-Chodorow (LCH) method [Budanitsky and Hirst, 2005] based on WordNet to detect the semantic similarity between a pair of concepts.

- Basic Learning Mechanism (BLM)

  Basic learning is applied with the purpose of improving the performance of the MTS and, consequently, the negotiation process. Once the OSAg has established the similarity between a pair of terms from different ontologies, this knowledge is stored in order to be available for future negotiation rounds. The performance improvement occurs with time: as the number of negotiations rounds increases, so does the amount of matched terms memorised.

Open MAS Composed of Heterogeneous and Competitive Agents

- Methodology for Creating and Maintaining Agents with Different Ontologies in JADE

  Since we decided to use the Java Agent DEvelopment Framework (JADE) Integrated Development Environment (IDE), we were faced with the problem of creating and maintaining JADE agents with disparate domain ontologies.

  The standard JADE's ontology support (version 3.3) is intended for a unique common ontology. As far as we know, there is no integration between JADE and any of the existing OWL-based ontologies tools. Perhaps, future versions of JADE will include this functionality since the JADE group has been discussing the incorporation of OWL-based Web Service Ontology (OWL-S) for more powerful semantic manipulation of Web services by JADE agents.

  To address this issue, the different domain ontologies representing the private knowledge of each agent are created with the Protégé ontology development tool [Noy *et al.*, 2000] and stored as Web Ontology Language (OWL) files. The resulting OWL files are them handled by the Semantic Web Framework for Java (JENA[4]) model interface. JENA extracts the ontological information from the OWL files and implements a transparent mapping mechanism from ontologies to agents.

[4]JENA - Semantic Web Framework for Java, http://www.hpl.hp.com/semweb/jena.htm, October, 2004

- Interaction Protocols

  The implementation of the negotiation process combines the FIPA Contract Net Interaction Protocol (FIPA-CNP) with an additional protocol called Ontology Interaction Protocol (OIP). The FIPA-CNP represents the general scenario of agents trading goods or services proposed by Foundation for Physical Intelligent Agents (FIPA[5]). Like other interaction protocols, FIPA-CNP structures complex tasks as aggregations of simpler ones. The OIP implements the message flow necessary for solving the interoperability problems, *i.e.*, the interaction between enterprise agents and the OSAg.

OntoServices Test-bed

The developed MAS test-bed contains four different types of agents: consumer and supplier enterprise agents (CEAg and SEAg), a facilitator agent (FAg) and a ontology-based services agent (OSAg). The latter includes the MTS, UCS and the BLM. The goal of enterprise agents is to trade products or services. Although the enterprise agents share the same application domain, each enterprise agent has its own private ontology. The MAS addresses a specific stage of the virtual enterprise formation process.

Since ontologies frequently need to be maintained and updated, we have integrated the Protégé editor with the JADE platform, facilitating the evolution of the implemented domain ontologies.

1.4 Outline of the Thesis

In Chapter 2 we give an introduction to the MAS in B2B electronic commerce area, briefly discussing the B2B and Business-to-Consumer (B2C) concepts. We review the B2B life cycle stages to understand the role of agents in each stage. To contextualise our work, we present the concept of the EI.

Chapter 3 presents an overview of the theoretical foundations of ontologies. We review the different meanings that the term ontology takes in Artificial Intelligence (AI), as well as the importance of developing ontologies. We explain the different types of ontologies that are found in the literature and we emphasise on e-commerce ontologies. The main ontology languages and ontology building-tools are discussed and compared.

In Chapter 4 we identify the main interoperability and heterogeneity problems found in open MAS and explain the importance of using ontologies. Ontologies play an important role as they can support the integration of heterogeneous and

[5]FIPA, *The Foundation for Intelligent Physical Agents*, http://www.fipa.org, April, 2005

distributed information sources. In addition, we also present the state of the art in the area, with special emphasis on the case of open systems where agents are using heterogeneous ontologies.

Chapter 5 presents the ontology mapping process. In the context of this work, the ontology mapping process is the most important mechanism implemented since it attempts to find the correspondence between terms from two distinct agent ontologies and, thus, solve the interoperability problems. In this Chapter we describe the most relevant similarity detection approaches used to perform ontology mapping as well as some ontology mapping systems and frameworks.

Chapter 6 discusses the services proposed to facilitate the negotiation process. We also describe the architecture of our system, describe the role of each agent and their interaction, with special emphasis on the OSAg. We explain the process of mapping multiple ontologies into JADE agents as well as the Ontology Interaction Protocol (OIP). Finally, the OSAg Basic Learning Mechanism (BLM) is explained.

In Chapter 7 we detail the implementation and present some experiments with different scenarios in the automobile assembling domain.

Finally, Chapter 8 discusses the conclusions and presents future possible directions that may emerge from this work.

Chapter 2

Multi-Agent Systems in B2B E-Commerce

This Chapter provides an overview of the role of MAS in the B2B e-commerce context. The B2B life cycle is presented and analysed in order to understand the interactions between the parties engaged and the tasks in which agents can assist the B2B activity. We also introduce the concept of Electronic Institution (EI) and explain the contribution of our work.

2.1 Introduction

Twenty years ago, distributed AI, and in particular MAS, emerged in the field of AI. Nowadays, MAS are not simply a research topic, but also an important subject of academic teaching and of industrial and commercial applications such as electronic commerce (e-commerce) and electronic marketplaces.

Electronic commerce refers to the online exchange of value, without geographical or time restrictions, between companies and their partners or customers [Singh *et al.*, 2001].

The electronic commerce world is traditionally divided into two parts [Subramani and Walden, 2000]: B2B and B2C. The difference between B2B and B2C is the nature of the relationship. B2B and B2C are concepts describing ways of doing business and not just descriptions of the participants in the process. In B2B e-commerce the idea is to establish a close relationship between two companies that will make some sort of complementary investments to enable one another's e-commerce strategy. Usually, it is necessary to adopt similar standards, implement extensive inter-firm communication and collaboration as well as perform a

joint investment in information technology. In B2C e-commerce there may be elements of branding, customer relationships and personalization. However, there is no mutual investment in standards, information technology or communication.

As in [Subramani and Walden, 2000], in this thesis we consider B2B as a joint action between companies; if an e-commerce initiative requires the participation of multiple companies, we consider it B2B e-commerce.

Due to the exponential growth of B2B e-commerce, the demand for agents is increasing since agents are computational entities which act on behalf of their owner to locate and retrieve information and make reasonable decisions based on the owner profile. Agents negotiate with multiple suppliers, monitor multiple auctions and use intelligent strategies to find the best deal for the users. Agents can also represent enterprises/organisations in a B2B context. In this latter scenario agents collaborate in order to achieve a common goal or select the best business partners through negotiation [He *et al.*, 2003].

An agent is a computer system that is situated in some environment and that is capable of autonomous action in this environment in order to meet its design objectives [Wooldridge, 2002]. Although in many cases agents can act separately to solve a particular problem, it is frequent for systems to be composed of several different agents developed to cooperate in a complex problem involving data, knowledge or distributed control [Oliveira *et al.*, 1999].

A MAS is a system composed of several agents with the capability of mutual interaction. A competitive or cooperative interaction occurs when two or more agents are brought into a dynamic relationship through a set of reciprocal actions. Therefore, interactions result in a series of actions whose consequences influence, in turn, the future behaviour of the agents. The agents interact through a series of events, during which they are in contact with each other in some way, whether this contact is direct or takes place through another agent or through the environment [Ferber, 1999]. Furthermore, MAS have the capability to be developed and implemented in modules, to represent multiple points of view (knowledge from different experts) and to be reusable.

2.2 B2B Life Cycle

[He *et al.*, 2003] presents the B2B life cycle model (Figure 2.1) and analyses the possible roles of agents. This model describes the interactions between two (or more) parties engaged in e-commerce and identifies the tasks in which agents can assist in a B2B e-commerce framework.

The B2B life cycle has the following stages: partnership formation, brokering, negotiation, contract formation, contract fulfilment and service evaluation. These stages are explained in the following sub-sections.

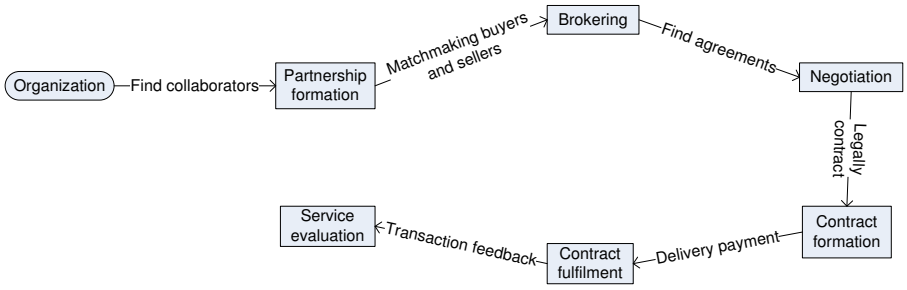


Figure 2.1: B2B Life Cycle Model

2.2.1 Partnership Formation

The formation of several new B2B e-businesses and dedicated Web sites provide evidence that companies (many are industries) are seeking to negotiate prices, broaden their supplier bases and streamline procurement processes using e-commerce [Barnes-Vieyra and Claycomb, 2001]. B2B e-commerce is changing the way companies purchase from and sell to each other. The companies that do not adapt to the emerging technologies and fail to adopt e-commerce strategies will loose market share and experiment profit decrease.

Through the use of the new Internet based technologies it is possible for an organisation to search for partners without having to take into consideration geographical and time restrictions. In the partnership formation stage it will be possible to form a new virtual enterprise.

Virtual Enterprise

"A Virtual Enterprise (VE) is a temporary alliance of enterprises that come together to share skills or core competencies and resources in order to better respond to business opportunities and whose cooperation is supported by computer networks" [Camarinha-Matos and Afsarmanesh, 1999].

"In a VE, a temporary consortium of partners and services is formed for specific purposes. These purposes could be a temporary special request, an ongoing goal to fulfill orders or an attempt to take advantage of a new resource or market niche. The general rationale for forming the VE is to reduce costs and time to market while increasing flexibility and access to new markets and resources" [Petrie and Bussler, 2003].

According to [He *et al.*, 2003], a VE is composed of companies cooperating and sharing their resources and skills to support a particular product effort (for as long as it is viable to do so).

The VO concept is similar to the VE concept since the VO is also an alliance of enterprises. However, while the VE is an alliance that is profit driven, the VO can be a non-profit driven alliance. The VE is a particular type of VO.

"Humans have limited ability to keep track of what is going on in the range of VO activities, given the tight time constraints and limited resources required and used by VO. This is exacerbated by frequent interruption of their work, with recent research indicating that white-collar employees receive a communication every five minutes. As a result, AI provides the VO MAS approach to mitigate the limitations and constraints of human agents in order to monitor and control substantial resources without the time constraints inherent in human organisations" [O'Leary *et al.*, 1997].

Due to these facts as well as to the fact that a VE is composed of a number of autonomous entities that need to interact with one another in flexible ways, agent technology is a natural underpinning model [Barnes-Vieyra and Claycomb, 2001]. Furthermore, appropriate platforms for multi-agent development have been released and experimented. Applications of such tools to the electronic business domain brought up the need for the creation, representation and exploration of domain ontologies.

The present work is focused on the VE formation process and not directly related to the concept of VO. In our B2B life cycle, the enterprises are profit driven entities interested in finding partners for buying/selling products. However, this does not mean that our approach cannot be applied to VO.

In detail, the formation of a VE has a life cycle with the following stages [Fischer *et al.*, 1996]:

- Identification of Needs: appropriate description of the product or service to be delivered by the VE.
- Formation (partners selection): automatic selection of the partners which, based on their specific knowledge, skills, resources, costs and availability, will integrate the VE.
- Operation: control and monitoring of the partners' activities including the resolution of potential conflicts and possible VE reconfiguration due to partial failures. It is important to monitor the actions in order to be sure that they deliver the expected services in an effective manner.
- Dissolution: breaking up the VE, distribution of the obtained profits and storage of relevant information for future use.

Sometimes, it is advisable to include a reconfiguration stage following "Operation", defining a sub-cycle inside the main cycle.

Several agents are needed to manage the ongoing operations. Moreover, several problems are involved in the VE formation process: (*i*) the need to describe the required product or service based on some ontology ("Identification of Needs" phase); (*ii*) the need to specify the knowledge, skills, resources, costs and availability in a way that it is intelligible to all participants ("Partners Selection" phase); (*iii*) the need for a consistent and comprehensible communication ("Operation" and "Dissolution" phases); and (*iv*) the lack of understanding that may arise during the agents' interaction due to structural as well as semantic heterogeneity.

An enterprise that wishes to successfully negotiate with other enterprises distributed all over the world must not only recognise and adapt to dramatic differences, *e.g.*, distinct domain knowledge, but also to minor specificities of its partners, *e.g.*, the use of different measurement and currency units.

2.2.2 Brokering

Brokering is the process that matches providers of goods together with buyers. This is one of the major challenges in an open B2B MAS: how to find agents (we often refer to agents rather than to the enterprises they represent since agents are the enterprises "virtual" counterparts) that might have the capabilities or products needed. In this context, suppliers advertise themselves to an appropriate broker agent. Requesters send all service requests directly to the broker, who farms them out to the providers - seeking to equalize the load among them [Decker *et al.*, 1997]. Suppliers are looking for potential buyers for their products while buyers are searching for the most appropriate goods/products/services provider.

This brokering phase should result in a customer having a list of potential trade partners [Trastour *et al.*, 2002]. Considering the increasing complexity of the trading environment, it is complicated to navigate and find the necessary information. Due to the complexity of the task, companies in B2B e-commerce are using brokers (also called matchmakers [Trastour *et al.*, 2002]) represented by an agent with several services or a MAS with specialised and collaborative agents. Some of the functions offered by a broker are [Foss, 1999]:

- Searching and information retrieval.
- Helping the user with online services.
- Profiling users.
- Filtering incoming information and protecting the user from intrusive access from other users or agents.
- Predicting user requirements.

- Negotiating payment schedules for the service.
- Commercial negotiation between customers and providers to achieve a contract agreement.
- Continually reassessing, on behalf of its clients, the market and maintaining awareness of their future potential requirements for information products and services.

The brokering process is initiated when some agent places an advertisement and is followed by inquiries regarding the advertisement made by other agents.

2.2.3 Negotiation

After the brokering stage, customers are already able to exchange information with potential suppliers. In this stage customer and suppliers try to find an agreement on the characteristics and conditions of the required products.

This automatic negotiation can significantly reduce negotiation time (by making large volumes of transactions possible in small amounts of time) as well as diminish the reluctancy some humans have to engage in negotiation (*e.g.*, because of embarrassment, personality, *etc.*) [Lomuscio *et al.*, 2000].

The strategies used during the negotiation stage depend on the specific characteristics of the scenario under consideration. Negotiations can be one-to-one, one-to-many or many-to-many and, as a consequence, many different protocols have been designed to carry them out [Trastour *et al.*, 2002]. Negotiation protocols define how agents interact with each other during the negotiation process.

One-to-one protocols are used when a seller offers a good at a fixed price and starts an iterated bargaining process. Another possibility is when the buyer sends the product requirements and the sellers who can meet those requirements make bids. One-to-many protocols apply when there is a seller and several buyers. This type of protocol includes the English auction, Dutch auction and the Contract Net interaction protocols. Many-to-many protocols are special auctions in which there are multiple kinds of goods to sell and bidders can bid on combinations of items. This type of protocols include the Continuous Double Auction and the Call Auction protocols.

2.2.4 Contract Formation

This stage represents the end of the negotiation and it is the moment when the terms, rights and duties are defined to create a binding contract between partners. As it is B2B e-commerce, it is necessary to automatically create electronic contracts.

Contracts formalise the ties binding groups of agents that jointly agree on a specific business activity. Contracts are used as a means of securing transactions between the involved parties. Electronic contracts are virtual representations of agreements and aim to improve the efficiency of the contracting act [Cardoso and Oliveira, 2005].

The business contract is important to enforce the terms of the contract, providing more guarantees to the involved parties and, thus, increasing the level of trust among business partners. [Goodchild, *et al.*, 2000] present a specification and implementation of business contracts, providing mechanisms to facilitate monitoring and enforcing terms and conditions.

2.2.5 Contract Fulfilment

By contract fulfilment it is meant that the involved parties carry out the agreed transaction according to the terms specified in the contract. Therefore, it is necessary to monitor the contracts, solve disagreements, disputes, payments, *etc.* Currently, there are few MAS implementing this stage because it implies complex legal issues which are subjected to court judgements.

2.2.6 Service Evaluation

Service evaluation does not play a major role in B2B MAS since it is a post-transaction stage where the end-users satisfaction is measured.

2.3 Electronic Institution

The need to regulate the interaction of heterogeneous agents in open environments led to the concept of EI [Dignum and Dignum, 2001]. This approach is particularly important when attempting to apply MAS technology to real-world scenarios such as e-commerce, where trust is a major concern.

One of the main roles of an EI is to support agent interaction within a coordination framework, making the establishment of business agreements more efficient. To achieve this goal, a set of institutional services is provided, among which are those devoted to the resolution of ontological conflicts as well as measurement or currency units discrepancies. These occurrences must be dealt with in order to establish meaningful business agreements.

In our domain, the EI is used to facilitate the interoperability between agents in the VE formation process. Below, we enumerate the generic steps for achieving both partner selection as well as operation monitoring in the VE formation process through an EI:

1. The enterprise agents register in the EI.
2. The enterprise agents in need of a service inform the EI about the basic requirements of that particular service.
3. The EI sends an announcement to the registered enterprise agents that might provide the required service.
4. The enterprise agents willing to provide the required service send an advertisement to the EI.
5. The enterprise agents that offer the best bids are selected through a flexible and adaptive negotiation process.
6. A contract is established between the selected partners to form a new VE according to sets of rules and norms applicable.
7. The EI monitors the VE activity, storing relevant information for future use.

Institutional services that increase the interoperability level among agents will lead to more efficient negotiation processes and, ultimately, to meaningful businesses contracts. In our application this role is played by the OSAg. In Figure 2.2 [Cardoso *et al.*, 2005] we present a global overview of the services provided by an EI, including our Ontology-based Services Agent (OSAg). The EI services that rely on the functionalities provided by the OSAg are shown in detail. In this figure registration and brokering are omitted.

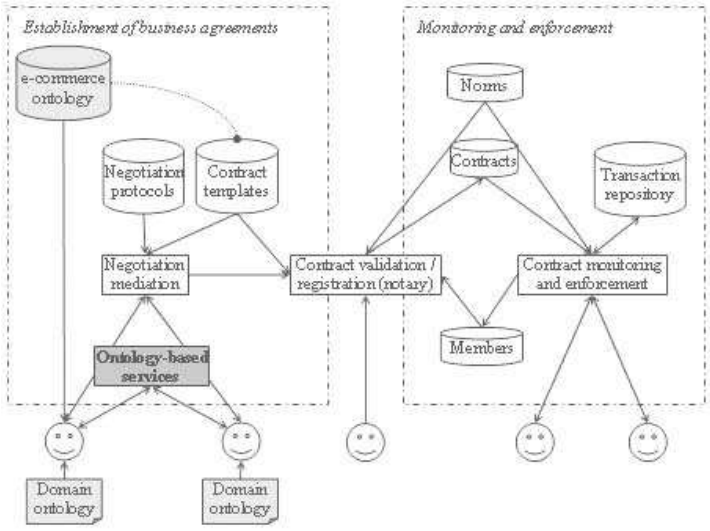


Figure 2.2: Services of an EI

The establishment of contracts based on appropriate negotiation protocols and contract templates are supported by the ontology-based services. These are crucial in open environment scenarios since different domain-dependent vocabulary may be used by different business entities. However, a common institutional ontology regarding general contract-related terms must always exist. The efficiency of a MAS EI relies on the use of a communication standard, including communication protocols, communication languages and ontologies.

The other EI services include negotiation mediation, contract validation/registration and contract monitoring and enforcement. While the negotiation mediation service and the normative framework that allows for contract monitoring and enforcement are described in [Rocha and Oliveira, 2001] and [Cardoso and Oliveira, 2005], this work is focused on the ontology-based services.

2.4 Conclusions

The popularity of B2B e-commerce is increasing among companies interested in automating negotiation and partner brokering while reducing time responses and costs. The success of the agent paradigm is undisputed in the open distributed e-commerce scenario. However, there are still several important challenges to be addressed prior to the industrialization of the multi-agent approach in B2B:

- Trust is still one of the issues to overcome since people still do not want to delegate on machines decision making.
- Brokering services need to improve the current partner-finding results.
- New negotiation protocols designed for the e-commerce environment, where contracts, norms and rules are contemplated in an integrated framework, need to be developed.
- Interoperability among agents in open MAS must be increased.

Although these are complex issues, it is necessary to contribute with new approaches. In our case, we address the interoperability problem and propose a set of services to support the VE formation stage.

Chapter 3

Ontologies

In this Chapter we present a theoretical overview of the ontologies domain. First, we start by providing some definitions and identifying the components and types of existing ontologies. Then, we address the design and development phases of an ontology, referring the main issues, tools and ontology languages. Some initiatives to classify products in the e-commerce domain are also described.

3.1 Introduction

The principles and methods for ontology representation were developed in the AI field in order to facilitate knowledge sharing and reuse. Since the beginning of the nineties ontologies have become a popular research topic investigated by several research communities, namely, knowledge engineering, natural language processing and knowledge representation groups. Nowadays, other areas such as intelligent information integration, Internet information retrieval, knowledge management and the semantic Web are also interested in this topic. The reason for this popularity is mainly due to the promise of providing a shared and common understanding of a specific domain that can be communicated between people and application systems [Fensel, 2001], [Duineveld *et al.*, 1999]. Ontologies have been developed to provide a machine-processable semantics of information sources that can be communicated between different agents (software or human entities) [Fensel, 2004]. They are also essential to the development and use of intelligent systems, particularly for the interoperation of heterogeneous systems. Ontologies are, thus, responsible for informing about the domain vocabulary and explaining the meaning that interacting systems attribute to terms.

Furthermore, an ontology facilitates the domain model construction since it is through the ontology that the vocabulary of terms and relations, with which it is possible to model the domain, is provided.

According to [Gruber, 1993] ontologies are developed to:

- Enable a machine to use knowledge in some application.
- Enable multiple machines to share knowledge.
- Help humans to understand more about some knowledge area.
- Help people build a consensus concerning some knowledge area.

Since ontologies intend to provide consensual domain knowledge, their development is frequently a cooperative process involving people from different origins. People and organisations that agree on some ontology are said to have an ontology commitment. Since ontology building is a difficult and time consuming task, it is usual to build new ontologies from existing ones, *i.e.*, using a part of an already existent ontology or building upon existent ontologies or ontology repositories. Different ontology tools and ontology languages are available to create ontologies.

The features presented so far do not provide a definition for ontology. The term ontology was transported from Philosophy to AI, where it is possible to find several definitions (some are even contradictory) depending on the specific research area.

3.2 Definitions of Ontology

There are several ontology definitions that have evolved in the last decades. However, there is a consensus among the ontology community of researchers about their role: they provide a common understanding regarding some domain knowledge. Diverse research communities, such as knowledge engineering, database and software engineering, use ontologies for many different purposes: natural language processing, electronic commerce, knowledge management, semantic Web, *etc.*

Some of the most representative ontology definitions are:

- [Uschold and Jasper, 1999] state that, although ontologies may have several forms, they always include a specific vocabulary and some specifications as far as their meaning is concerned. They include definitions and relations of how the concepts are interrelated, which imposes a collective domain structure and restricts the possible interpretations of the term.

- [Neches *et al.*, 1991] define ontology not only as a collection of terms and relations about the vocabulary of an area's topic, but also as the rules to combine the terms and relations to define extensions to the vocabulary.
- [Gruber, 1993] defines ontology as an explicit specification of a conceptualisation. "Conceptualisation" is basically the idea of the world that a person or a group of people may have; "explicit" means that the type of concepts and the restrictions about their use are explicitly defined.
- [Borst, 1997] presents a modified version of Gruber's definition by which ontology is a formal specification of a shared conceptualisation. In this context, "shared" means that the ontology should reflect the consensual knowledge accepted by a group; "formal" refers to the fact that a computer should understand the ontology.
- [Grüninger and Fox, 1995] declare that ontology is a formal description of entities, properties, relations, constraints and behaviours.
- [Huhns and Singh, 1997] argue that an ontology is the knowledge representation of some domain, which is available for all the other components in an information system.
- [Weiss, 1999] defines ontology as a object specification of concepts and relations in the interest area. For Weiss, ontology is more than a simple taxonomy of classes, since it describes the relations.
- [Noy and McGuinness, 2001] say that ontology is a formal explicit description of concepts in a discourse domain, where properties of each concept describe several characteristics, attributes of concepts and attributes' constraints.

Several other definitions are based on the process used to build the ontology:

- [Swartout *et al.*, 1997] states that if one uses large ontologies, *e.g.*, SENSUS with more than 70 000 concepts, to build new ontologies and knowledge bases in a specific domain, the ontology should be defined as a set of terms hierarchically structured. The resulting set of terms can be used as a skeleton for a knowledge base. According to this definition, the same ontology may be used to build several knowledge bases, which could share the same skeleton. Extensions of this structure should be possible to cover new areas: in a low level, by adding sub-concepts; in a medium level, by adding intermediate concepts; and, in a high level, by adding new high-level concepts. If concepts are built from the same ontology, they share a common structure and inference mechanism, making the task of merging and sharing knowledge bases easier.

- [Studer *et al.*, 1998] consider taxonomies as complete ontologies.
- [Noy and McGuinness, 2001] assume that the United Standard Products and Services Codes (UNSPSC) , e-cl@ss, RosettaNet (e-commerce ontologies) and Yahoo!Directory (taxonomy to search in the Web) are ontologies because they provide consensual conceptualisation in a specific domain.

Additionally, the ontology community distinguishes between ontologies that are mainly a taxonomy from ontologies that model a domain in a deeper way, providing constraints about the semantics of the domain:

- **Lightweight ontology** includes concepts, taxonomy of concepts, relations between concepts and properties that describe the concepts.
- **Heavyweight ontology** adds to the previous definition axioms and constraints.

Lightweight and **heavyweight** ontologies can be modelled with different modelling techniques and can be implemented in different languages and tools [Uschold and Gruninger, 1996]. Ontologies can be classified in different categories depending on how they are expressed.

For the purposes of this thesis, an ontology is a formal explicit description of concepts in a discourse domain.

3.3 Components of Ontology

There are different techniques that can be used to model and represent ontologies such as frames, first-order logic [Gruber, 1993a], description logics [Baader *et al.*, 2003], software engineering techniques [Cranefield and Purvis, 1999] or database technologies [Thalheim, 2000].

Although each of these techniques can represent the same knowledge with different degrees of formality and granularity, they have the same basic components:

- **Classes** to model the concepts of the domain or task. They are usually organised in taxonomies and inheritance can be applied. The class taxonomy is represented in a tree structure. Since multiple inheritances are permitted, one class may have several super-classes. Classes can be concrete or abstract. In contrast to abstract classes, concrete classes may have direct instances.

- **Attributes** to represent the characteristics of the concepts. Attributes are also called slots and sometimes roles or properties. They are usually distinguished from relations because their range is a data type (*string*, *number*, *boolean*, *etc.*).

- **Relations** to model types of associations between concepts. Binary relations are sometimes used to express concept attributes. However, the range of relations is different from the range of the attributes: the range of a relation is a concept.

- **Instances** to represent specific elements. They are specific entities of a given class. New instances can be created and values can be assigned to the attributes and relations. A form to enter data is generated automatically when an instance is created.

- **Functions** represent special cases of relations in which the n-th element of the relation is unique for the n-1 preceding elements.

- **Axioms** to model sentences that are always true. Axioms are used to verify the consistency of the ontology or the consistency of the knowledge stored.

Concepts, relations, attributes and instances are used to model lightweight ontologies. Heavyweight ontologies include also axioms and functions.

In order to model an ontology, we suggest the following step-by-step approach:

- Determine the domain and scope of the ontology by establishing the domain and use of the ontology, the types of questions the ontology should provide answers to and who will use and maintain the ontology.

- Consider reusing existing ontologies.

- Enumerate important terms in the ontology.

- Define the classes and the class hierarchy.

- Define the properties of classes - slots.

- Define the facets of the slots: value type, allowed values and cardinality.

- Create instances of a class.

Figures 3.1 and 3.2 provide screenshots of Protégé's ontology development tool [Noy *et al.*, 2000], showing the class editor and the slot editor to illustrate some of the components of the ontology.

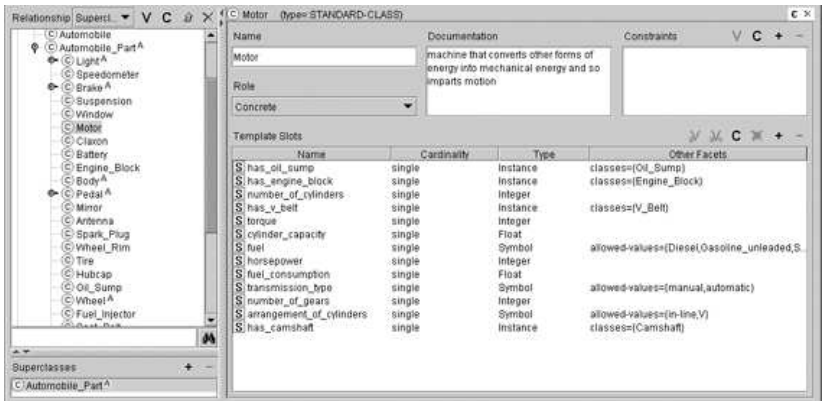


Figure 3.1: Edition of class "Motor" in Protégé

Figure 3.1 shows the class "Motor", a part of the "Automobile Assembling Ontology", in the Protégé editor. On the main frame, all defined template slots are listed and the taxonomy of entities is represented in a tree structure on the left side. Motor has only one superclass ("Automobile_Part").

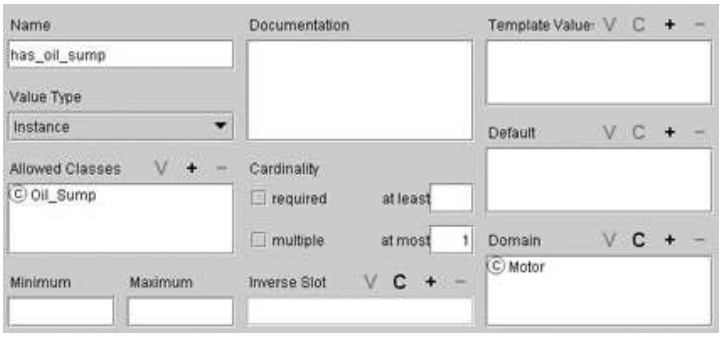


Figure 3.2: Edition of slot "has_oil_sump" in Protégé

Figure 3.2 shows the slot "has_oil_sump", which is defined in the domain "Motor" and is related to an instance of class "Oil_Sump".

3.4 Types of Ontologies

In the literature, ontologies are classified according to different features. We herein present the most used types.

For [Uschold and Grüninger, 1996], ontologies differ in the degree of formality by which the terms and their meaning are expressed. The knowledge expressed in the ontology may be the same, but it may differ when taking into account the way it is expressed. These authors classify ontologies as:

- **Highly informal** when expressed in natural language. This kind of ontology will be probably ambiguous due to the intrinsic ambiguity of natural language.
- **Semi-informal** when expressed in a restricted and structured form of natural language. In this kind of ontology there are improvements in clarity and reduction of ambiguity.
- **Semi-formal** when expressed in artificial languages, which are formally defined.
- **Rigorously formal** when precisely defined with formal semantics, theorems and proofs of properties such as soundness and completeness.

[van Heijst *et al.*, 1997] classify ontologies according to two dimensions: the amount and type of the conceptualisation structure and the subject of the conceptualisation. According to the type of structure of the conceptualisation, they distinguish three categories, mainly concerned with the level of granularity:

- **Terminological ontologies** which are made of lexicons that specify the terminology which is used to represent knowledge in the domain of discourse.
- **Information ontologies** that define the structure of a database such as a database schemata.
- **Knowledge modeling ontologies** that specify conceptualisations of the knowledge. They are often specified according to a particular use of the knowledge they describe.

As far as the subject of the conceptualisation is concerned, these authors distinguish four categories, according the type of knowledge that is modelled:

- **Application ontologies** that define the concepts necessary to model the knowledge of a particular application. Usually, they specialise terms taken from more general ontologies such as domain and generic ontologies. These ontologies are hardly reusable.
- **Domain ontologies** which define concepts and their relations in a specific domain. These ontologies are reusable in a given specific domain (automobile, engineering, medical, *etc.*)
- **Generic ontologies** that define general domain independent concepts. The common sense knowledge represented here is reusable across domains. They define concepts such as events, time, space, causality, behaviour, function, *etc.*

- **Representation ontologies** which capture the representation primitives used to formalize knowledge under a given knowledge representation paradigm. They provide formal definitions of the representation primitives used mainly in frame-based languages and allow building other ontologies by means of frame-based conventions.

[Guarino, 1998] classifies ontologies according to the level of generality. He presents the following classification:

- **Top-level ontologies** that describe very general concepts or common sense knowledge that are independent of a particular problem or domain.
- **Domain ontologies** that provide vocabularies about a generic domain.
- **Task ontologies** which define concepts related to the execution of a particular task or activity.
- **Application ontologies** that describe concepts depending on a particular domain and on a particular task. These ontologies often extend and specialise the domain and task ontologies.

In [Lassila and McGuinness, 2001] ontologies are classified based on the richness of their internal structure. The main categories are:

- **Controlled vocabularies** which are the simplest notion of ontology, *i.e.*, are made of a finite list of terms. A typical example is a catalogue.
- **Glossaries** that are lists of terms and meanings, which are usually expressed in natural language statements.
- ***Thesauri*** which provide additional semantic between terms, *i.e.*, they provide information as a synonym relation. *Thesauri* do not provide an explicit hierarchical structure.
- **Informal *Is-a* hierarchies** that contain a general notion of generalisation and provide specialisation although not as a strict subclass hierarchy, *e.g.*, the Yahoo! ontology in the Web.
- **Formal *Is-a* hierarchies** which organise the concepts according to a strict subclass hierarchy.
- **Frames** that include classes and their properties which can be inherited by lower levels classes of the formal *is-a* taxonomy.
- **Value restriction** that allow the application of restrictions on the values associated with properties.

- **General logical constraints** that are usually written in a very expressive ontology language such as Ontolingua [Farquhar *et al.*, 1997], which permits the specification of first-order logic constraints on concepts and their properties.

In our system, ontologies are classified according to the level of formality as terminological ontologies and, to the level of granularity, as domain ontologies once they include concepts which are organised in a hierarchy and concept definitions that are expressed in natural language. As far as our work is concerned, the domain is the automobile assembling domain.

3.5 Design Principles of Ontologies

[Gruber, 1993] proposes a set of preliminar principles for the design of ontologies. He identifies five principles:

- **Clarity**: the ontology should effectively communicate the intended meaning of the defined terms; definitions must be objective. While the motivation to define a concept appears from the social situations or computational requirements, the definitions should be independent of the social or computational context.

- **Coherence**: the ontology should be coherent, *i.e.*, the ontology should sanction inferences that are consistent with the definitions. Coherence should be applied also to concepts informally defined, *e.g*, the concepts described in natural language documentation. If a sentence that can be inferred from the axioms contradicts a definition or example given informally, then the ontology is incoherent.

- **Extensibility**: the ontology should be designed to anticipate the use of shared vocabularies. It should be possible to define new terms for special uses based on existent vocabularies, so that it does not require the revision of any pre-existing definitions.

- **Minimal encoding bias**: the conceptualisation should be specified at a knowledge level independent of a particular symbol-level encoding. Encoding bias should be minimized because agents sharing knowledge can be implemented in different systems and use different representation styles.

- **Minimal ontological commitments**: the ontology should require a minimum ontological commitment in order to support shared activities.

3.6 Ontology Development Tools

Ontology editors help knowledge engineers to build ontologies. Typically, an ontology editor supports the definition of concepts, attributes, axioms, constraints and hierarchies of concepts. Editors provide graphical interfaces and comply with existing standards for Web-based software development. Ontology editors enable ontology development, inspection, browsing, codifying and maintenance.

There are several ontology development tools such as Ontolingua, WebONTO, WebODE, Protégé, OntoEdit, OilEd, Apollo, SymOntoX, OntoSaurus, DagEdit, DOE, IsaViz, SemTalk, Icom, OntoBuilder, *etc.* These tools usually use different representation languages which are briefly described and referred in Section 3.7. In this Section we present the most popular ontology languages:

- **Ontolingua** [Farquhar *et al.*, 1997] was created by the Knowledge Systems Laboratory (KSL) at Stanford University. The system consists on a server and a representation language. Ontolingua provides an ontology repository allowing the creation of new ontologies and the modification of existent ones. This server was designed to allow several users to cooperate in the development of the ontology. The main modules are available for public use through a standard Web browser. The server uses the notion of users and groups to grant privileges: the proprietary of an ontology can give reading and writing access to specific users or groups of users. The server notifies the other users when some change occurs in the ontology. The ontology language used is the Knowledge Interchange Format (KIF). Ontolingua exports ontologies to several languages: KIF, LOOM, Chemical Markup Language (CML), Epikit, Common Object Request Broker Architecture - Interface Definition Language (CORBA IDL) and Programming in Logic (Prolog). Ontolingua, which was the first ontology environment developed, has no inference engine.

- **WebOnto** [Domingue, 1998] was created at the Open University by the Knowledge Media Institute. It was designed to support the creation and edition of ontologies through a Web browser. The language used to model the ontologies is the Operational Conceptual Modelling Language (OCML). OCML can be translated to Ontolingua as well as translated to/from Resource Description Framework Schema (RDF(S)).

- **WebODE** [Arpírez *et al.*, 2001; Corcho *et al.*, 2002] is an ontological engineering workbench (WebODE) created by the Ontology Group at Universidad Politécnica de Madrid (UPM). WebODE contains an editor, an ontology-based knowledge management system (ODEKM), an automatic Semantic Web portal generator (ODESeW), a Web resources annotation

tool (ODEAnnotate) and Semantic Web services editing tools (ODESWS). The ontologies can be accessed using a Java Application Programming Interface (API) *via* a local service or through an application running on the same computer where the ontology server is installed. WebODE ontologies can be in Extensible Markup Language (XML), RDF(S), Ontology Inference Layer (OIL), DARPA Agent Markup Language + Ontology Inference Layer (DAML+OIL), OWL, XMLization of the description logic language CARIN (XCARIN), Frame Logic (FLogic) and Prolog.

- **OntoEdit** [Sure *et al.*, 2002] or Ontology Engineering Environment (OntoEdit) is an ontology engineering environment created by the Institute of Applied Informatics and Formal Description Methods (AIFB) at the Karlsruhe University. Currently, it is commercialised by Ontoprise GmbH and there are free and professional versions available. The professional version includes a set of plug-ins, graphical rule editors, inference engine, export and import modules, *etc.* It is a Java stand-alone application that can be installed and run locally, but it is not open source. The ontologies can be exported to several ontology languages such as OntoEdit Extensible Markup Language (OXML), FLogic, RDF(S) and DAML+OIL.

- **OilEd** [Bechhofer, *et al.*, 2001] is a simple ontology editor that supports the construction of OIL-based ontologies. The tool does not provide collaborative ontology development such as versioning, integration or merging of ontologies. The design of OilEd is concentrated on demonstrating how the frame paradigm can be extended to deal with a more expressive modelling language and how reasoning can be used to support the design and maintenance of ontologies.

- **Protégé-2000** [Noy *et al.*, 2000] was created by the Stanford Medical Informatics (SMI) group at Stanford University and it is freely available. Protégé is a Java-based stand-alone application that is intended to be installed and run on a local computer. The Protégé tool is an open-source Java-based knowledge-modelling platform. The core of the application is the ontology editor with a Graphical User Interface (GUI) which allows the user to construct some domain ontology, to customise automatically generated data entry forms and to enter data. External stand-alone Java applications can be developed using Protégé's API in order to build and access domain models. One of Protégé's major advantages is its extensibility: the architecture allows the development and integration of plug-ins. Plug-ins are additional modules that extend the Protégé system's core. They provide functionalities not provided by the standard Protégé distribution. Most of the existing plug-ins are developed either by Stanford University or by external partners and are available at Protégé Contributions Library. The

ontologies can be exported and imported from and into Protégé using with the help of back-ends plug-ins, namely, RDF(S), XML, XML Schema and OWL plug-ins. All ontologies in this thesis have been developed with the aid of Protégé (version 2.1.2) and of the following additional modules:

- **OWL plug-in** that is a back-end plug-in (also called storage plug-in) that allows the user to load and store ontologies in OWL format. Additionally, the OWL plug-in also allows: (*i*) editing and visualising OWL classes and their properties; (*ii*) defining logical class characteristics as OWL expressions; (*iii*) reasoning; and (*iv*) editing OWL individuals for Semantic Web markup.

- **BeanGenerator plug-in**[1] that is a so-called tab plug-in since it is embedded in the editor's GUI. It maps objects in the Protégé model to the corresponding Java classes. These automatically generated Java classes comply with the JADE specifications [Caire and Cabanillas, 2004]. Intelligent software agents can profit from this mechanism since the resulting Java source files can be accessed easily from any Java program.

- **OntoViz** which is a tab plug-in that provides a convenient graphical visualisation of ontology models.

Although there are several similar ontology development tools, none is complete. A sensible selection depends on the user's needs. A good approach is to identify the characteristics (description, architecture, interoperability, representation, inference services and usability) of each tool and choose the one that is most appropriate for the project at hand.

Usually, different tools are not able to interoperate and, consequently, problems occur when trying to integrate ontologies coming from distinct ontology IDE. It is also complicated to merge ontologies built with different tools or coded in diverse languages. There are neither studies regarding ontology portability between different tools nor about the eventual loss of knowledge experienced in the translation process.

[1] BeanGenerator, http://hcs.science.uva.nl/usr/aart/beangenerator/index25.html, October 2004

3.7 Ontology Languages

Knowledge representation is a multidisciplinary subject that applies theories and techniques from three areas [Sowa, 2000]:

- **Logic** that provides formal structure and inference rules.
- **Ontology** which defines the types of things that exist in the application domain.
- **Computer Science** that supports the application that distinguishes knowledge representation from philosophical knowledge.

Knowledge representation may be seen as the application of logic and ontologies to the task of building computational models for some domain. As a result, ontologies are of high relevance to any Knowledge Based System.

As the choice of a development ontology tool depends on the developer's individual preferences, the same occurs with the ontology language. There are several ontology implementation languages that can be classified as:

- Classical languages, such as Cyc representation language (Cycl), Ontolingua, LOOM, OCML, FLogic and KIF.
- Web-standard languages, such as XML and RDF.
- Web-based language, such as Simple HTML Ontology Extensions (SHOE), XML-based ontology-exchange language (XOL), OIL, DAML+OIL and OWL.

Some of these languages are briefly explained below:

- **CycL** (Cyc Language) [Lenat and Guha, 1990] is a formal language based on frames and first-order logic. The vocabulary of CycL consists of terms. The terms can be divided in constants, non atomic terms, variables and some other objects. The terms are combined in CycL significant expressions, which can be used to do assertions in the knowledge base Cyc[2]. Cyc is a huge knowledge base with common sense knowledge created by the Microelectronics and Computer Technology Corporation (MCC).
- **Ontolingua** [Farquhar *et al.*, 1997] was developed at the Knowledge Systems Laboratory of Stanford University. Ontolingua, which is an ontology language based on KIF and on the Frame Ontology, is the ontology-building language used by the Ontolingua Server (described in Section 3.6.). The Ontolingua ontologies are kept at the Ontolingua Server.

[2]CyCorp, http://www.cyc.com/, October, 2005

- **LOOM** [MacGregor, 1991] is a language and an environment to build intelligent applications. The core of LOOM is a knowledge representation system used to provide deductive support to the declarative part of the LOOM language. The declarative knowledge in LOOM consists of definitions, rules and facts. LOOM is a research project of the Information Science Institute research group at University of Southern California. The objective of the project is to develop advanced tools to represent knowledge and reasoning in AI.

- **OCML** [Domingue *et al.*, 1999] was developed at the Knowledge Media Institute of the Open University to provide operational modelling capabilities for the VITAL workbench of VITAL project. OCML, which is mainly based on Ontolingua, is a frame-based language with a Lisp-like syntax. Therefore, OCML provides primitives to define classes, relations, functions, axioms and instances. It is also possible to define rules and procedural attachments.

- **FLogic** [Kifer *et al.*, 1995] was developed at the Department of Computer Science of the State University of New York. FLogic integrates features from object-oriented programming, frame-based knowledge representation languages and first-order logic.

- **KIF** [Genesereth and Fikes, 1992] is a language designed for knowledge exchange between systems from different computer systems (created by different programmers at different times and with different programming languages). It uses a sintaxe like Lisp to express first-order predicate logic assertions. Although KIF is an expressive language, it is a low level language to represent ontologies.

- **XML** [Bray *et al.*, 2004] is a subset of the Standard Generalized Markup Language (SGML) and its goal is to enable generic SGML to be served, received and processed on the Web. XML has been designed for ease of implementation and for interoperability with both SGML and HyperText Markup Language (HTML).

- **Resource Description Framework (RDF)** [Lassila and Swick, 1999] was developed by the World Wide Web Consortium (W3C) as a language for processing metadata. It provides interoperability between applications that exchange machine-understandable information on the Web. RDF emphasises the facilities needed to enable automated processing of Web resources.

- **RDF Schema** [Brickley and Guha, 2002] is a language that, like RDF, was developed by W3C and that is specified in terms of the basic RDF information model.

- **RDF(S)** [Brickley and Guha, 2002] is a combination of RDF and RDF Schema. It is highly expressive since it allows the representation of concepts, taxonomies of concepts and binary relations. An inference machine has been created to be used with language, mainly to check constraints.

- **XOL** [Karp *et al.*, 1999] was developed by Pangea Systems Inc. and the Artificial Intelligence Center of SRI (Scientific Research Institute) International to facilitate the creation of shared ontologies (it was originally developed for use in bio-informatics). The language is intended to be used as an intermediate language for transferring ontologies between database systems, ontology-development tools or application programs.

- **SHOE** [Luke and Heflin, 2000] was developed at the University of Maryland. SHOE, which is an extension of HTML, aims to incorporate machine-readable semantic knowledge in Web documents and to provide specific tags for representing ontologies.

- **OIL** [Fensel *et al.*, 2000] is a proposal for a Web-based representation and inference layer for ontologies which combines the widely used modelling primitives from frame-based languages. OIL is compatible with RDF Schema and includes precise semantics for describing the meanings of terms.

- **DARPA Agent Markup Language (DAML)**[3] is an extension of XML and RDF.

- **DAML+OIL** [Horrocks *et al.*, 2002] is an updated version of DAML that provides a rich set of constructors to create ontologies and generate machine readable and understandable information.

- **OWL** [McGuinness and van Harmelen, 2004] is a W3C recommendation intended to be used when the information encapsulated in documents needs to be automatically processed by applications as well as humans. This language can be used to represent explicitly the meaning of terms and the relations between terms. OWL is more expressive than XML, RDF and RDF(S), and contains additional facilities to represent machine interpretable content on the Web. OWL is a review of the DAML+OIL language.

One of the questions that arises is what OWL provides that XML and XML Schema do not. XML provides syntax for structured documents, but does not define semantic constraints on the meaning of the documents. XML Schema is a language for restricting the structure of XML documents. RDF is a data model for representing objects and their relations which provides simple semantics and that can be represented in XML syntax. RDF Schema is a vocabulary for

[3]DAML, http://www.daml.org/, November, 2003

describing properties and classes of RDF resources with semantics for the generalisation of hierarchies of such properties and classes. OWL is an enhanced RDF with more vocabulary for describing properties and classes, including relations between classes, cardinality, equality, richer typing of properties, characteristics of properties and enumerated classes.

The concepts organised in taxonomies, binary relations and instances are the unique components that can be represented in all the languages. While Ontolingua and SHOE allow the creation of n-ary relations, in the other languages this kind of relations is represented by decomposition. Functions can be defined in Ontolingua, LOOM, OCML, FLogic, KIF, OIL, DAML+OIL and OWL. Formal axioms can be defined in Ontolingua, LOOM, OCML and Flogic. Rules can be defined in LOOM, OCML and SHOE. Procedures can be defined in CycL, Ontolingua, LOOM, OCML and KIF. In terms of inference mechanisms, several types are used.

When building an ontology, first, it is necessary to define the needs of the application in terms of expressiveness and inference services since each language allows components to be represented in different ways, depending on the reasoner. If the designer intends to design a heavyweight ontology and to make complex reasoning, the representation of basic information such as concepts, taxonomies and binary relations is insufficient.

3.8 E-commerce Ontologies

"The proliferation of different standards and joint initiatives for the classification of products and services reveals that B2B markets have not reached a consensus on coding systems, level of detail of their descriptions, granularity, *etc.*" [Corcho and Gómez-Pérez, 2001].

Large and consensual knowledge models for e-commerce applications are difficult and expensive to build. Nevertheless, several e-commerce ontologies have been proposed in the last years to ease the information exchange between customers and suppliers.

We briefly present the main e-commerce initiatives (UNSPSC, RosettaNet and e-cl@ass) in the next subsections. Other similar approaches, such as the North American Industry Classification System (NAICS[4]) and the Standard Classification of Transported Goods (SCTG[5]), exist and are also available.

[4]NAICS, http://www.naics.com, June, 2004
[5]SCTG, http://www.bts.gov/programs/cfs/sctg/welcome.htm, October, 2005

3.8.1 The United Standard Products and Services Codes

The United Standard Products and Services Codes (UNSPSC[6]) is a non-profit organisation composed of partners such as 3M, America Online (AOL), Arthur Andersen, British Telecom (BT), Castrol and others. It is a global commodity code standard that classifies general products and services. UNSPSC is designed to facilitate electronic commerce through the exchange of products' descriptions. The UNSPSC coding system is organised as a five-level taxonomy of products. These levels are organised as follows [Gómez-Pérez *et al.*, 2004]:

- **Segment** that represents the logical aggregation of families for analytical purposes.
- **Family** which represents a commonly recognised group of inter-related commodity categories.
- **Class** that represents a group of commodities sharing a common use or function.
- **Commodity** that represents a group of products or services that can be substituted.
- **Business Function** which represents the function performed by an organisation in support of the commodity. This level is seldom used.

UNSPSC contains about 20 000 products organised in 55 segments. The main drawbacks [Corcho and Gómez-Pérez, 2001] of UNSPSC are: (*i*) the lack of vertical cover of the products and services classified; (*ii*) the impossibility of attaching attributes to the concepts represented in the taxonomy; (*iii*) the fact that the design of the classification does not take into account the inheritance between the products described; and (*iv*) the inability to generate different views of the classification, *e.g.*, views that take into account cultural or social differences.

3.8.2 E-cl@ass

E-cl@ss[7] is a German initiative to create a standard classification of materials and services for information exchange between suppliers and customers. Leading international companies from different areas (*e.g.*, automotive, chemical, electronics, power generation and distribution, services, trade, *etc.*) are behind the development of e-cl@ss. Their common aim is to expand and internationally

[6]UNSPSC, http://www.unspsc.org, June, 2004
[7]e-cl@ass, http://www.eclass-online.com, June, 2004

distribute e-cl@ss in line with current and future market requirements [Cologne Institute, 2005].

The e-cl@ass is organised as a four-level taxonomy of concepts with a numbering code similar to the one used in UNSPSC. The levels are organised as segment, main group, group and commodity class. The e-cl@ass allows finding terms in different languages and it is available online.

E-cl@ass contains about 12 000 products organised in 21 segments. The e-cl@ass classification suffers from the same drawbacks as UNSPSC [Corcho and Gómez-Pérez, 2001].

3.8.3 RosettaNet

RosettaNet[8](RN) is a standard created by RosettaNet which is a self-funded, non-profit consortium of industry leaders. This standard is used and endorsed by more than 500 companies around the world.

RosettaNet, rather than using a numbering system like UNSPSC and e-cl@ass, uses a classification based on the name of the product. This classification is related to UNSPSC classification and provides the UNSPSC code for each product. The RosettaNet (RN) is organised as a two-level taxonomy, such as RN Category representing a group of products and RN Product representing a specific product. RosettaNet contains about 150 products organised in 14 categories.

According to [Corcho and Gómez-Pérez, 2001], the main drawback of RosettaNet is that there are only two classification levels which implies that the taxonomy structure is very simple. The RN classification also suffers from some of the problems of UNSPSC, namely, the lack of attributes and fact that the design does not consider the inheritance between concepts represented in the taxonomy.

3.9 Conclusions

As we point out in this Chapter, although there is not a unique definition for ontology, there is a large consensus about the need to use ontologies. Several authors identified different types of ontologies and the most representative were listed and discussed here. The design principles, which should be followed when building ontologies, were also discussed. Finally, several development tools and languages for ontology creation have been summarised.

There are several initiatives in the e-businesses domain for adopting standards to promote the interoperability and interchange of information between informa-

[8]Rosettanet, http://www.rosettanet.org, June, 2004

tion systems. However, there is a high level of overlap between them, allowing diverse classifications for the same product or service.

"Today most companies coding systems have been very expensive to develop. They typically take up to a year to create and, for each new item coded, it takes, on average, an hour and a half to assign a code. A company's suppliers usually do not adhere to the coding schemes of their customers - if they assign codes at all. Much duplicated effort and expense have gone into making codes. If there was a single universal coding convention that all companies could draw from - even if the companies wanted to customize it for specific purposes - there would be a great deal of saving" [Granada Research, 2001]. Usually, when companies represent products they adopt the names and classifications used in their catalogues or ontologies and do not waste time coding the products according to some external standard.

Ontologies are now a popular research topic in several research communities and are applied in distinct domains such as e-commerce, medicine, engineering, enterprise modelling, chemistry, knowledge management, *etc.* Moreover, ontologies are widely used by the MAS community in dramatically different domain applications. In this thesis we use ontologies in a MAS dedicated to B2B e-commerce.

Chapter 4

Ontologies in Multi-Agent Systems

This Chapter is focused on the role of ontologies in open MAS. In a MAS environment, when two autonomous, heterogeneous agents meet, they are likely to engage in communication; however, unless they share some content language ontology, they have little chance of understanding each other. Since both agents have private ontologies, it is necessary to provide some bidirectional translation between ontologies. The heterogeneity and interoperability problems, which hamper the communication between agents in open MAS, are explained and the operations involving ontologies are presented. The state of the art, concerning open MAS where agents using distinct ontologies interact, is summarised. Finally, the different approaches presented in the state of the art are analysed and compared.

4.1 Introduction

The efficient operation of the EI MAS requires communication standards, including communication protocols, communication languages and ontologies.

In the last years, standards for agent's communication language such as Knowledge Query and Manipulation Language (KQML) and FIPA Agent Communication Language (FIPA-ACL) were developed. Platforms for distributed agent's communication, such as JADE and Jini from Sun Microsystems, and content description languages, such as KIF and Content Language Specification (FIPA-SL), were also created. As far as ontologies are concerned, despite the current number of initiatives [Fensel *et al.*, 2000], [van Harmelen *et al.*, 2003],

[Karp *et al.*, 1999], there is neither a standard ontology language nor a standard ontology knowledge representation.

This lack of standardization, which hampers communication and collaboration between agents, is known as the interoperability problem [Willmott, 2001]. However, even if a standard ontology language existed, there would still be interoperability problems. If two agents that belong to the same domain are to communicate, then they need to agree on the terminology they will use to describe the domain. This terminology is what we consider to be the ontology; it describes the ontological commitments made by the set of agents, enabling communication in a certain discourse domain.

Agents in MAS are characterised by holding different views of the world. These distinct perspectives are explicitly defined through different ontologies, *i.e.*, diverse sets of concepts, relations and constraints [Falasconi *et al.*, 1996]. These different views need to be reconciliated by a commitment to use some common ontology in order to allow the agents to interoperate and cooperate while maintaining their autonomy.

Apparently, when a common domain ontology is used, it seems easier to understand the contents of a discourse. However, even with a common domain ontology, people may use different terms to represent the same item or choose a more general or detailed representation. Several difficulties are involved in providing a mapping capability between ontologies and several studies are being pursued trying to find a solution to that problem [Wache *et al.*, 2001].

It is also clear that when agents use different ontologies it is more difficult to establish a fruitful conversation. Suppose a specific negotiation where there is an enterprise agent capable of providing the required product at a better price and conditions than the remaining supplier agents. However, this agent may not even participate in such a negotiation because it is unable to realise that a negotiation involving that item (product/good/service) is undergoing. This scenario may be caused by the simple fact that the enterprise agent has a different ontology and, as a result, is unable to understand the meaning of the conversation.

In order to use simultaneously ontologies developed by independent sources, a system needs to perform ontological operations. This can be done through ontology integration [Pinto *et al.*, 1999], which means that the ontologies can either be merged into a new ontology or be kept separate. In both cases, the ontologies need to be aligned, meaning that it is necessary to define a mutual commitment or common understanding.

4.2 Terminology

Some of the terminology used to designate the operations between ontologies can be found in [Pinto, 2000], [McGuinness *et al.*, 2000] and [Klein, 2001]. To avoid possible misunderstandings, we now present the main definitions of the terms used throughout this thesis.

- **Combination** is the process of using two or more ontologies. Combination can be used to mean alignment, merge or integration of different ontologies. The combined ontologies usually hold data which is relevant to all ontologies involved.
- **Merge** is the process of building a single ontology though the merging of several source ontologies. Usually the source ontologies cover similar or overlapping domains.
- **Integration** is the process of creating a new ontology from two or more ontologies by overlapping the common parts. The domains of the source ontologies are different from the domain of the resulting ontology, but there is a relation between these domains.
- **Alignment** is the process of reaching global compatibility between two or more ontologies so that the resulting ontology is consistent and coherent.
- **Mapping** is the process of relating similar concepts or relations from different sources through some equivalence relation.
- **Articulation** is the set of connections between two aligned ontologies, *i.e.*, the specification of the alignment.
- **Translation** is the process of changing the representation formalism of the ontology while preserving its semantics.
- **Transformation** is the process of changing the semantics of the ontology, possibly also of the representation formalism, with the intent to make the new ontology suitable for purposes different from the original ones.
- **Version** is the result of creating a new ontology based on an existing ontology by introducing some changes. The new ontology can coexist with the original one.
- **Versioning** is the process of identifying the newly created ontologies by preserving the relations between the pre-existing ontologies and the data that ensures their consistence.

4.3 Heterogeneity and Interoperability Problems

In a B2B context, where agents are representing suppliers and buyers, heterogeneity may cause interoperability problems. Online e-commerce marketplaces for buying and selling products tend to bring together several heterogeneous suppliers and buyers. Each supplier and buyer may use its own formalism, concepts and characteristics to represent the same products. Even when both supplier and buyer use ontologies, these may differ significantly either syntactically or semantically. Whenever different ontologies are used, the different representations and terminologies prevail unless a formal mapping between the ontologies is established.

A simpler scenario occurs when the agents involved in a transaction have homogeneous knowledge representation structures and belong to the same discourse domain. In this case it is possible to develop and use a common ontology that provides a solution for the communication problem. The use of a common ontology guarantees the consistency and the compatibility of the information shared within the system. However, in distributed autonomous open systems, the intrinsic heterogeneity between components causes unavoidable interoperability problems. Ontologies not only are developed by different people, but suffer frequent modifications due to changes introduced in the conceptualisation of the ontology or in the domain. This continuous ontology update causes incompatibilities [Klein *et al.*, 2002] and makes the negotiation and cooperation processes difficult.

Heterogeneity can be regarded as an advantage as well as a disadvantage by system designers. On one hand, heterogeneity is positive because it is closely related to system efficiency. On the other hand, heterogeneity in data and knowledge systems is considered a problem since it is an important barrier for their interoperation. The lack of standards is the main obstacle to the exchange of data between heterogeneous systems [Visser *et al.*, 1997].

In this thesis, heterogeneity means that the agents representing enterprises communicate using different ontologies. Four types of heterogeneity are distinguished by [Visser *et al.*, 1997]:

- **Paradigm heterogeneity** that occurs if distinct agents express their knowledge using different modelling paradigms.
- **Language heterogeneity** which occurs if distinct agents express their knowledge in different representation languages.
- **Ontology heterogeneity** that occurs if distinct agents make different ontological assumptions about their domain knowledge.
- **Content heterogeneity** which occurs if distinct agents express different knowledge.

While paradigm and language heterogeneity are examples of non-semantic heterogeneity, ontology and content heterogeneity are examples of semantic heterogeneity.

Although there are no automatic procedures available, different tools and techniques for the mapping, aligning, integration and merging [Hakimpour and Geppert, 2001], [McGuinness *et al.*, 2000], [Noy and Musen, 2000], [Pinto, 1999], [Stumme and Maedche, 2001] of ontologies are at disposal. These tools require human supervision to ensure the establishment of correct mappings. Since in open MAS inter-ontology mappings need to be established on a large scale, this prerequisite is unacceptable [van Diggelen *et al.*, 2005].

Mapping is a difficult task whose success depends on the detection and resolution of ontology mismatches. Recent research about ontological discrepancies has been conducted by [Visser *et al.*, 1997] and [Klein, 2001]. However, none of the available tools tackle all types of discrepancies (structure mismatch, attribute-assignment mismatch and concept-and-term mismatch) [Hameed *et al.*, 2001]. Unfortunately, ontologies are not a global *panacea* and, so far, no one has constructed an ontology that is comprehensive enough. Moreover, even if such ontology did exist, it probably would not be adhered to, considering the dynamic and eclectic nature of the Web and other information sources [Hugns, 2004]. One way of solving ontology discrepancies is to automatically detect the similarities between the involved ontologies.

Similarity evaluations between ontologies may be achieved whenever the involved ontologies share some components. If two ontologies have at least one common component (relation, hierarchy, type, *etc.*) then they may be compared. Since the characteristics (attributes) of concepts (products) capture the details of the concepts, they provide a good opportunity to find similarities.

Problems with data heterogeneity are already well known within the distributed database systems community [Wache *et al.*, 2001]. They can be distinguished as follows:

1. **Structural heterogeneity** that occurs when different information systems use different structures to store their data.

2. **Semantic heterogeneity** which considers the contents of an information item and its intended meaning. There are three main causes for semantic heterogeneity:

    - **Confounding conflicts** that occur when information items seem to have the same meaning, but are in fact distinct, owing to different temporal contexts.

- **Naming conflicts** which occur when the naming schemes of the information differ significantly.
- **Scaling conflicts** that occur when different reference systems are used to measure the same value. An example is the use of different currency units.

Figure 4.1 [Malucelli and Oliveira, 2004b] shows a simple example where, using Unified Modeling Language (UML) schemes, we may observe structural and semantic conflicts. Ontology A and Ontology B represent different views of the music compact disc domain. The discovery of corresponding items between both ontologies is not straightforward. The scenario depicted shows that: (*i*) Ontology A has an **Audio Compact Disc** concept with a **publisher** attribute; (*ii*) Ontology B has a **Music CD** concept with a **publishing House** attribute; and (*iii*) the **publisher** and **publishing House** attributes have the same meaning. Thus, the relation **is composed by** between **Audio Compact Disc** and **Artist** in Ontology A corresponds to the relation **has-performer** between **Music** and **Performer** in Ontology B.

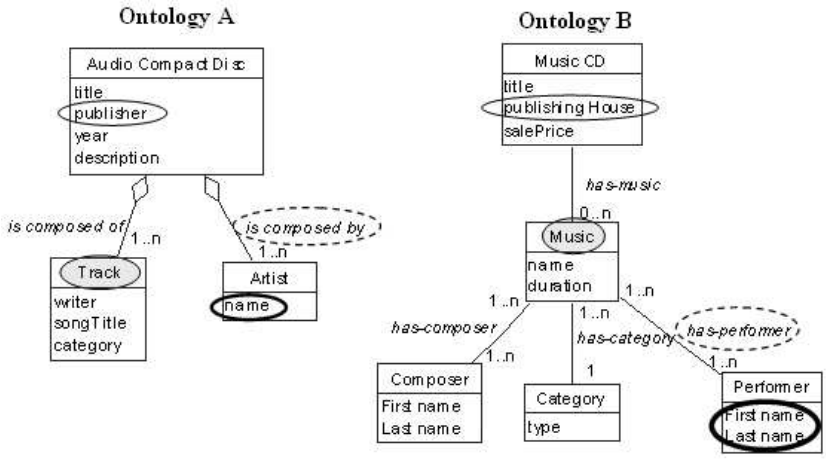


Figure 4.1: Structural and Semantic Conflicts

Whereas the specification a simple product like a music compact disc is relatively easy and there is a chance of always finding similarities in the description, defining a more complex product like a car or a plane presents serious challenges.

4.4 Ontology Approaches

Agents may use different ontologies to represent their views of a domain, leading to potential ontology mismatches. [Wache *et al.*, 2001] present three different

directions on how to employ ontologies in a distributed scenario: the single ontology approach, the multiple ontology approach and the hybrid ontology approach. Figures 4.2, 4.3 and 4.4 illustrate the three types of architectures derived from these approaches:

- **Single Ontology Approach** uses a global ontology to provide a shared vocabulary for the specification of the domain semantics. All information sources are related through the global ontology. The global ontology can also be a combination of several specialised ontologies.

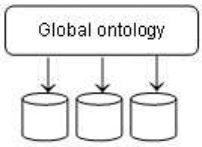


Figure 4.2: Single Ontology Approach

- **Multiple Ontology Approach** allows each information source to be described by its own ontology. In principle, the "source ontology" can be a combination of several other ontologies but it cannot be assumed that the different "source ontologies" share the same vocabulary.

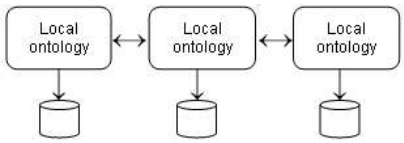


Figure 4.3: Multiple Ontology Approach

- **Hybrid Ontology Vocabulary** is similar to multiple ontology approaches where the semantics of each source is described by its own ontology. However, in order to ensure that the source ontologies are comparable, they are built upon one global shared vocabulary. The shared vocabulary contains basic terms of the domain.

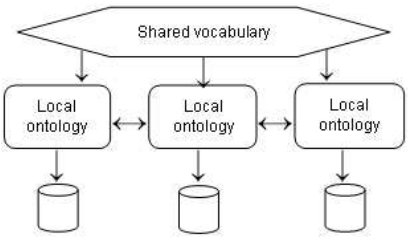


Figure 4.4: Hybrid Ontology Approach

In this thesis we are dealing with the multiple ontology approach where each agent explores its own ontology. We chose this approach because the overall system architecture is an open MAS.

4.5 State of the Art

Research concerning ontology mapping between different ontologies using semantic relations, lexical taxonomies, semantic similarities, linguistic similarities of terms, taxonomic relationships and text information [Jiang and Conrath, 1997], [Klein *et al.*, 2002], [Rodríguez and Egenhofer, 2003], [Maedche and Staab, 2001], [Mitra and Wiederhold, 2002], [Welty and Guarino, 2001] has been conducted by several groups. The research goals as well as the application domains are highly diversified.

In the case of open heterogeneous MAS, FIPA has identified and analysed the different types of interoperability problems that arise and has, consequently, proposed the creation of an Ontology Agent (OA) to assist the community of agents in the alignment of ontologies. However, the implementation of such service is left to system developers. Furthermore, the FIPA Ontology Service Specification[1] classifies this domain-dependent task as very complex and possibly not always achievable. An implementation of the OA is presented in [Suguri *et al.*, 2001]. It is a sample application of an ontology shopping service that integrates multiple database schemata to verify and demonstrate the specification. However, no mechanism is provided to match terms between different ontologies.

Only in recent years has the problem of handling different ontologies in MAS been addressed. In this Section we provide a brief summary of the main contributions in this domain, which are:

- [Steels, 1998] proposes a complex adaptive system approach of an ontology and a shared lexicon in a group of distributed agents with only local interactions and no central controlling agency. The ontologies are adaptive. Agents have limited knowledge, *i.e.*, they cannot inspect the internal states of other agents. Moreover, agents engage only in local interactions with other agents and are autonomous, *i.e.*, they acquire their own knowledge and decide for themselves what to do and how to communicate. The interaction between agents is modelled as a game, so the language is a game language. To perform a speech act, the speaker must conceptualise the objects so as to find a description which distinguishes the topic from other objects in the context. This requires an ontology, *i.e.*, a set of distinctive

[1] FIPA Ontology Service Specification, http://www.fipa.org/specs/fipa00086/XC00086C.html, June, 2002

features. The speaker must find words to encode the distinctive features thus found and transmit these words to the hearer. The hearer receives the transmitted message, decodes it into one or more possible interpretations and checks whether the interpretations are compatible with the present situation. If they are compatible, the game succeeds. Some failures occur due to missing categories in the ontology of the speaker or hearer and missing or wrong linguistic conventions. New categories are created by extending the ontology, by creating a new distinctive feature or refining existing distinctive features. New linguistic conventions are generated by creating new words or by adopting words used by the speaker. Agents record the success of words and choose to use the most successful words. The ontology creation produces distinctive lexicons. Lexicalisation becomes successful whenever the words are also used by other agents. Each agent has its own set of words and a lexicon which is initially empty. An agent can associate a single word with several meanings and a given meaning with several words. The words are matched using distance measurements. When performing experiments with MAS all agents use the same ontology creation mechanisms.

- [Bailin and Truszkowski, 2002] describe an approach to ontology negotiation that allows Web-based information agents to resolve mismatches in real time without human intervention. The system employs the WordNet lexical database as a data source to extend each ontology's concept repertoire. However, the essencial mechanism relies on the information exchanges between agents since WordNet by itself does not allow the agents to interpret each other's concepts. These data exchanges are structured by the rules of the Ontology Negotiation Process (ONP) and allow each agent to ask for the clarification of previous messages and for the confirmation or correction of attempted interpretations. The proposed ONP consists of the tasks of interpretation, clarification, relevance evaluation and ontology evolution. Although the interpretation and clarification tasks may take the form of simple substitution by synonyms, the protocol provides more complex forms such as formal logical definitions, operational descriptions and approximations to a concept's meaning. In this work, the authors explore the possibility of ongoing ontology evolution under the control of a dedicated autonomous agent. Additionally, they define an API tool to provide functionalities to support ontology evolution. The terms exchanged between the standard agents consist either of queries or answers to queries. Both contents can be viewed as keywords describing the document that is either desired (query) or found (answer). The negotiation process culminates with one or both agents modifying their ontology to introduce a new concept, a new distinctive feature or simply a new term for an existing concept.

- [van Eijk *et al.*, 2001] developed a communication mechanism in which translators between the vocabularies of agents are generated. These translators are not defined by the programmer, but instead, they are dynamically constructed during the execution of the system and are based both on the information the agents exchange and on their underpinning ontologies. The dynamic construction of translations takes place during successive communication steps, in which a particular information is exchanged between agents. This means that the information provided by a telling agent is translated into the vocabulary of the asking agent. During each communication step, the translations from the previous communication rounds are refined to include the present information exchanged. The authors assume that the ontologies are represented by a first-order theory and, in the typical situation in which this theory is finite, simply by a first-order formula. In this approach there is no global shared ontology and each agent has its own private ontology. Initially, before launching the system, no connections between individual ontologies of agents are assumed. Instead, the connections are constructed dynamically during execution. Moreover, these connections are established on a demand-driven basis, *i.e.*, only the connections that are required during some communication step are established between the ontologies of both agents.

- [Tzitzikas and Meghini, 2003] consider peer-to-peer systems in which peers employ taxonomies for describing the content of their objects and for formulating semantic-based queries to the other peers of the system. Each peer uses its own taxonomy and is equipped with inter-taxonomy mappings in order to carry out the required translation tasks. As these systems are *ad-hoc*, peers should be able to create or revise mappings on demand and at runtime. Since there is no central server or mediator, each participating source must have (or be able to create) mappings or articulations between its conceptual model and the conceptual models of its neighbours in order to be able to translate the received queries to queries that can be understood (and thus answered) by the recipient sources. This work introduces a data-driven method for automatic taxonomy articulation, which the authors call an ostensive method because the meaning of each term is explained by ostension. In this data-driven approach, the mappings are discovered by examining how terms are used in indexing the objects. This methodology does not make any assumptions on how the involved taxonomies are constructed or how they are used, but it requires the presence of two databases that contain several common objects.

- [Burnstein *et al.*, 2003] have sketched an approach to automatic derivation of "glue code" - programs for translating the output of a source agent to the

input representation of a target agent. The authors describe how agents can autonomously derive transformation functions, *i.e.*, "glue code", to translate between heterogeneous ontologies. The methodology is based on λ-*calculus*. In this approach, it is assumed that there is a common underlying theory to which the private ontologies of the agents are linked. However, the approach does not provide any method for establishing a mapping. The common theory of the application domain provides symbols for the concepts, operations and relationships in the domain. Its axioms constrain the meanings of the vocabulary.

- [Doherty *et al.*, 2005] combine logic-based techniques with approximate reasoning. This proposal provides software or robotic agents with the ability to ask each other approximate questions concerning unclear or unknown terms and actions. Agents have a local ontology consisting of concepts/relations, which is a subset of a global ontology. Although agents have common concepts, there are some disjoint concepts/relations. Each agent can communicate in the language of the other agents (by using a first-order or fixpoint formula) and has a mediation function. The generated formulas can be understood by the agent because they are formulated using its own concepts/relations. As a result, these formulas can be used to query the relational database of the respondent agent. The data or knowledge bases associated with the agents are formalised as approximated databases and the questions that may be asked are represented as first-order or fixpoint queries. The weakest sufficient and strongest necessary conditions are used to model approximate queries with specific sub-languages common to each pair of agents involved in an utterance.

- [Williams *et al.*, 2003] propose a methodology for agents to develop local consensual ontologies as a means to support the communication within a multi-agent system of B2B agents. The agents have to be able to find related services (ontology concepts) between inter-organisation ontologies and intra-organisation ontologies. Agents may have diverse ontologies, but, before extensive translation can take place, they must be able to relate services at a slightly higher level. Agents create only relatively small local consensus ontologies to facilitate the discovery and understanding of the Web service for current and future B2B systems. The approach allows finding semantic and syntactic similarities by comparing, for each pair of agents, both ontologies without the use of a global ontology and by merging these ontologies into a local consensual ontology. They provide a straightforward tool for generating ontologies which are stored and represented in XML. The authors demonstrate how this approach works by merging two diverse Web service messaging ontologies.

- [Wiesman and Roos, 2004] propose a domain-independent methodology for handling interoperability problems by learning mappings between ontologies. The learning method is based on exchanging instances of concepts defined in the ontologies. They focus on establishing a mapping between two concepts, one from each ontology. No restrictions are placed on the structure of a concept. A concept may be defined as an aggregation of attributes and sub-concepts. This aggregation may even, directly or indirectly, contain the concept that is being defined. The way the agents establish a mapping is inspired by a game language called joint attention, where an agent tries to interpret the utterances of another agent by creating and evaluating associations between the received utterances and categorisations of observed entities. After establishing the corresponding concepts, one of the agents proposes associations between the labels of two utterances on the basis of the proportion of corresponding words that the two labels have in common.

- [van Diggelen *et al.*, 2005] address the problem of establishing a suitable communication vocabulary in a formal and abstract way. Each agent has a private ontology which is incomprehensible to the other agents. They assume that there is a common ground ontology in which the private ontology of each agent is rooted. This common ground enables them to discover the relations between foreign concepts and their private ontologies. All the terms used in a private ontology can be expressed as some complex combination of basic terms defined in the ground ontology. Agents use as communication vocabulary their own private concepts. Thus, the communication is more efficient than without using the complex combination of basic terms for a defined ground ontology. The set of shared concepts is represented in the communication vocabulary. Ontologies are formalized using L, which is equal to the description logic language - Attribute Concept Language with Complements, but without roles. The communication vocabulary is also formalised as an ontology. For agents to adapt to a communication vocabulary, they must be able to discover the relations between the concepts in the communication vocabulary and the concepts in their private ontology. The private concepts of each individual agent can eventually be defined by means of the shared ground ontology since every primitive concept is present in this ground ontology. To preserve soundness, the agents translate (adopting a distribution function) private concepts into equivalent or more general shared concepts. Since an optimal communication vocabulary should contain sufficiently accurate concepts, the shared concepts should not be too general. A sending agent should be allowed to translate a private assertion into a more general concept in the communication vocabulary as long as the translated concept remains equivalent to

the original concept relative to the receiving agent's ontology. The authors develop dialogue strategies in which the agents can apply to establish an optimal common communication vocabulary or an optimal communication distribution function.

- Our approach [**Malucelli and Oliveira, 2006**] is focused on the resolution of negotiation conflicts in a B2B domain. We define a set of services for tackling with the interoperatiblity problems which arise during inter-agent communication. The most important service provided is the resolution of ontological conflicts. We propose a methodology to assess the similarities between the concepts represented in the different ontologies without the need to build *a priori* a shared ontology. The identified similarities are regarded as bridging elements between the involved ontologies and can be used to support the inter-agent negotiation process. Our approach differs from the described proposals since it uses a mediator agent called OSAg. This agent is responsible for the resolution of all negotiation conflicts that occur within the MAS. Although we used the Protégé ontology editor and OWL ontology language to create and store the ontologies, these are not requirements. MAS developers are free to create and store ontologies using different editors or languages and still use our approach. Each agent has its own private ontology and is ignorant of the ontologies of the other agents. Moreover, the private ontologies remain unchanged during the entire negotiation process. The matched concepts are memorised by the OSAg and kept for future negotiation rounds. The mapping between ontologies is established by comparing, for each pair of concepts, the attributes (grouped by data type), the relation *has-part* and the descriptions of the concepts. The comparison includes both syntactic and semantic measurements.

4.6 Comparative Analysis

The previous Section summarises the most relevant work that addresses the problem of supporting communication among agents using different ontologies. In particular, we are interested in comparing the described approaches in the light of our application domain - the B2B MAS. Some important questions arise:

1. Do the agents have their own Private Ontology (PO) or/and Shared Instances (SI) and/or Shared Ground Terms (SGT)?

2. Do authors use ontology development tools (Ontology Editor (OE)) to create the ontologies? If yes, which ontology editors are used?

3. Which Ontology Languages (OL) are used to represent the ontologies?

4. Are there any Mandatory Ontology Creation Mechanisms (MOCM), *i.e.*, a specific editor and language, for all the agents involved in the MAS?

5. Is each Individual Agent (IndA) responsible for building the mapping between the different existing ontologies or is there a global ontology Mediator Agent (MA) responsible for providing this service?

6. Is it a B2B Application Domain (B2B-AD)?

7. How are the created ontologies classified (Lightweight Ontologies (LwO) or Heavyweight Ontologies (HwO))?

8. Which are the Ontology Mapping Methods (OMM) used?

We have created three tables where these main characteristics are compared. Some of the approaches analysed omit details regarding some of the selected comparative features. Table 4.1 answers questions 1, 2 and 3. Table 4.2 provides answers for questions 4, 5 and 6. Finally, Table 4.3 answers questions 7 and 8.

Approaches	**PO**	**SI**	**SGT**	**OL**	**OE**
Steels, 1998	✓		✓	Language game	
Bailin and Truszkowski, 2002	✓			API	
van Eijk *et al.*, 2001	✓			First-order theory	
Tzitzikas and Meghini, 2003	✓	✓	✓		
Burnstein *et al.*, 2003	✓		✓	Data structure	
Doherty *et al.*, 2005	✓		✓		
Williams *et al.*, 2003	✓	✓	✓	XML	
Wiesman and Roos, 2004	✓	✓			
van Diggelen *et al.*, 2005	✓		✓	L	
Malucelli and Oliveira, 2006	✓			OWL/Free	Protégé/Free

Table 4.1: Comparison Considering the Ontology Approach, Ontology Language and Ontology Editor

Approaches	**MOCM**	**IndA**	**MA**	**B2B-AD**	**Other AD**
Steels, 1998	✓	✓			✓
Bailin and Truszkowski, 2002	✓	✓			✓
Van Eijk *et al.*, 2001	✓	✓			
Tzitzikas and Meghini, 2003	✓	✓			✓
Burnstein *et al.*, 2003	✓				✓
Doherty *et al.*, 2005	✓	✓			✓
Williams *et al.*, 2003	✓	✓		✓	
Wiesman and Roos, 2004	✓	✓			✓
van Diggelen *et al.*, 2005	✓	✓			✓
Malucelli and Oliveira, 2006			✓	✓	

Table 4.2: Comparison Considering Ontology Creation Mechanisms, Inter-ontology Mapping and the Application Domain

These tables summarise the main characteristics of the state of the art in our research domain. We concluded that: (*i*) the majority of the approaches does not use ontology editors to build ontologies; (*ii*) the ontology languages differ; and (*iii*) there are no standard ontology languages, *i.e.*, each system uses a specific ontology language to facilitate the interoperability. As a result, these systems require the use of the exact same mechanisms to create all ontologies involved. Furthermore, the majority of the ontologies are lightweight ontologies and few are approaches concerning agents using different ontologies in the B2B domain. Ontology mapping is present only in some approaches and, whenever it is available, the mapping is performed by the individual agents. As a result, any agent that wishes to participate in these MAS must be equipped with some ontology mapping methodology.

The research literature devoted to EI does not emphasize the importance of having ontology mapping services. In this respect and as far as we know, our approach is original. Existing approaches either avoid the resolution of the heterogeneity problems or develop a domain ontology to be used by all the MAS agents. An example of the first case is the work of [Dignum, 2001], who points out the need to have a common ontology available for all parties inside the institution, describing both general and domain-dependent concepts. The latter case is represented by [Vázquez-Salceda and Dignum, 2003]. These authors model the specification of norms within electronic organisations, which are formed inside an EI. They start from an abstract institutional level and address the translation of abstract norms into concrete ones, which are described in terms of a particular e-organization's ontology. A domain-ontology defines the vocabulary to be used by all the agents in an e-organization.

Approaches	**LwO**	**HwO**	**OMM**
Steels, 1998	✓		Linguistic communication, distance between objects and association-set.
Bailin and Truszkowski, 2002	✓		Queries on WordNet lexical database, locate synonyms in the source agent's ontology, provide instances and generalisation from source's ontology.
van Eijk *et al.*, 2001	✓		The mapping consists of a set of translation formulas each expressing an equivalence between expressions.
Tzitzikas and Meghini, 2003	✓		Ostensive articulation for taxonomy-based sources.
Burnstein *et al.*, 2003		✓	No mapping between ontologies is provided.
Doherty *et al.*, 2005	✓		No mapping between ontologies is provided.
Williams *et al.*, 2003	✓		Syntactic and semantic similarity and semantic relation discovery.
Wiesman and Roos, 2004	✓		Learning method based on exchanging instances of concepts defined in the ontologies.
van Diggelen *et al.*, 2005	✓		Distribution function (communication vocabulary).
Malucelli and Oliveira, 2006	✓		Syntactic and semantic similarity between attributes, relations and descriptions.

Table 4.3: Comparison Considering the Ontology Classification and the Ontology Mapping Methodology

4.7 Conclusions

One way to ensure an efficient communication within a MAS is to use a shared ontology, *i.e.*, to make the agents use a common domain ontology. However, there are several reasons why this is not the best way to proceed [Steels, 1998]:

- It is hard to imagine how there could ever be a world-wide consensus about ontologies and associated languages for every possible domain of MAS application.
- MAS are a typically open systems, which means that the ontologies rather than being defined once and for all, are expected to expand as new needs arise.

- MAS are typically distributed systems with no central control unit. This raises the issue of how evolving ontologies might spread to already operational agents.

The problem of using different ontologies in MAS has only recently been addressed. We reported the most representative work in this domain as well as identified and compared the main characteristics of the different approaches found in the literature.

Our research has been conducted in order to provide services to facilitate the negotiation between agents in a B2B context. One of these services - the most important one - is devoted to the resolution of inter-agent communication problems. In order to try to solve this type of problems, some matching techniques and mechanisms have been researched and will be thoroughly explained in Chapters 5 and 6. Our mapping approach, which exhibits some new and different aspects when compared with the existing ones, will be explained in Chapter 6.

Chapter 5

Ontology Mapping

In a MAS B2B scenario, agents from diverse origins and holding knowledge represented through different ontologies need to interact. Therefore, it is necessary to establish correspondences between the individual ontologies so that the agents are able to understand each other and engage in fruitful negotiations. The process of implementing the correspondence between ontologies is named ontology mapping. Ontology mapping finds the correspondent relations and similarities between two ontologies. There are several approaches that can be used to determine the similarity between ontologies. This Chapter overviews the main semantic relatedness measurements, element-level techniques and structure-level techniques. The combination of different ontology mapping approaches in order to obtain a more precise and trustworthy result is also addressed. Finally, the most representative ontology mapping tools (systems as well as ontology development frameworks) are described.

5.1 Introduction

Agents are computing entities with the ability to communicate. This communication is achieved trough speech-act inspired languages which determine the content of the messages and enable agents to position themselves within a particular interaction context. The actual content of the messages is expressed in knowledge representation languages and often refers to some ontology. Consequently, when two autonomous and independently designed agents meet, they have the possibility of exchanging messages, but little chance of understanding each other unless they share a common content language ontology. Thus, it is necessary to provide

the possibility for these agents to match their ontologies [Shvaiko and Euzenat, 2005].

A possible solution to receive and understand messages expressed in an unknown ontology is to implement an ontology mapping process. This ontology mapping process can be seen either as a service, which will be requested whenever necessary, or as a module residing in each agent, *i.e.*, the individual agents implement ontology mapping for message translation. This way, agents that meet for the first time and use different ontologies can interact and engage, with a high probability, in a meaningful conversation.

Mapping is an important and critical operation in traditional applications, such as information integration, query answering, data translation and data warehousing, as well as in dynamic applications, such as agent communication, peer-to-peer databases and Web services integration [Shvaiko and Euzenat, 2005].

Ontology mapping is the process of finding correspondences between the concepts of two ontologies. If two concepts correspond, then they mean the same thing or closely related things [Dou *et al.*, 2003]. Currently, the mapping process is regarded as a promise to solve the heterogeneity problem between ontologies since it attempts to find correspondences between semantically related entities that belong to different ontologies. It takes as input two ontologies, each consisting of a set of components (classes, instances, properties, rules, axioms, *etc.*), and determines as output the similarity matchings.

There is some confusion about the meaning of translation and mapping. The word translation is used by authors to describe two different things. The first meaning refers to the translation between formal languages, *e.g.*, from Ontolingua to Prolog. This translation, which changes the syntactic structure of axioms but not the vocabulary, is unrelated to ontology mapping. The second meaning, which is related with ontology mapping, is concerned with the translation of vocabulary. The difference between translation and mapping is that the former denotes the process of defining the collection of functions used to specify the correspondence between concepts and relations from both ontologies, while the latter is the application of the mapping functions that actually translate the sentences from one ontology into another. This presupposes that the ontologies share the domain in which the respective vocabularies are interpreted [Kalfoglou and Schorlemmer, 2003].

In order to find out which mappings need to be created, the similarities between concepts, relations and other components have to be established. The similarity between ontologies can either be established manually or automatically using the Match operator [Bruijn *et al.*, 2004]. The Match operator takes as input two ontologies and returns as output a list of similarities between entities of the two source ontologies [Rahm and Bwenstein, 2001].

The similarity relation is essential in reasoning frameworks where unclear or approximate concepts are represented. Since there are different types of measurements and techniques that can be applied to detect similarities between ontologies, the selection of the most appropriated metric depends on the features of the data available.

5.2 Similarity, Relatedness and Distance

Similarity is a special case of semantic relatedness. Similar concepts are semantically related because of their similarity. However, dissimilar concepts may also be semantically related by some lexical relationship such as meronymy (*part-of*, *e.g.*, window-house), antonymy (*opposite-of*, *e.g.*, hot-cold), or just by any kind of functional relationship or frequent association (*e.g.*, ocean-cruiser).

"Two concepts are close if their similarity or relatedness is high, and otherwise they are distant" [Budanitsky and Hirst, 2005]. Similarity, semantic relatedness and semantic distance measurements are used in applications such as word sense disambiguation, determination of the structure of texts, text summarization and annotation, information extraction and retrieval, automatic indexing, lexical selection, *etc.* In our case, we are applying semantic measurements to establish the similarity between concepts from different ontologies.

There are several approaches that can be applied to determine the semantic relatedness between two ontologies, namely, dictionary-based, *thesaurus*-based and approaches based on the semantic network. They are summarised in the next subsections using [Budanitsky, 1999] and [Budanitsky and Hirst, 2005] classification.

5.2.1 Dictionary-based Approaches

Some initiatives have been made to adapt dictionaries to the task of measuring semantic distance computationally. The Longman Dictionary of Contemporary English (LDOCE) was the first dictionary available to researchers in a machine-readable format. LDOCE is the most widely used English dictionary for language processing. The use of controlled vocabulary in headword (*lemma*) definitions is the most exploited LDOCE feature. A natural way of turning a dictionary into a semantic network is to create a node for every headwork and connect this node to the nodes corresponding to all the headwords encountered in its definition.

Spreading activation on the network can directly compute the similarity between any two words in the Longman Defining Vocabulary and, indirectly, the similarity of all the other words in LDOCE. The similarity represents the strength

of lexical cohesion or semantic relation and also provides valuable information about similarity and coherence among texts [Kozima and Furugori, 1993].

5.2.2 Thesaurus-based Approaches

A *thesaurus* is a standardised list of terms, often restricted to a specialised field or subject, in which the terms with similar meaning are grouped together. The hierarchy of a *thesaurus* usually includes classes, categories and subcategories. An important *thesaurus* feature is its index, which contains category numbers along with labels representative of those categories for each word. A *thesaurus* merely groups related words without attempting to explicitly indicate how and why they are related. No numerical value for the semantic distance can be obtained, thus, algorithms using a *thesaurus* compute implicitly a distance and return a *boolean* value of close or not close similarity [Morris and Hirst, 1991], [Okumura and Honda, 1994].

5.2.3 Semantic Network based Approaches

A semantic network is a graphic notation for representing knowledge described as interconnected nodes and arcs. Semantic networks were initially used in the fields of philosophy, psychology and linguistics. Currently, they are implemented and used in Artificial Intelligence (AI) and automatic machine translation. The semantic network representation may be used to represent knowledge as well as to support automated reasoning systems.

Most of the methods presented in this subsection are based on WordNet [Miller, 1995], a lexical database constructed on lexicographic and psycholinguistic principles, under active development for the past twenty years at the Cognitive Science Laboratory of Princeton University. It is designed for use under program control and provides an effective combination of traditional lexicographic information and modern computing.

The success of the use of WordNet in computational linguistic is partly due to its vast coverage containing 138 838 English words, compared with the 35 958 in the LDOCE and 43 943 in Roget's Thesaurus. Furthermore, it contains hundreds of thousands of links that represent important psycholinguistic relationships between words which are, in turn, grouped according to syntactic categories to reflect the different sorts of relationships observed in the diverse categories [Veres, 2004].

In WordNet a form is represented by a string of ASCII characters and the sense is represented by the set of one or more synonyms (synset) with the same meaning. WordNet contains more than 118 000 different word forms, more than

90 000 different word senses and includes the following semantic relations: synonymy (*same-name*), antonymy (*opposite-name*), hyponymy (*sub-name*), hypernymy (*super-name*), meronymy (*part-name*) and holonymy (*whole-name*). Each of these semantic relations is represented by pointers between word forms or between synsets (sets of synonyms). Moreover, definitional glosses are included in most synsets along with the synonyms that represent the meaning.

WordNet has been used for automated disambiguation of ontology terms. In this work, we access WordNet locally and apply WordNet-based measurement techniques of semantic relatedness to help in the ontology mapping process. A detailed description of our approach can be found in Chapters 6 and 7.

Computing Taxonomic Path Length

A simple way to compute semantic relatedness in a taxonomy such as WordNet is to analyse the underlying graph and identify relatedness using path length measurements between concepts. "The shorter the path from one node to another, the more similar they are" [Resnik, 1995]. The most representative measure techniques used to compute taxonomic path length based on semantic networks (WordNet) are:

- **Hirst and St-Onge**

  Hirst and St-Onge (HSO) measure technique considers more relations than the *is-a* relation and, thus, is able to calculate the relatedness between speech parts. HSO can also determine the relatedness between nouns and verbs [Pedersen *et al.*, 2003].

  Hirst and St-Onge (HSO) classify two words in WordNet as strongly related if one of the following conditions holds:

  - They share a synset.
  - They are associated with two different synsets that are connected by the antonymy relation.
  - One of the words is a compound word that includes the other and there is some kind of link between synsets associated with each word.

  Two words are related in a medium to strong or regular relation if there is some path connecting a synset associated with each word. If a path that is neither too long nor too winding exists, then there is a medium to strong relation between the concepts. The score given to a medium to strong relation considers the path length between the concepts and the number of changes in direction of the path. "The longer the path and the more changes of direction, the lower is the final weight" [Budanitsky, 1999].

HSO measure is summarised by the following formula (5.1):

$$Path\_weight = C - path\_length - (k * \#\_change\_in\_direction) \quad (5.1)$$

where C and k are constants.

- **Leacock and Chodorow**

  Leacock and Chodorow (LCH) calculate the length of the shortest path between two nouns (concepts) in an *is-a* hierarchy. The similarity measurement proposed by LCH finds the shortest path between two concepts by counting up the number of edges between the meanings in the *is-a* hierarchy of WordNet. The retrieved value is then scaled by the maximum path length in WorldNet's *is-a* hierarchy. If no error occurs, a relatedness value is obtained by taking the negative logarithm of this scaled value. Otherwise, an error string is created. This is also the case if no path exists between the two words, *e.g.*, when one of the words is not represented in WordNet.

  LCH measure is summarised by formula (5.2):

$$sim(c_1, c_2) = [max - log(length(c_1, c_2)/(2 * D))] \quad (5.2)$$

  where c_1 and c_2 are concepts, $length(c_1, c_2)$ is the shortest path length and D is the maximum depth of the taxonomy.

- **Wu and Palmer**

  This measure technique is based on the distance between a concept and the root node. Wu and Palmer (WUP) calculate the distance from the root to the most specific node that intersects the path of the two concepts in the *is-a* hierarchy. This intersecting concept is the most specific concept that the two concepts under analysis have in common and is known as the lowest common subsumer (lcs). The distance of the lcs is then scaled by the sum of the distances of the individual concepts to the node. Thus, the relatedness is calculated considering the depths of the two synsets in the WordNet taxonomies along with the depth of the lcs.

  WUP measurement is summarised by formula (5.3):

$$sim(c_1, c_2) = 2 * depth(lcs(c_1, c_2))/depth(c_1) + depth(c_2) \quad (5.3)$$

  where *depth* is the distance from the concept node to the root of the hierarchy.

5.2.4 Integrated Approaches

The integrated approaches differ from the other approaches because the measure techniques applied use a linguistic corpus (a body of one or more texts selected

for analysis) to increase the information already present in the network. Two of these techniques are presented below:

- **Resnik**

  Resnik (RES) measure technique is based on the information content (IC) of noun concepts as found in the *is-a* hierarchies of WordNet. The principle behind this measure technique is that the semantic relatedness of two concepts is proportional to the amount of information they share. The quantity of information common to the two concepts is determined by the information content of their lowest common subsumer [Pedersen *et al.*, 2003].

  RES measure is summarised by formula (5.4):

  $$sim(c_1, c_2) = IC(lcs(c_1, c_2)) \tag{5.4}$$

  This measure technique does not take into consideration the information content of the concepts being measured nor the path length between them.

- **Jiang and Conrath**

  Jiang and Conrath (JCN) propose a measure technique based on the semantic distance between nouns. The difference between the information content of the individual concepts and that of their lowest common subsumer will reveal how similar or different they are. If the sum of their individual information contents is close to that of their lowest subsumer, then it suggests that the concepts are close in the concept hierarchy. Thus, the authors take the sum of the information content of the individual concepts and subtract it from the information content of their lowest common subsumer.

  JCN measure is summarised by formula (5.5):

  $$Dist(c_1, c_2) = IC(c_1) + IC(c_2) - 2 * IC(lcs(c_1, c_2)) \tag{5.5}$$

  Since JCN is a distance measurement, concepts that are more similar have a lower score than less similar ones.

The performance of the semantic relatedness measure techniques presented depends on the application domain features. Consequently, we evaluated the performance of these techniques in the determination of the similarity between concepts representing products in our B2B scenario. A detailed explanation of the tests and results obtained is presented in Chapters 6 and 7.

These WordNet-based semantic measure techniques may be combined with other techniques for ontology mapping. The mapping may be computed by analysing ontology elements in isolation, *i.e.*, without considering their relations

(element-level techniques), or by analysing how entities are related in a structure (structure-level techniques). The combination of both types of techniques is typical of mapping systems. The existing techniques, mapping systems and frameworks are described in the next sections based on the classification presented in [Shvaiko and Euzenat, 2005].

5.3 Element-Level Techniques

The selection of the most appropriated technique depends on how the input information is interpreted. A concept may be a string, a word or a phrase in some natural language. Therefore the techniques are classified as string-based, language-based, constraint-based, based on linguistic resources or based on alignment reuse. These techniques are now briefly explained:

- **String-based techniques** are used to compare concepts as strings (sequence of letters in an alphabet). The principle underlying these techniques is that the more similar the strings are, the more likely they denote the same concepts. This comparison results in a number that indicates the degree of similarity. Some examples of this kind of techniques are edit distance, n-grams, guth, prefix and suffix.

    1. **Edit distance** takes as input two strings and computes the edit distance between them. The output is a similarity degree value between zero and one. The technique consists on determining the number of insertions, deletions and substitutions of characters that are necessary to transform one string into another, normalized by the length of the longest string. The maximum number of operations to transform the string is equal to the size of the longest string. The formula to calculate the edit distance is presented in (5.6):

$$S(S_1, S_2) = 1 - \frac{EditDist(S_1, S_2)}{max(lenght(S_1), length(S_2))} \tag{5.6}$$

    2. **N-grams** takes as input two strings and computes the number of common n-grams between them. N-grams are sequences of n characters, *i.e.*, the algorithm searches for subsets of one string in another. These subsets are called grams and the quantity of characters in each gram is defined by n. Therefore, when a search is performed with three characters it is a trigram.
    3. **Guth** takes as input two strings and compares each position of the characters in both strings, searching for identical characters. This algorithm is useful for name recognition.

4. **Prefix** takes as input two strings and checks if the first string starts with the second one. This technique is efficient in matching similar acronyms.
5. **Suffix** takes as input two strings and checks if the first string ends with the second one.

- **Language-based techniques** are based on Natural Language Processing (NLP) techniques exploiting morphological properties of the input words. Some examples of these techniques are the tokenization, lemmatization and elimination.

  1. **Tokenization** is the operation of splitting up a string of characters into a set of tokens to recognize punctuation, blank characters, digits, *etc.*
  2. **Lemmatization** determines the *lemma* for each token form that occurs in text. The strings are morphologically analysed to find all of their possible basic forms. As a rule, lemmatization entails that verb forms are taken back to the infinite tense, nouns to the singular form, *etc.*
  3. **Elimination** is a procedure where the tokens that are articles, prepositions, conjunctions, *etc.*, are marked to be eliminated.

  All these techniques may be applied for comparing strings. Usually, they are applied before running string-based or lexicon-based techniques in order to improve the final results.

- **Constraint-based techniques** deals with constraints such as data types, cardinality of attributes or value ranges. As data types can be analysed objectively, it is possible to determine how close an attribute data type is to another. The same happens with regard to cardinality.

- **Linguistic resources** uses common knowledge or domain specific *thesauri* to match ontology entities based on linguistic relations (synonyms, hyponyms, *etc.*). Common knowledge, such as WordNet, when combined with some measure techniques as the ones explained previously, may help to match similar concepts. However, domain specific *thesauri* store restrict domain knowledge, which usually is not available as common knowledge.

- **Alignment reuse** explores external resources which contain alignments of previously matched ontologies. If a set of ontologies describe the same application domain there is a high probability that the ontologies to be matched are similar to previously matched ones. This is useful when dealing with large ontologies composed of hundreds or thousands of entities.

5.4 Structure-Level Techniques

A hierarchy may be considered as a graph (nodes related by edges) or as a taxonomy (concepts organised by some relations). The structure-level techniques are graph-based, taxonomy-based, based on repositories of structures and model-based techniques.

- **Graph-based techniques** compare pairs of nodes from two ontologies based on the analysis of their positions within the different graphs. If two nodes from two ontologies are similar, their neighbours might also be similar. Matching graphs is a combinatorial problem that can be computationally expensive. In ontology matching, the problem is encoded as an optimization problem which is resolved with the help of graph matching algorithms. Other approaches compute the similarity between nodes of the different graphs based on the similarity of their children nodes, leaf nodes or based on relations between nodes.

- **Taxonomy-based techniques** use graph algorithms which consider only the *is-a* relation. The neighbours may also be somehow similar. The taxonomy-based result may be calculated using bounded path matching or super(sub)-concept rules. The bounded path matching algorithm takes two paths with links between classes defined by the hierarchical relations, compares the terms and their positions along the paths and identifies the terms. The latter case uses matchers based on rules: if super(sub)-concepts are similar, then the actual concepts are also similar.

- **Repository of structures** store ontologies and their fragments together with the previously calculated similarity coefficients. Before calculating the mapping between new structures, the repository is checked for identical structures which have been already matched.

- **Model-based techniques** implement ontology mappings based on semantic interpretation. Some algorithms use propositional satisfiability and description logics reasoning techniques.

5.5 Mapping Systems

Mapping systems are tools that combine several of the techniques presented before according to specific aims or domain features. We now present the results of three surveys on mapping systems made by [Shvaiko and Euzenat, 2005], [Euzenat *et al.*, 2004] and [Kalfoglou and Schorlemmer, 2003]:

- **Anchor-PROMPT** (extension of PROMPT) [Noy and Musen, 2001] is an ontology merging and alignment tool with sophisticated prompt mechanisms for matching terms. This system takes as input two ontologies and a set of anchor-pairs of related terms (which can be identified with the help of string-based techniques) that can be introduced by the user or by some matcher computing linguistic similarity/dissimilarity algorithm. Then it refines the input relations based on the ontology structures and user feedback. Finally, based on the frequency counts and user feedback, the algorithm determines new matching candidates.

- **Analysis of Requirements: Tool Environment for Multiple Information Systems (Artemis)** [Castano, *et al.*, 2000] was designed as a module of the MOMIS[1] mediator system for creating global views. It performs affinity-based analysis (calculates the name, structural and global affinity coefficients by exploiting a common *thesaurus*) and hierarchical clustering of schema elements. Based on global affinity coefficients, a hierarchical clustering technique categorises classes into groups with different levels of affinity. For each cluster it creates a set of global attributes. Logical correspondence between the attributes of a global class and source schema attributes is determined through a mapping table.

- **Cupid** [Madhavan *et al.*, 2001] implements a hybrid matching algorithm comprising linguistic and structural schema matching techniques and computes similarity coefficients with the assistance of domain specific *thesauri*. The matching algorithm consists of three phases and operates only with tree-structures (to which non-tree cases must be reduced). The first stage computes linguistic similarity coefficients between schema element names (based on morphological normalization, categorization and string-based techniques) and a *thesauri* look-up. The second stage computes structural similarity coefficients weighted by leaves which measure the similarity contexts in which elementary schema occur. The last phase computes weighted similarity coefficients which are higher than a threshold.

- **COmbination of Matching Algorithms** (**COMA**) [Do and Rahm, 2002] implement composite generic matchers. COMA provides an extensible library of matching algorithms, a framework for combining obtained results and a platform for the evaluation of the effectiveness of the different matchers. The matching library is extensible and contains six elementary matchers, five hybrid matchers and one reuse-oriented matcher. While several of these matchers implement string-based techniques as background algorithms and share techniques with Cupid, the reuse-oriented matcher

[1] MOMIS is explained later.

attempts to reuse previously obtained results for entire new schemas or schemas' fragments.

- **Formal Concept Analysis (FCA-merge)** [Stumme and Mäedche, 2001a] is a five step procedure. The first step extracts the instances from the documents provided. The input is made of two ontologies and of the instances belonging to both ontologies. The second step computes two formal contexts, *i.e*, two boolean tables identifying which instance belongs to which ontology concept. The third step merges both contexts by renaming the concepts and adding both contexts. A pruned concept lattice (a structured graph for representing knowledge that can be used by the representation system) is generated using classical formal concept analysis. In the forth step, the lattice is pruned of all the concepts which are not more general than the concepts represented in the original ontologies. The last step consists, with the help of user, in further simplifying the lattice and generating the taxonomy of the new ontology. This final step of deriving the merged ontology from the concept lattice requires human interaction.

- **Glue** [Doan, 2002] employs machine learning techniques to semi-automatically find mappings. It takes two ontologies as input and finds the most similar concepts. Glue first applies statistical analysis to the available data and then generates a similarity matrix based on the probability distribution of the data considered. Finally, it uses constraint relaxation in order to obtain an alignment from the similarity matrix.

- **Ontology-Mapping Method based on Information-Flow Theory (IF-Map)** [Kalfoglou and Schorlemmer, 2003a] aligns two local ontologies by establishing how both ontologies are mapped from a common reference ontology. Whereas it is assumed that the reference ontology is not populated with instances, local ontologies usually are. IF-Map generates possible mappings between the unpopulated reference ontology and a populated local ontology by taking into account how local communities classify instances with respect to their local ontologies.

- **Naive Ontology Mapping (NOM)** [Ehrig and Sure, 2004] is an approach that uses a wide range of features and measure techniques similar to COMA. Some innovations are found in the set of elementary matchers based on rules which are used to exploit codified knowledge such as information about super and sub-concepts, super and sub-properties, *etc.* NOM also includes a set of instance-based techniques.

- **OWL-Lite Aligner (OLA)** [Euzenat and Valtchev, 2003] relies on the classical similarity-based paradigm for entity comparison. First, the OWL ontologies are compiled into graph structures unveiling all relationships between entities. The similarity between nodes from different graphs depends on the category of the node considered and takes into account all the features of that category. Distance-based algorithms convert definitions of distances based on all input structures into a set of equations. These distances are almost linearly aggregated. Finally, the algorithm looks for a matching between the ontologies that minimises the overall distance between them.

- **Quick Ontology Mapping (QOM)** [Ehrig and Staab, 2004] is a successor of the NOM system. The similarity computation is based on a wide range of ontology features and heuristic combinations. QOM avoids the complete pair-wise comparison of trees in favour of a (n incomplete) top-down strategy. The aggregation of single methods is only performed once per candidate mapping and is therefore not critical for the overall efficiency. QOM first iterates to find mappings based on lexical knowledge and then iterates to find mappings based on knowledge structures.

- **Similarity Flooding (SF)** [Melnik *et al.*, 2002] takes two graphs as input and produces as output a mapping between the corresponding nodes of both input graphs. Depending on the matching goal, a subset of the mapping is chosen using a set of filters. The input to the algorithm is first converted into directed labelled graphs. Then, an iterative fixpoint computation is determined. The results show which nodes of the first graph are similar to the nodes of the second graph. The underlying principle used to compute the similarities is that elements of two distinct models are similar when their adjacent elements are similar, *i.e.*, the similarity between two elements propagates to their respective neighbours.

- **Schema-based matching (S-Match)** [Giunchiglia *et al.*, 2004] takes two graph-like structures as input and produces as output a mapping between the nodes of the two graphs that have semantic correspondence. Each node is encoded with a formula in propositional logic. The formula contains the conjunction of all possible senses, within the context of the node, of each word in the label and of all formulas of ancestor nodes. For each pair of nodes, each possible type of semantic relation is checked as a problem of propositional SATisfiability (SAT), using a SAT solver.

5.6 Frameworks

Several ontology development frameworks use mapping as one of its internal components. The tasks related with multiple-ontology management include ontology versioning, ontology merging and ontology mapping. A brief explanation of some of these frameworks is summarised below:

- **Chimaera** [McGuinness *et al.*, 2000] is a merging and diagnostic Web-based browser ontology environment to support users in creating and maintaining distributed ontologies on the Web. It was developed at the Stanford University - KSL. Chimaera major functions are the merging of multiple ontologies and diagnosing individual or multiple ontologies. It supports users in such tasks as loading knowledge bases in differing formats, reorganizing taxonomies, resolving name conflicts, browsing ontologies, editing terms, *etc.*

- **OntoMerge** [Dou *et al.*, 2003] is an online system for ontology merging and automated reasoning. The merging process between two ontologies is computed by taking the union of the axioms defining them and using XML namespaces to avoid name collision. The ontology mapping is performed adding bridging axioms that relate the terms in the first ontology to the terms in second ontology. Inferences may be done in the merged ontology in a demand-driven or data-driven way.

- **Rondo** [Melnik *et al.*, 2003] is an environment for model engineering which provides many primitive units for the manipulation and composition of models. It uses high-level operators to manipulate and to calculate mappings between models. Rondo converts schemas into directed labelled graphs whose nodes are candidate aligned pairs and whose arcs represent shared properties.

- **Ontology MApping FRAmework Toolkit** (**MAFRA**) [Maedche *et al.*, 2002] is a framework for mapping distributed ontologies in the Semantic Web. It is based on the idea that the best approach to complex mapping is achieved through reasoning in a decentralised environment like the Web. MAFRA presents semantic bridges and service-centric approaches. Semantic bridges define the structure of specific mappings and the transformation functions to transfer instances from one ontology to the other. The service-centric approach complements the semantic bridges mechanism by providing the transformation services necessary to perform the mapping transformations.

- **Ontology Based System Enhanced with Relationships for Vocabulary Heterogeneity Resolution** (**OBSERVER**) [Mena *et al.*, 1999] aims to overcome problems caused by heterogeneity between distributed data repositories by using component ontologies and the explicit relationships between these components. It presents an architecture consisting of component nodes from different ontologies and an Inter-ontology Relationship Manager (IRM) responsible for maintaining a mapping between the ontologies at the different component nodes [Bruijn *et al.*, 2004].

- **Mediator envirOnment for Multiple Information Sources** (**MOMIS**) [Bergamaschi *et al.*, 1999] intends to provide the user with a global view of the information coming from heterogeneous information sources. MOMIS implements a semi-automatic methodology in which a global schema (ontology) is obtained from the data sources, *i.e.*, a view over the data sources is associated with each element of the global schema (ontology), relating the element with the data sources.

5.7 Conclusions

As presented in this Chapter, there are different methodologies available for implementing ontology mapping. However, there is no integrated framework capable of performing a fully automatic ontology mapping process. To make matters worst, the majority of the existing ontology mapping systems assume as input two complete ontologies which, in a B2B, is neither practicable nor desirable since ontologies may contain strategic business knowledge.

In order to obtain efficient mappings, it is necessary to implement hybrid approaches. Currently, some mapping systems take into account the advantage of combining several techniques, *e.g.*, ontologies, *thesauri*, plain texts, Web pages, *etc.* However, most of the presented systems do not consider them. Additionally, they do not address problems like the interoperability between distributed knowledge sources and do not apply ontology mapping to the domains such as e-commerce. These are applications where ontology mapping plays an important role and few contributions are available.

The best approach depends on the application features, the available ontology information and the components of the ontologies. Several algorithms are available for determining the similarity between concepts, but no experimental results are available to help selecting the best algorithm for each case. A most interesting challenge would be to develop a framework with several ontology mapping techniques which could be experimented for different domains.

In Chapter 6 will use some of the approaches described in this Chapter to support the interaction between heterogeneous agents using different ontologies.

Chapter 6

Ontology-based Services

In this Chapter we present the agent based ontological services[1] *developed to support the activity of an open B2B MAS community. These services, which are designed to supply the enterprise agents with an efficient business information exchange, include a matching terms service, with several similarity matching algorithms, and a units conversion service. The architecture of the MAS as well as of the individual agents deployed, the roles of the different types of agents implemented, the communication protocols used and the selected development platform are described. The Ontology Interaction Protocol (OIP) implemented to support the interaction between the enterprise agents and the agent based ontological services is explained. Finally, the limitations of the selected development platform regarding the creation, maintenance and update of agents using different ontologies are addressed and a solution is proposed.*

6.1 Introduction

Considering a B2B context where heterogeneous agents (from different origins, modelling different companies, developed by different programmers and using different ontologies) are intended to interact, several interoperability problems are bound to occur. The enterprise modelling agents may use different vocabularies to represent product information, different currencies to represent prices, different measuring units to represent characteristics, *etc.* These are the types

[1] Some of the implementation reported here has been done under the supervision of myself and Professor Eugénio Oliveira by a German student, Daniel Palzer, holding a scholarship in our NIAD&R (LIACC-FEUP laboratory). This work has been reported in [Malucelli *et al.*, 2006], [Malucelli *et al.*, 2005b] and [Palzer, 2005].

of interoperability problems addressed in this work by means of ontology-based services.

FIPA has identified and analysed several interoperability problems that occur in heterogeneous MAS and, as a result, proposed the development of an Ontology Agent (OA)[2] for MAS platforms. Additionally, FIPA suggested that an OA should provide the following functionalities:

- Maintain ontologies by defining, modifying or removing elements.
- Respond to queries about the terms represented in an ontology or regarding the relationship between ontologies.
- Translate expressions between different ontologies or different content languages, possibly as a wrapper to an ontology server.

We claim that a system implementing an OA should at least provide one of these functionalities. We have created an Ontology-based Services Agent (OSAg) that provides the following services:

- Matching Terms Service (MTS).
- Units Conversion Services (UCS).

Although recent research work on ontological discrepancies has been conducted, none of the available proposals tackle all types of discrepancies, as explained in detail in Section 4.3. Some of the problems that occur when finding the similarities between different ontologies are related to the following facts:

- Different ontologies may use different concept names to represent the same meaning and description, *e.g.*, "tyre" and "tire".
- Different ontologies may use the same concept name with different meanings and descriptions, *e.g.*, "wheel" meaning "wheel" and "wheel" meaning "steering wheel".
- Different ontologies may use the same concept name to represent the same meaning, but the description includes different characteristics and relations, making it very hard to detect similarities, *e.g.*, the concept "car" with characteristics "brand", "colour" and "doors" and the same concept "car" with characteristics "manufacturer", "colour" and "number_of_doors".

[2]Ontology Service Specification, XC00086D, 08/10/2001, http://www.fipa.org/specs/fipa00086, April, 2005

Similarity evaluations among ontologies may be achieved if the concepts under analysis have representations that share some components. If two ontologies use at least one common component (attribute, relation, hierarchy, type), then the ontologies may be compared. Usually, the attributes are a good starting point since they capture the details of the concepts.

Our research goal is to provide ontology-based interoperability services through the use of agents. Figure 6.1 shows a global view of our proposal: the application is intended to reinforce agent interoperability by supplying enterprise agents with services specialised in solving interoperability problems. These services are the Matching Terms Service and the Units Conversion Services. In order to match terms (concepts), the MTS applies the similarity measure techniques implemented. After matching the terms, an automatic similarity classification is generated and a basic learning process occurs to improve future negotiation rounds. The MTS operates with full or partial ontologies, which, in this work, are provided by the individual enterprise agents. The UCS are implemented as Web services.

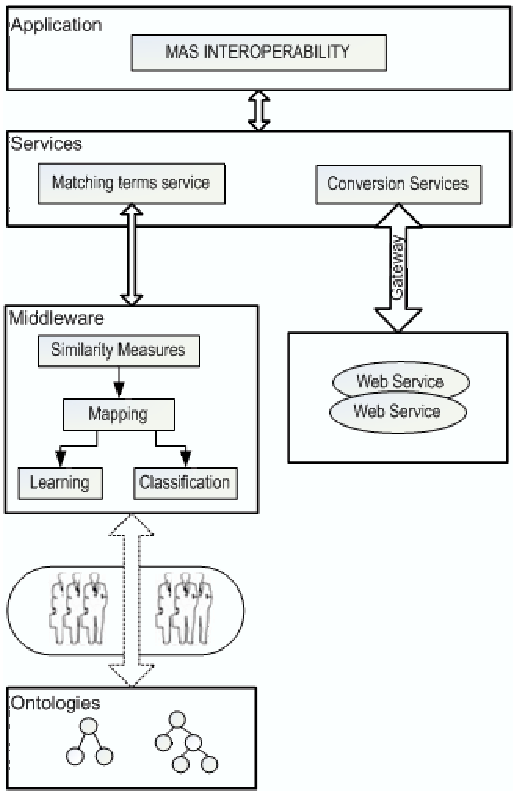


Figure 6.1: Global Vision of an Agent-based Service System

As agents are intended to communicate with other agents, a number of interaction languages, tools and platforms have been developed [He *et al.*, 2003].

In order to select a development setup appropriate to the requirements of the problem domain, it is necessary to assess beforehand the potential as well as the impact of each language, tool and platform [Malucelli *et al.*, 2005a].

Although there are several efforts to develop appropriate platforms, tools and languages to deal with the multi-agent interoperability problem, it seems reasonable to integrate and improve the different approaches in order to explore their full potential.

To address the problem of how to create agents with different ontologies using an automated and integrated approach, we developed a new methodology. Since we chose the JADE platform to develop our test-bed, we were faced with the problem of creating JADE agents with different ontologies. Moreover, we had to combine two communication protocols to implement our negotiation methodology (which is crucial to ensure interoperability): the FIPA Contract Net Interaction Protocol (FIPA-CNP) and the Ontology Interaction Protocol (OIP). The former represents the general scenario of agents trading goods or services proposed by FIPA. The latter is the protocol used to support the resolution of the interoperability problems that arise during the agent interaction [Malucelli *et al.*, 2006].

6.2 Ontology-based Interoperability Problem

Consider the following negotiation examples regarding automobile components. A customer agent needs to buy a "wheel" (a simple artefact consisting of a circular frame with spokes or of a solid disc that can rotate on a shaft or axle) and a supplier agent offers a "wheel" (a handwheel that is used for steering). These two components, which belong to the automobile domain, have identical syntax but are semantically different. However, the two agents, unaware of the misunderstanding, are likely to engage in negotiation. A second possible scenario can be played by a customer agent that requires a "tyre" (a thick rubber ring often filled with air, which is fitted around the outer edge of the wheel of a vehicle). A supplier agent that sells "tire" (a thick rubber ring, often filled with air, which is fitted around the outer edge of the wheel of a vehicle) does not make any offer since it is unaware of the existing semantic correspondence. In this case, both terms belong to the automobile domain, but they are syntactically different and semantically equivalent. In the former case, agents will waste time negotiating under different products and, in the latter, when the negotiation could be fruitful, they fail to understand each other.

Our application domain describes the scenario of an automobile assembling domain where a customer enterprise engages in a negotiation process to buy several components from different supplier enterprises to assemble an automobile. For each component there may be several potential suppliers. Even when adhering

to terminology standards used by the automobile industry and with suppliers providing offers in a syntactically unified framework, the same term may have different meanings or the same meaning may be associated with different terms and have different representations [Malucelli and Oliveira, 2004b].

In order to include as many suppliers as possible in each negotiation round, it is necessary to ensure that each possible supplier correctly understands the customer requirements. The relevant agents, *i.e.*, the potential business partners, have to be able to participate in the negotiation.

Figure 6.2 (a) and (b) shows two UML diagrams holding partial views of the ontologies of a customer agent and supplier agent. Figure 6.2 (a) depicts a partial view of the customer agent ontology (Automobile Assembling Ontology), while Figure 6.2 (b) represents a view of the supplier agent ontology (Car Assembling Ontology). Both views are composed of a set of concepts. Each concept has a description in natural language, relationships with other concepts and a set of characteristics called attributes. Each attribute has a data type and is assigned a value.

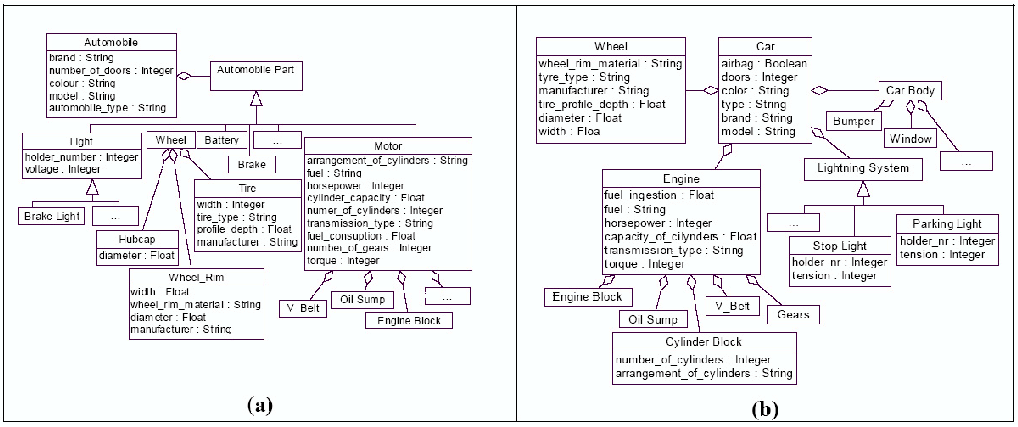


Figure 6.2: Partial View of the Ontologies of the Customer and Supplier Agents

Through these examples we may observe some differences that will cause interoperability problems during the negotiation process. For example, in the "Automobile Assembling Ontology" there is a concept (referring to a product) named "Motor" and in the "Car Assembling Ontology" there is a correspondent concept called "Engine".

In our scenario, the customer agent and the supplier agent have the same objective: they want to trade products in the same application domain and still use their own private ontologies. Due to this common objective, we provide an e-commerce ontology that defines an e-commerce vocabulary just for the trading, *i.e.*, a negotiation protocol. This vocabulary contains terms which are used during the negotiation process. Thus, we ensure that all agents will uniformly interpret

the negotiation messages exchanged. This does not imply identifying correctly the specific content of the messages, *i.e.*, the requested products/items, since each agent interprets this information based on its own private domain ontology.

6.3 MAS Architecture

The main advantage of the software agent's paradigm is that communities of agents are much more powerful than any individual agent. The MAS paradigm relies on data exchange between agents and, as a result, requires a high level of interoperability.

In our MAS approach we will have at least four types of agents: the Customer Enterprise Agent (CEAg), the Supplier Enterprise Agent (SEAg), the Ontology-based Services Agent (OSAg) and the Facilitator Agent (FAg).

The facilitator agent and the enterprise agents - suppliers and customers - interact with the objective of providing or obtaining goods/products/services while keeping their own preferences and objectives. The OSAg supports the negotiation process between customer and supplier agents and helps in the calculation of prices (that may be necessary when dealing with different currency units) and in the conversion between different measurement units.

The agents are deliberative, autonomous, intelligent entities that use different ontologies and apply goal driven reasoning to achieve their objectives.

Agents have an inference engine which derives the recommendations from the knowledge base (ontologies) and problem-specific data in instances (see Figure 6.3). Each agent has its own private Automobile Assembling Domain Ontology (AADO) and shares the E-Commerce Domain Ontology (E-ComDO) with the remaining system agents. A user interface allows the interaction between the user and the agents.

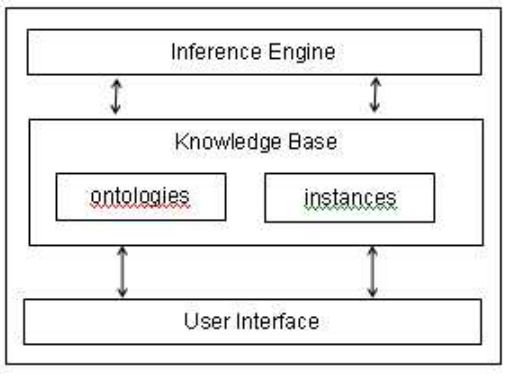


Figure 6.3: Agent Architecture

Figure 6.4 shows an overview of our MAS architecture. Each enterprise agent (Supplier or Customer) has its own architecture and functionalities.

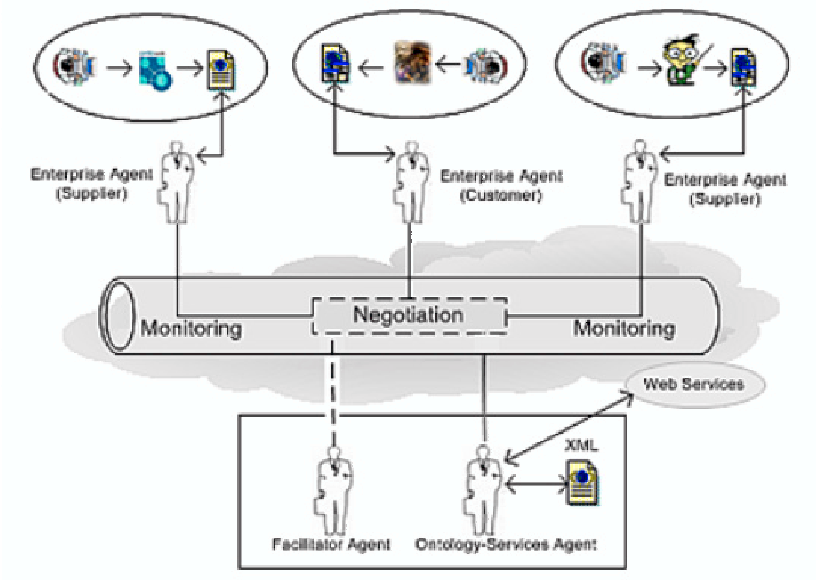


Figure 6.4: System Architecture

Our MAS is composed of several enterprise agents, *i.e.*, multiple instances of Customer (CEAg) and Supplier Enterprise (SEAg) agents, a Facilitator (FAg) agent and the Ontology-based Services agent (OSAg). These four types of agents play the following roles:

- The **CEAg** represents enterprises interested in buying components to build a final product. Several suppliers in the world may have these components at different prices and conditions. The user representing the registered customer enterprise inserts the information the agent requires in order to negotiate a product (item). This includes the name of the item, the quantity and the currency. The currency chosen is the currency in which the CEAg wishes to receive proposals, *e.g.*, Dollar (USD).

- The **SEAg** represents enterprises interested in providing some kind of product/service/good. Whenever a product is needed, the FAg conveys this announcement to all registered interested SEAg. Once alerted by the FAg, the SEAg makes an offer if the announced item is currently in stock and if it understands what the customer is asking for. If the SEAg wants to make a proposal but the product is priced in a currency different from the one used in its ontology, *e.g.*, Euro (EUR), the SEAg requests the assistance of the OSAg, *i.e.*, contacts the UCS. Similarly, if the characteristics of the product are represented in different measurement units, *e.g.*, inches

versus centimeters, the SEAg contacts the UCS. At least one SEAg has to be present in the platform so that communication can be established.

- The **FAg** is the entity that matches the prospective partner agents and supports the negotiation process. It interacts with the Directory Facilitator (DF) of the FIPA-compliant JADE[3] platform which provides a yellow-pages service. Each time a new SEAg registers in the platform, the DF informs the FAg.

- The **OSAg** is responsible for providing the services that support the negotiation among enterprise agents. This approach is compliant with the FIPA proposal of creating a specialised OA for open MAS platforms. The idea is to support the interoperability between agents with different individual ontologies by means of a dedicated agent. Our implementation of the ontology agent is named OSAg and follows the FIPA recommendation. It is responsible for providing services to the enterprise agents that help solving the interoperability problems. Its task consists of applying different methodologies to detect lexical and semantic similarities between two concepts as well as providing units conversion services. The OSAg is an autonomous server-side agent that does not require any direct user interaction. The user interface only controls the tasks performed by the OSAg.

The development of the enterprise agents started with the creation of their private ontologies. Ontology creation for the automobile assembling domain involved vast literature search and knowledge elicitation with experts. After careful consideration and test of several different ontology building tools, we selected the ones that seemed more appropriated to our specific problem. First we modelled the ontologies in UML and then, with the aid of the Protégé ontology development tool, we built and store the ontologies in OWL files. However, the use of these tools is optional and the ontologies may be created by means of different tools and be stored using other knowledge structures.

The multi-agent development environment we chose was JADE. JADE is a popular framework for setting up MAS that includes the implementations of the FIPA library of behaviours and interaction protocols. Furthermore, JADE implements the basic FIPA specifications which include FIPA-ACL, content languages, encoding schemes, ontology and transport protocols. Based on these specifications, FIPA agents exist, operate and communicate.

As presented in Section 3.7, Protégé may store ontologies in various formats, *e.g.*, RDF(S), XML, OWL, *etc.* Since JADE agents represent ontologies as Java objects, it is necessary to map the different ontologies created with Protégé into corresponding Java objects.

[3]JADE - *Java Agent DEvelopment Framework*, http://jade.tilab.com, June, 2002

6.4 Mapping from Multiple Ontologies to JADE

JADE proposes the use of the Content Reference Model (CRM) [Caire and Cabanillas, 2004], which is a classification of all possible elements that occur in the discourse domain, to support ontologies.

Figure 6.5 presents an UML diagram of the CRM. The important types in this case are "Predicate", "Concept" and "AgentAction", since these are the types a JADE ontology uses.

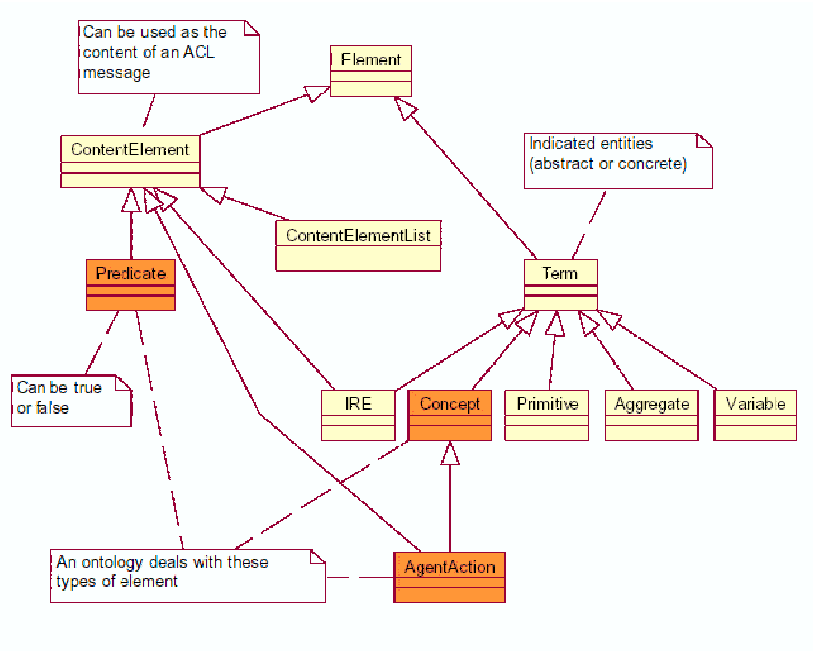


Figure 6.5: Content Reference Model

In detail, the types of elements an ontology deals with are defined as follows:

- **Concepts** are expressions that indicate entities that "exist" and that agents talk and reason about.
- **Predicates** are expressions that say something about the status of the world and usually evaluate to true or false.
- **AgentActions** are expressions that indicate something that can be executed by some agent.

As already mentioned, JADE agents require ontological information to be represented in terms of Java objects, while Protégé stores ontologies in various formats such as OWL, *etc.* If a Protégé project is intended to be translated into

a set of Java objects, the CRM has to be considered *a priori*, *i.e.*, during the ontology definition stage. It plays an important role since it determines certain constraints. The scheme for the types "Concept", "Predicate" and "AgentAction" has to be included when creating an ontology in Protégé.

Protégé allows importing an ontology either by merging it with an existing ontology or by including it as external terms of another ontology. The former means copying all definitions in the current ontology so that they can be edited. The latter defines the terms in another ontology and thus permits editing. This feature plays an important role in our case since the user has to include a project called `SimpleJADEAbstractOntology` in its current project. This project is provided along with the BeanGenerator plug-in. Thereby, "Concept", "Predicate" and "AgentAction" types are included in the class taxonomy tree and the new classes have to be defined as subclasses of these classes. In detail, the concepts (*e.g.*, "Automobile_Part" or "Brake") are created by specialising the class "Concept" , the agent actions (*e.g.*, "Sell" or "Buy") by specialising "AgentAction", the agents (*e.g.*, "Seller" or "Buyer") by specialising the class AID and the predicates (*e.g.*, "Owns", "IsPurchasable", *etc.*) by specialising the class "Predicate".

6.4.1 From Protégé Ontologies to JADE Representation

When the CRM is considered, the ontology definition process in Protégé is straightforward, *i.e.*, the classes representing an ontology for object-oriented programming languages can be built and used directly. The BeanGenerator plug-in for Protégé can be used to perform this task. It automatically generates Java source files directly from the Protégé model. This is quite convenient, otherwise ontological Java classes would have to be written "by hand", making unpractical the use of an ontology editor like Protégé.

The BeanGenerator associates the class structure defined in Protégé with java sources. So, when using BeanGenerator, the types Concept, AgentAction, and Predicate are interfaces. The sum of all created objects that implement the interfaces `jade.content.Concept`, `jade.content.Predicate` and `jade.content.AgentAction` represents the ontology. Furthermore, a common class defining the vocabulary of all classes, registering the "Predicates", "Concepts" and "AgentActions", and storing name mappings is created.

All classes generated by the BeanGenerator plug-in consist of attributes specified in Protégé with the corresponding set-/get-methods. In order to be used by JADE agents, the generated ontology files have to be imported into the agent project and the ontology has to be registered inside the agent's code.

While JADE agents represent internally the ontological information in terms of Java objects, when communicating and for simplicity and compatibility rea-

sons, the Java objects are serialised. Thus, the content of an FIPA-ACL message is made of a string (sequence of characters) or a sequence of bytes.

6.4.2 Ontologies Following the CRM

JADE ontologies that follow the CRM allow inheritance, support the combination of ontologies and, thus, facilitate code reuse. Both supplier and customer enterprises want to trade products and operate in the B2B e-commerce domain. In order to bear this commonality in mind, two types of ontologies were created: a generic ontology for the e-commerce domain (E-Commerce Ontology) and the private automobile assembling domain ontologies (*e.g.*, the Car Assembling Ontology, the Automobile Assembling Ontology, *etc.*) of the implemented enterprise agents.

All ontologies defined are CRM-compliant and are stored as Java objects in JADE. Figure 6.6 shows a scenario where both enterprise agents share the automobile assembly domain ontology (Car Assembling Ontology) together with the E-Commerce Ontology. The figure also depicts the relation between the created ontologies. Every agent that wishes to negotiate must be acquainted with the ontological classes ("CLS") whose objects are used to fill the content of the FIPA-ACL messages. Both the CEAg and the SEAg are able to interpret unambiguously the exchanged e-commerce messages. To indicate this, they register the name of the ontology they use. Both the e-commerce ontology and the car assembling ontologies are recognised as Java classes. The FIPA-ACL communication uses the E-Commerce Ontology.

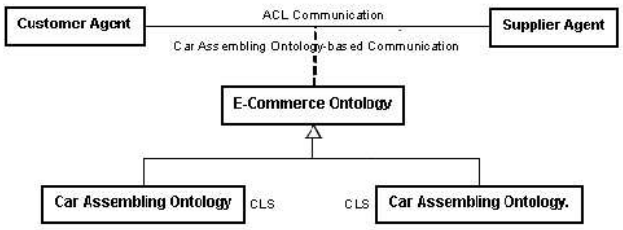


Figure 6.6: Communication based on the Car Assembling Ontology

The shared e-commerce ontology specification includes domain-independent business terms and domain-specific vocabulary.

6.4.2.1 E-Commerce Ontology

The E-Commerce Ontology contains terms which are used during the negotiation process, ensuring a meaningful communication since all agents will uniformly in-

terpret the purpose of the messages exchanged. Moreover, the contracts resulting from successful negotiations are based on this ontology.

The E-Commerce Ontology defines "Concepts" (*e.g.*, "Price"), "AgentActions" (*e.g.*, "Buy") and "Predicates" (*e.g.*, "IsPurchasable") which describe the basic e-commerce concepts and relationships. The E-Commerce Ontology may be applied to any e-commerce domain since it is not concerned with the identification of the requested products/items. This task is performed by the individual agents based on their private domain ontologies. Figure 6.7 shows the E-Commerce Ontology. In this ontology, "AgentActions" consist of "Buy" and "Sell", the only "Concept" is "Price" and "Predicates" include "Costs", "IsPurchasable" and "NotInStock".

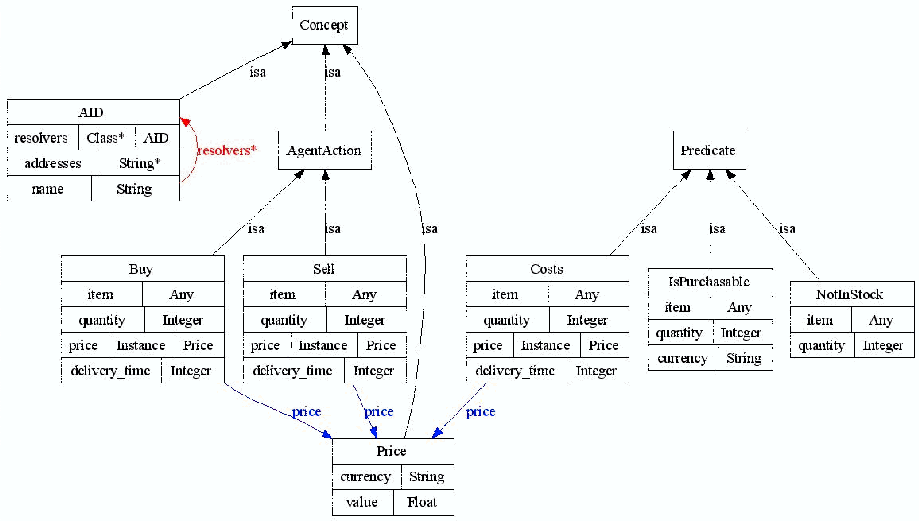


Figure 6.7: E-Commerce Ontology

The intention of the negotiation messages exchanged between agents can be understood with this ontology. The concrete items and properties under negotiation require additional knowledge found in the domain ontologies of the individual agents.

6.4.2.2 Car Assembling Ontology

A domain ontology models the knowledge regarding the elements in a certain domain of discourse. In this case, it is a conceptualisation of the components of an automobile. Figure 6.8 shows a part of the car domain ontology according to the CRM.

Compared with the E-Commerce Ontology, this domain ontology defines only concepts, *i.e.*, subclasses of "Concept". These concepts are the entities that agents

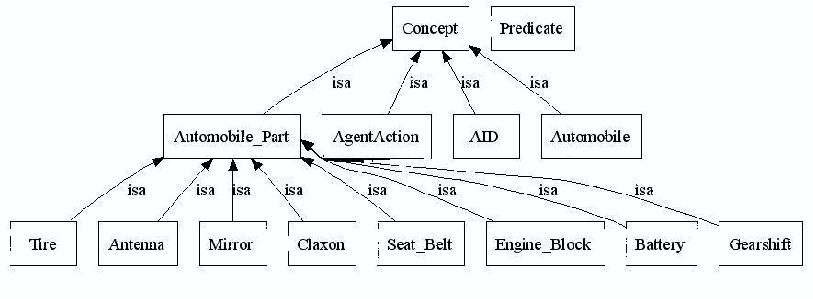


Figure 6.8: Partial View of the Car Assembling Ontology

communicate and negotiate about. "Predicates" and "AgentActions" are not necessary to represent the components of an automobile.

All agents in our platform communicate by exchanging FIPA-ACL messages. A FIPA-ACL message contains a set of one or more message parameters, *e.g.*, type of communicative act, participants in communication, content of message, description of content and control of conversation[4]. The communication follows the FIPA Contract Net Interaction Protocol (FIPA-CNP). The type of communicative act, defined by the performative parameter, is a required parameter of all FIPA-ACL messages. The participants consist of sender and receiver.

Figure 6.9 shows an example of a message that is sent to all known Supplier Enterprise Agents with the performative Call-For-Proposals (CFP).

```
(CFP
   :receiver (set
      (agent-identifier
         :name supplier@sutton:1099/JADE
         :addresses (sequence http://sutton:7778/acc)))
   :content (isPurchasable (
                        :item Motor
                        :quantity 1
                        :currency EUR))
   :language fipa-sl
   :ontology carAssembling
   :protocol fipa-contract-net)
```

Figure 6.9: FIPA-ACL Message Using the Car Assembling Ontology

The content parameter contains the predicate "IsPurchasable" which is defined in the E-Commerce Ontology and denotes that the CEAg seeks proposals for an item named "Motor". This term originates from the CEAg's private ontology. The ontology parameter has the value **carAssembling**. Even though

[4]FIPA ACL Message Structure Specification, SC00061G, 12/03/2002. http://www.fipa.org/specs/fipa00061/SC00061G.html, Abril, 2005

the predicate "IsPurchasable" is defined in the E-Commerce Ontology, it can be used due to inheritance in the content parameter of the message. The predicate "IsPurchasable" requires values for the parameters item, quantity and currency.

6.4.2.3 Using Different Ontologies

As already explained, when the agents in a MAS share a common ontology, it is possible to use JADE's standard ontology approach which consists of including the common ontology components in the source code of all registered agents. However, in an open MAS, where heterogeneous agents can join and leave the platform freely, it is reasonable to preview the use of several ontologies. Agents operating in the same application domain may use different ontologies and, as a result, hard-coding a shared ontology directly into the agents' code is not a good solution.

Since JADE's does not support the use of different ontologies, it is not intended for the development of open MAS. This limitation was detected during the implementation stage.

Consider Figure 6.9 where Agent_A sends a message to Agent_B asking for some automobile component. In this scenario, since both agents use the same ontologies (E-Commerce Ontology and Car Assembling Ontology), they have access to the same ontological Java classes. Thus, the receiver Agent_B fully understands the incoming FIPA-ACL message. Since Agent_B knows the class "Automobile_Part", it detects that the incoming message is a predicate typed "IsPurchsable" and that the item under negotiation is an object of type "Automobile_Part". However, it still is unaware of the meaning of the item.

Whenever agents use the same ontology, they will definitely find out the type of the object since the casting of the object will succeed. Its characteristics will then be accessible via the specific set-/get-methods of the object. However, an agent will not find the type of object if it does not match with any of the objects it knows. Agent_B just knows that Agent_A asked for an "Automobile_Part", but this information is vague. In the end, neither the name of the item nor any of its characteristics are known by Agent_B.

Whilst agents using the same domain ontology can always identify the type of the objects requested, agents dealing with different ontologies face additional difficulties. In the latter scenario, there is no guarantee whatsoever that Agent_B is aware of class "Automobile_Part". Suppose Agent_B knew class "Part_of_Automobile" instead of "Automobile_Part", then it would be unable to interpret the content of the incoming message. Again, the required casting of objects poses a problem. The same problem arises if, *e.g.*, the predicate "IsPurchasable" is not defined. Even the OSAg would not be able to help since it would

have to deal with unknown objects. In a nutshell, ontologies have to be identical and available to all negotiating agents if the agent development fully follows JADE's ontology support. The solution we propose in this thesis combines the use of the CRM with different ontologies stored in OWL format.

The final implementation of the platform considers two possibilities: (*i*) the case where supplier and customer agents use the same hard-coded ontology (JADE's ontology support); and (*ii*) the case where the user selects different domain ontologies for customer and supplier agents.

6.4.2.4 Combining the CRM and OWL

A specific domain ontology can be used just by one agent or by a group of agents. However, when agents are negotiating, different domain ontologies will co-exist. Whenever agents use different ontologies - although designed to describe the same domain of discourse - there are no classes that can be shared. Furthermore, a merging between two or more ontologies is inadequate in a competitive context since it requires enterprises to reveal their ontologies to third parties.

To solve this problem, we use a different approach to access the information contained in the private ontology of an agent. The implementation partially abstains from the ontological classes and chooses another knowledge format to provide the required information. We overcome the JADE's inability to support agents using different ontologies by storing the domain ontologies in OWL and processing the OWL files with the JENA interface model. Figure 6.10 illustrates this scenario.

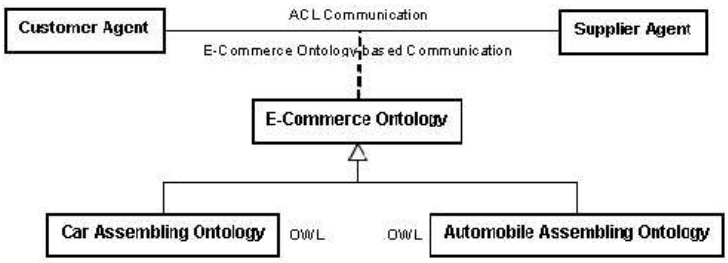


Figure 6.10: E-Commerce Ontology-based Communication

While both enterprise agents share the E-Commerce Ontology, they explore their own private automobile assembling domain ontology: the CEAg uses the Car Assembling Ontology and the SEAg uses the Automobile Assembling Ontology. The E-Commerce Ontology is a common ontology and, consequently, is represented using the JADE ontology support, *i.e.*, it is a set of Java classes. The domain ontologies, that represent the private knowledge of the enterprises, are

stored in OWL files and are processed according to the JENA interface model. JENA extracts the ontological information from the OWL files and implements a transparent mapping mechanism from ontologies to agents.

In this scenario, the agents expect, on one hand, to know exactly the purpose of the messages exchanged since they all share the E-Commerce Ontology and, on the other hand, to experience semantic interoperability problems because of the different private domain ontologies. To solve these interoperability problems we propose the use of ontology-based services.

All agents willing to negotiate must register the E-Commerce Ontology. As a result, they implicitly agree on using the vocabulary defined in the E-Commerce Ontology and, consequently, are able to interpret the negotiation messages unambiguously.

Figure 6.11 shows an example of a message sent to all known Supplier Enterprise Agents with the performative CFP (Call-For-Proposals). The content parameter contains the predicate "IsPurchasable" which is defined in the E-Commerce Ontology and denotes that the Customer Enterprise Agent seeks proposals for an item named "Motor". This term originates from the customer's private ontology.

```
CFP
  :sender
      (agent-identifier
       :name customer@sutton:1099/JADE
       :addresses (sequence http://sutton:7778/acc))
  :receiver
      (set
       (agent-identifier
         :name cust1@sutton:1099/JADE
         :addresses (sequence http://sutton:7778/acc)))
  :content (IsPurchasable (
                        item: Motor
                        quantity: 1
                        currency: EUR))
  :language fipa-sl
  :ontology e-commerce
  :protocol fipa-contract-net)
```

Figure 6.11: FIPA-ACL Message Using the E-Commerce Ontology

The message looks very similar to the example in Figure 6.9. Only the value of the parameter ontology is different. In this case, the E-Commerce Ontology allows the agent to interpret correctly the purpose of the message. When the message arrives, the incoming message object, which is called CFP, is read and the predicate "IsPurchasable" is extracted. The name of the requested product is assigned to the parameter item. In order to find out if this item exists, the receiver agent has to access its OWL ontology.

With this approach, even if the receiver agent, *i.e.*, the supplier, does not understand which product is being requested, it knows its name. This information,

which is in a human-readable string representation, is then passed to the OSAg for "translation".

6.5 Negotiation Protocols

The implementation of our negotiation process combines the FIPA Contract Net Protocol (FIPA-CNP) with an additional protocol called Ontology Interaction Protocol (OIP), as presented in Figure 6.12. The former represents the general scenario of agents trading goods or services proposed by FIPA. As other interaction protocols, it structures complex tasks as aggregations of simpler ones. The latter implements the message flow necessary for solving the interoperability problems, *i.e.*, it supports the interaction of customer and supplier agents with the OSAg.

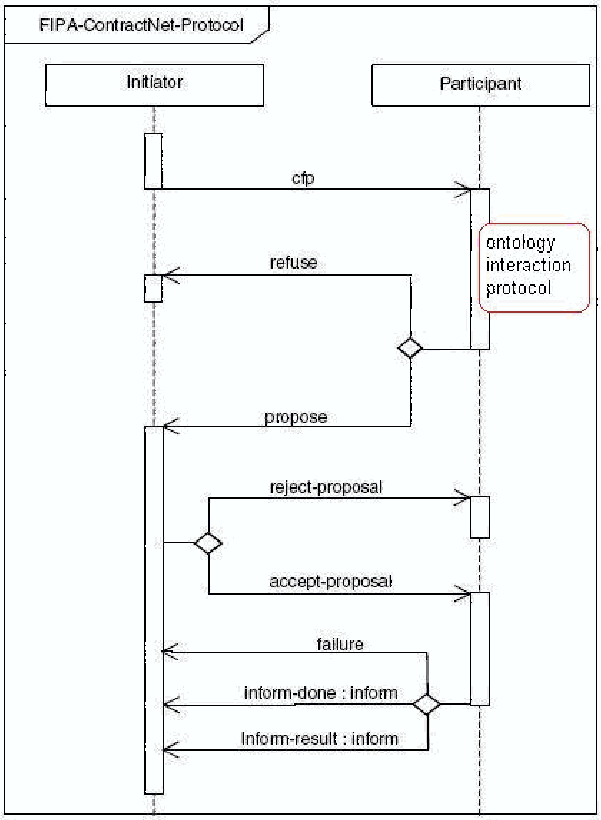


Figure 6.12: FIPA Contract Net Interaction Protocol and OIP

In our scenario, the CEAg plays the role of the initiator, while the SEAg is the respondent. The initiator agent wishes to perform some task by optimising

its cost/benefit function. This characteristic is commonly expressed as the price, but can also be the soonest time to completion, fair distribution of tasks, *etc*[5].

For a given task, the participants may respond with a proposal or a refusal to negotiate. Negotiations then continue only with the participants that offered proposals. The initiator selects, among all proposals received, the best offer based on its own cost-benefit criteria and replies to all suppliers that made offers with an acceptance or rejection message, depending on the case. In the former case, once the SEAg has completed the task, it sends a message to the CEAg in the form of an INFORM-DONE performative or, using a more explanatory version, in the form of an INFORM-RESULT performative. However, if the participant fails to complete the task, a FAILURE message is sent.

The sequence diagram (Figure 6.12) of the FIPA Contract Net Protocol shows how contracts in general are accomplished. JADE provides classes that implement the FIPA Contract Net Protocol.

An agent answering a CFP performative should answer with a proposal specifying its conditions to perform the required action. The responder conditions should be compatible with the conditions originally contained in the CFP. Suppose the CFP calls for a set of tires and specifies that the currency unit must be Euro. A compatible proposal would be "500 Euro for a set of 4 Michelin tires". An incompatible proposal would be "400 Rand for a set of 4 Michelin tires".

The sequence diagram in Figure 6.13 represents the implemented ontology interaction protocol (OIP), which intends to find correspondent concepts in two different ontologies with the assistance of the set of services provided by the OSAg. The CEAg and the SEAg will be the initiators of all interactions with the OSAg.

Three different situations may occur when a supplier receives a CFP message: (*i*) the agent refuses the CFP, (*ii*) the agent accepts the CFP or (*iii*) the agent contacts the OSAg because it is unable to fully understand the CFP. In the first case, the respondent answers with a REFUSE performative. In the second case, the agent responds with a PROPOSE performative stating its conditions to perform the specified action. The responder's conditions should be compatible with the conditions originally contained in the CFP. In the last case, the agent, upon detection of an interoperability problem, follows the sequence diagram presented in Figure 6.13:

[5]FIPA Contract Net Interaction Protocol Specification, SC00029H, 12/03/2002, http://www.fipa.org/specs/fipa00029/SC00029H.html, April, 2005

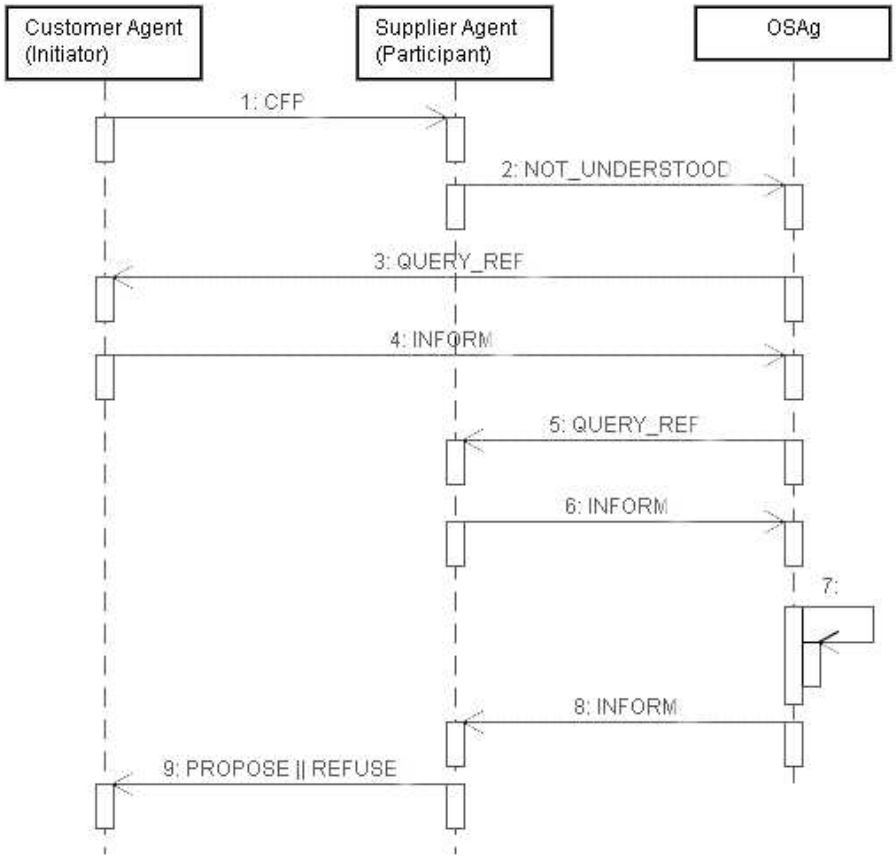


Figure 6.13: Ontology Interaction Protocol

- After having received a CFP (1) as part of the FIPA Contract Net Protocol and not being able to interpret the requested item, the SEAg sends a message with the performative NOT_UNDERSTOOD to the OSAg (2), identifying both the sender of the CFP (CEAg) and the name of the unknown item. The OSAg generates and sends a QUERY_REF message to the CEAg (3) inquiring about the unknown data item.

- The CEAg analyses the request and answers with an INFORM performative containing the attributes, relations, attribute data types, price and the description of the concept, *i.e.*, all the information about the required data item (4). The price is taken from its pricelist.

- Stages (5) and (6) refer to the pre-selection process where correspondent candidate concepts are searched within the SEAg private ontology with the help of the OSAg. After having received the requested information from the CEAg (4), the OSAg knows the price of the product under negotiation and sends it to the SEAg (5). The SEAg applies the pre-selection process,

which is based on the price attribute, to reduce the set of potential matching concepts. This step is absolutely essential in large ontologies, otherwise the number of pairs of concepts to compare would be too high. If the SEAg uses a currency unit different from the one specified in the CFP, it requires the assistance of the OSAg's currency conversion service. The SEAg selects the products whose price is in a range between 75% and 125% of the received value. Once the pre-selection is finished, the SEAg sends an INFORM performative holding a list of names, documentation and attributes of potential correspondent concepts (6).

- After receiving all the information about the item under negotiation and the list of possible corresponding items, the OSAg applies the implemented similarity detection methods (7). These ontology mapping methodologies aim at detecting syntactic and semantic similarities between terms. Every candidate term is compared with the unknown term. The similarity detection methodologies applied are described in the next Section.

- In step (8), the OSAg informs the SEAg (INFORM performative) of the outcome of the process, *i.e.*, sends the name and classification of the best match or informs that it was unable to find any corresponding item. If necessary, the CEAg requests the UCS.

- The SEAg is then able to respond to the CEAg CFP (9), either with a PROPOSE or with a REFUSE performative according to the FIPA-CNP.

6.6 Services for Negotiation

The OSAg provides services for enabling the interoperability between agents that represent distinct organisations, use different ontologies and adopt diverse currency or measurement units. These services support the negotiation of specific items, leading to appropriate conversations and making agreements possible.

The similarity between terms is detected through three different approaches: (*i*) similarity matching between attributes and relations; (*ii*) similarity matching between descriptions; and (*iii*) similarity matching between concepts.

6.6.1 Matching Terms Service

Ontology mapping is the process of finding correspondence or similarity between concepts from two ontologies. If two concepts correspond, they are likely to mean the same thing or closely related things [Dou *et al.*, 2003]. Our approach aims at finding correspondences between concepts based on their names, characteristics, relations and descriptions.

In a competitive context, enterprises are not willing to share strategic corporate knowledge. Therefore, each enterprise agent has a private domain ontology which reflects its preferences and role in the domain. Each entity, which usually has a partial view of the overall knowledge domain, will only represent the components it identifies and is free to describe these components with the level of detail it chooses. However, we believe that experts, even when using distinct characteristics, structures and relations to describing two different concepts that belong to the same domain, will always use a set of common relevant characteristics, *e.g.*, material, width, height, *etc.* Moreover, when defining the same product, it is not usual to find radically different descriptions since some technical and common terms are likely to be used.

Based on these ideas we start the mapping process by applying lexical similarity measure techniques. Since the lexical comparison may not deliver a fully reliable result, we combine it with a WordNet-based technique that discovers semantic similarities between concepts. Whenever a mapping between two concepts is established and validated, the OSAg memorises the result avoiding the future triggering of the mapping process for the same pair of concepts.

6.6.1.1 Similarity Matching between Attributes and Relations

The similarity matching process implemented is based on the following assumption: if two different ontologies represent the same domain, there is a high probability that the concepts described have similar syntax and share similar attributes.

We conducted a series of tests to determine the most appropriate lexical similarity measurement methodology for our specific case. After applying several string-based similarity methodologies [Levenshtein, 1966], [Guth, 1976], we concluded that the n-grams approach [Damashek, 1995] was the most suitable to our data features. The n-grams measurements have been used as alternatives to word-based retrieval in a number of systems [Natrajan *et al.*, 1997]. Furthermore, n-grams can be used to distinguish between documents in different languages in multi-lingual collections and to gauge topical similarity between documents in the same language. N-grams are consecutive overlapping sequences of n characters formed from an input stream. The algorithm looks for subsets of one string in another. These subsets are called grams and the number of characters in each gram can be defined. After some experiments we chose to use a 3-gram comparison.

To achieve better results, we group the attributes according to their data types (*string*, *float*, *integer*, *boolean*) and consider the relation *has-part*. Since the data types adopted to hold the attributes of corresponding terms tend to be identical, even when different syntax is used to describe the attributes themselves, this classification ensures that the comparisons performed between attributes make

sense. Applying the n-grams technique without grouping the attributes by data type could lead to undesirable results.

The example presented previously in Figure 6.2 depicted two enterprise agents (a CEAg and a SEAg) with different domain ontologies: while the CEAg represents the knowledge domain through the Automobile Assembling Ontology (6.2 (a)), the SEAg uses the Car Assembling Ontology (6.2 (b)). In this scenario, if the CEAg requests a "Motor", the SEAg, which ignores concept "Motor", requests the assistance of the OSAg.

Once the OSAg holds all the necessary information provided by both CEAg and SEAg, it first groups the attributes by data type and then uses the n-grams approach to calculate the similarity between attributes with identical data type, as presented in Figure 6.14 and 6.15. Next, the OSAg applies the n-grams comparison between the *has-part* relations associated with "Motor" with those of the candidate concepts found in the ontology of the supplier agent. The classes pointed by the relations *has-part* are also considered in the comparison. As a result, the comparison between the attributes and relations of "Motor" and "Engine" (Figure 6.14) is decomposed in the comparisons between their *string*, *integer* and *float* attributes and *has-part* relations.

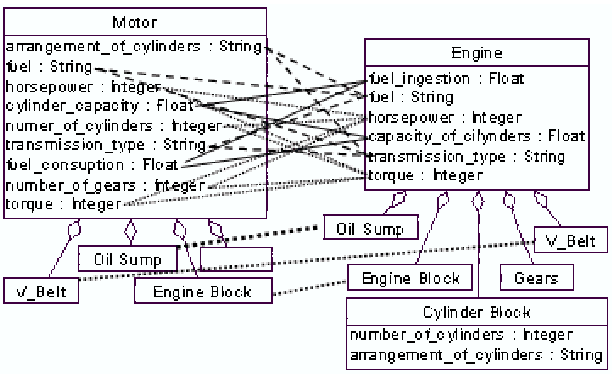


Figure 6.14: Ontology Mapping (grouped by type) between "Motor" and "Engine"

The comparison between "Motor" and "Wheel" (Figure 6.15) results in comparing their *string* and *float* attributes because the concept "Wheel" contains neither *integer* attributes nor any relations *has-part*.

Since the n-grams algorithm delivers a value for each pair of words and not for a collection of words, the individual results concerning each data type (*string*, *float*, *integer*, *boolean*) as well as the *has-part* relation have to be combined.

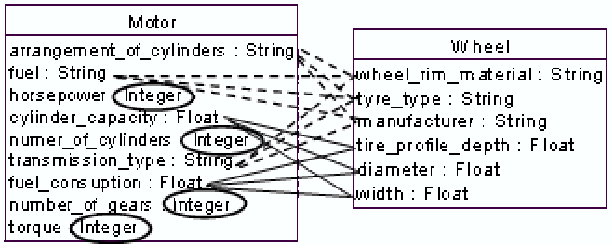


Figure 6.15: Ontology Mapping (grouped by type) between "Motor" and "Wheel"

The final n-grams result for the *has-part* relation and per data type $r_{n-grams}$ is calculated using formula (6.1):

$$r_{n-grams} = \frac{\sum_{i=1}^{n} max_i}{n} \tag{6.1}$$

where max_i is the maximum value of all comparison results that exist for each attribute data type (marked bold in Table 6.1) as well as for the *has-part* relation, and n indicates the number of max_i. Table 6.1 shows the n-gram results between the attributes of "Motor" and "Engine".

Type	Attributes of Motor	Attributes of Engine	N-grams
integer	Horsepower	horsepower	**1.0**
	Horsepower	nr_of_cylinders	0.0
	Horsepower	Torque	0.0
	number_of_cylinders	horsepower	0.05
	number_of_cylinders	nr_of_cylinders	**0.7**
	number_of_cylinders	torque	0.0
	number_of_gears	horsepower	0.0625
	number_of_gears	nr_of_cylinders	**0.3125**
	number_of_gears	torque	0.0
	Torque	horsepower	0.0
	Torque	nr_of_cylinders	0.0
	Torque	torque	**1.0**
	Total ($r_{n-grams}$)**: 3.0125 / 4 = 0.7531**		
string	arrangement_of_cylinders	transmission_type	**0.04**
	arrangement_of_cylinders	fuel	0.0
	Fuel	transmission_type	0.0
	Fuel	fuel	**1.0**
	transmission_type	transmission_type	**1.0**
	transmission_type	fuel	0.0
	Total ($r_{n-grams}$)**: 2.04/ 3= 0.68**		

Type	Attributes of Motor	Attributes of Engine	N-grams
float	fuel_consumption	capacity_of_cylinders	0.0
	fuel_consumption	fuel_ingestion	**0.4705**
	cylinder_capacity	capacity_of_cylinders	**0.7272**
	cylinder_capacity	fuel_ingestion	0.0
	Total ($r_{n-grams}$)**: 1.1977 / 2 = 0.5989**		
has-part	oil_sump	engine_block	0.0
	oil_sump	cylinder_block	0.0
	oil_sump	oil_sump	**1.0**
	oil_sump	v-belt	0.0
	oil_sump	gears	0.0
	v_belt	engine_block	0.0
	v_belt	cylinder_block	0.0
	v_belt	oil_sump	0.0
	v_belt	v-belt	**1.0**
	v_belt	gears	0.0
	gearing	engine_block	0.0
	gearing	cylinder_block	0.0
	gearing	oil_sump	0.0
	gearing	v-belt	0.0
	gearing	gears	**0.5**
	camshaft	engine_block	0.0
	camshaft	cylinder_block	**0.0667**
	camshaft	oil_sump	0.0
	camshaft	v-belt	0.0
	camshaft	gears	0.0
	engine_block	engine_block	**1.0**
	engine_block	cylinder_block	0.3333
	engine_block	oil_sump	0.0
	engine_block	v-belt	0.0
	engine_block	gears	0.0
	Total ($r_{n-grams}$)**: 3.5666/ 5 = 0.7133**		

Table 6.1: N-grams Results between "Motor" and "Engine"

Each line of the Table holds a pair of terms (characteristics) and the respective n-grams result. The total result $r_{n-grams}$ for each data type is calculated by formula (6.1). The final similarity value, which takes into account all comparisons performed between data types and relations, is given by formula (6.2):

$$sim_{attrSet1/attrSet2} = \frac{\sum r_{n-grams}}{n} \tag{6.2}$$

where n is the number of different attribute types that the requested product contains (in this example the concept "Motor"). The final result $r_{n-grams}$ ranges between zero and one, where zero signifies total dissimilarity and one means complete identity.

Table 6.2 presents the final result for the similarity comparison between the attributes of "Motor" and "Engine", including data types *string*, *integer* and *float* and the relation *has-part*. In this example, the sum of all comparisons is divided by four, since the concept "Motor" contains attributes of type *string*, *integer*, *float* and the relation *has-part*.

Attribute Type	**N-grams**
Integer	0.7531
String	0.68
Float	0.5989
has-part **relation**	0.7133
$sim_{attrSet1/attrSet2}$: **3.4316 / 4 = 0.6863**	

Table 6.2: Overall Similarity between "Motor" and "Engine"

The comparison between "Motor" and "Engine" results in a $sim_{attrSet1/attrSet2}$ value equal to 0.6863 which indicates a quite significative similarity.

As a counter-example, Table 6.3 compares the attributes of "Motor" and "Wheel". The sum of the relevant n-grams for *string* values is divided by three since "Motor" contains three *string* attributes. The sum of the relevant n-grams for *float* values is divided by two because "Motor" contains two *float* attributes. In this case, the total value of ($r_{n-grams}$) is 0.1101 for the *string* data type and 0.0277 for the *float* data type.

Table 6.4 presents the final result of the similarity comparison between the attributes of "Motor" and those of "Wheel". In this case, it was only possible to calculate the similarity between the *string* and *float* types since they are the only attribute data types of "Wheel".

The final similarity value $sim_{attrSet1/attrSet2}$ is 0.0345. This value implies a very low similarity between the two given concepts.

In order to find the most similar concept (product), the concept "Motor" can be compared with all concepts contained in the target ontology, resulting in a slow, inefficient process. To avoid this brute force algorithm, we apply the pre-selection mechanism explained previously.

Type	Attributes of Motor	Attributes of Wheel	N-grams
String	arrangement_of_cylinders	manufacturer	**0.0**
	arrangement_of_cylinders	wheel_rim_material	0.0
	arrangement_of_cylinders	tyre_type	0.0
	fuel	manufacturer	0.05
	fuel	wheel_rim_material	**0.0526**
	fuel	tyre_type	0.0
	transmission_type	manufacturer	0.0
	transmission_type	wheel_rim_material	0.0
	transmission_type	tyre_type	**0.2777**
	Total($r_{n-grams}$)**: 0.3303 / 3 = 0.1101**		
float	fuel_consumption	diameter	**0.0**
	fuel_consumption	tire_profile_depth	0.0
	fuel_consumption	width	0.0
	cylinder_capacity	diameter	**0.0555**
	cylinder_capacity	tire_profile_depth	0.0
	cylinder_capacity	width	0.0
	Total($r_{n-grams}$)**: 0.0555 / 2 = 0.0277**		

Table 6.3: N-gram Results for Attributes of "Motor" and "Wheel"

Attribute Type	N-grams
String	0.1101
Int	not available
Float	0.0277
has-Part **relation**	not available
$sim_{attrSet1/attrSet2}$: **0.1378 / 4 = 0.0345**	

Table 6.4: Overall Similarity between "Motor" and "Wheel"

6.6.1.2 Similarity Matching between Descriptions

When experts describe concepts that belong to the same domain, they adopt a common technical terminology. As a result, it is reasonable to expect finding similarities in the descriptions of corresponding concepts.

The OSAg first removes "stopwords", which are words that, from a non-linguistic point of view, are irrelevant (*e.g.*, a, about, above, across, the, for, *etc.*). Then, the most representative words are extracted from the description, leading to short, precise descriptions of the requested product and of the matching candidates. Words that occur several times in one description are considered only once and punctuation marks ignored.

The description of the requested product will be compared with the description of each candidate product (obtained from the pre-selection process) resulting in a similarity matrix. By adding the matrix values, we obtain the result of comparing both descriptions.

As an example, let's compare the description of "Battery" with the description of "Storage_Battery". The description for "Battery" is "a device that produces electricity" while "Storage_Battery" is described as "a device that stores energy and produces electric current by chemical action". After removing the "stopwords", the n-grams matrix can be calculated (Table 6.5). The pre-processed description of the requested product ("device", "produces" and "electricity") is displayed in the first line of the table.

	device	produces	electricity
device	**1.0**	0.0	0.0
stores	0.0	0.1111	0.0
energy	0.0	0.0	0.0833
produces	0.0	**1.0**	0.0
electric	0.0	0.0	**0.6666**
current	0.0	0.0	0.0
chemical	0.0	0.0	0.0
action	0.0	0.0	0.0

Table 6.5: Matrix with N-grams for Battery *vs.* Storage_Battery Descriptions

The n-grams results are combined using the formula (6.1). Hence, the result for this example is (Formula 6.3):

$$r_{n-grams} = \frac{\sum_{i=1}^{n} max_i}{n} = \frac{1.0 + 1.0 + 0.6}{3} = 0.8 \tag{6.3}$$

The values that appear in the numerator of this formula are the highest scores obtained for each relevant word selected from the description of the product. Their sum is divided by three since the description of "Battery" consists of three relevant words. Table A.1, in Appendix A, presents results of similarity comparisons performed between several descriptions of products.

6.6.1.3 Similarity Matching between Concepts

WordNet is a lexical database designed for automatic processing that provides an effective combination of traditional lexicographic information and modern computing. WordNet contains more than 118 000 different word forms and more than 90 000 different word senses and includes synonymy (same-name), antonymy (opposite-name), hyponymy (sub-name), hypernymy (super-name), meronymy (part-name) and holonymy (whole-name) relations.

We made experiments with WordNet using a Comprehensive Perl Archive Network (CPAN) module[6] that implements a variety of semantic similarity measures [Budanitsky and Hirst, 2001]. In particular, we evaluated the measure techniques proposed by Resnik, Jiang-Conrath, Leacock-Chodorow, Hirst-St-Onge and Wu-Palmer in order to select the most appropriate to our case. After this evaluation, which is presented in Chapter 7, we chose the Leacock-Chodorow (LCH).

The similarity measure technique proposed by Leacock-Chodrow finds the shortest path between two concepts, counting up the number of edges between the senses in the *is-a* hierarchy of WordNet. The retrieved value is then scaled by the maximum path length in the WorldNet *is-a* hierarchy. If no error occurs, a related value is obtained by taking the negative logarithm of this scaled value. Whenever an error occurs or no path exists between the words since one of the words is not represented in WordNet, an error string is created.

The LCH measure technique requires two word senses as input parameters. The input format is **word#pos#sense**, where **word** is a term, **pos** identifies the type of the word (**n** for noun, **v** for verb, **a** for adjective and **r** for relation) and **sense** is a positive integer and represents the meaning of the word in WordNet. For example, the format car#n#1 refers to the first meaning of the noun car in WordNet. Several taxonomies exist inside WordNet, but in our scenario only nouns are relevant. Thus, only **n** is considered for **pos**.

Since a word in WordNet has usually several meanings, the relevant meaning has first to be chosen. It has to be clear which meaning of the concept is correct in the given context. Normally this is done by requesting the user to choose the meaning by gloss or by a set of synonyms (synset). This approach is not

[6]CPAN, http://www.d.umn.edu/~tpederse/similarity.html, November, 2004.

appropriate since it would constrain the autonomy of our OSAg. For example, when querying WordNet for the concept "car" there are five meanings:

1. Car, auto, automobile, machine, motorcar – (4-wheeled motor vehicle; usually propelled by an internal combustion engine; "he needs a car to get to work").
2. Car, railcar, railway car, railroad car – (a wheeled vehicle adapted to the rails of railroad; "three cars had jumped the rails").
3. Cable car, car – (a conveyance for passengers or freight on a cable railway; "they took a cable car to the top of the mountain").
4. Car, gondola – (car suspended from an airship and carrying personnel and cargo and power plant).
5. Car, elevator car – (where passengers ride up and down; "the car was on the top floor").

These results are presented as a set of synonyms (synset) followed by its gloss, *e.g.*, for the first meaning, while "car, auto, automobile, machine, motorcar" represents the synset, "he needs a car to get to work" constitutes the gloss.

Our proposed solution takes into account all possible meanings. Each meaning of one concept is compared with the meanings of the other concept. The pair of meanings with the highest value is considered the best similarity value. However, this procedure does not guarantee that the correct meaning is selected. Our experiments showed that this procedure generally delivers applicable results since the highest meaning indicates, in most cases, that those two concepts from the same domain have been matched with success.

A disadvantage of this approach is the high number of comparisons that will be performed if both words have several meanings. Therefore, all results are stored for future negotiation processes. This way, whenever the same concepts need to be compared again, the repetition of the whole process is avoided.

Table 6.6 presents the LCH scores obtained when comparing "klaxon", which has only one meaning in WordNet, with "horn", which has ten meanings. The value in bold shows that the pair klaxon#n#1 and horn#n#8 delivers the highest score. As a result, the 8^{th} meaning of "horn" is the one that best matches the concept "klaxon". The other scores are disregarded.

WordNet has several synonyms for certain concepts and few or none for others. In specific domains, like the automobile assembling domain, this may happen. In theses cases, we consider only the similarity matching results calculated using the attributes, the *has-part* relations and the descriptions of the involved concepts.

word 1	word 2	LCH score
klaxon#n#1	horn#n#1	1.7918
klaxon#n#1	horn#n#2	1.0186
klaxon#n#1	horn#n#3	0.8109
klaxon#n#1	horn#n#4	1.0986
klaxon#n#1	horn#n#5	1.5041
klaxon#n#1	horn#n#6	1.0186
klaxon#n#1	horn#n#7	1.0986
klaxon#n#1	horn#n#8	**2.8904**
klaxon#n#1	horn#n#9	1.5041
klaxon#n#1	horn#n#10	2.1972

Table 6.6: LCH Results for "klaxon" *vs.* "horn"

6.6.1.4 Final Matching Result

If at least two of the three matching mechanisms described deliver a convincing result, the OSAg believes it has established a well-founded correspondence between the terms under analysis. Nevertheless, if only one of the similarity methods has provided sound results (the n-grams-based comparison between Descriptions (D), the n-grams-based comparison between Attributes + Relations (AR) and the WordNet-based LCH measure technique), then the OSAg is not confident on the result. In this case, the OSAg calculates a final similarity value by performing a weighted combination of the single results obtained.

Weighted Combination

The idea of using weights to compute a final result is based on the assumption that the three approaches, due to their intrinsic characteristics, exhibit different reliability levels. LCH, since is based on WordNet, delivers the most trustworthy results. The other algorithms depend more on the arbitrariness of the ontology developers. The comparison between descriptions is the algorithm associated with highest uncertainty since it is possible to describe the same product using completely different words.

In order to reflect these assumptions, different weights are applied to the single results obtained prior to the calculation of the final result using formula (6.4):

$$sim_{term1/term2} = \frac{\sum_{i=1}^{n} result_{methods} * weight}{n} \tag{6.4}$$

where n can be two or three, depending on the number of single results.

The numerical values used for weighting the different types of results (D, AR and LCH) were established based on experimental tests. The final weights combinations are presented in Table 6.7. The weights also depend on the number of results available for combination.

Results Available	D+AR+LCH	D+AR	D+LCH	AR+LCH
D	0.5	0.5	0.5	-
AR	1	1.5	-	0.5
LCH	1.5	-	1.5	1.5

Table 6.7: Weights Combination

Using the weighted combination, the similarity results are in general higher, not only when the terms describe the same product, but also when terms that are unrelated. Our experiments provided reasonable results since the similarity values of matching terms are higher than the matching results involving dissimilar terms. These results are presented and discussed in the next Chapter.

Classification

The final matching results are then classified into three qualitative categories: high-confidence match, moderate-confidence match and weak-confidence match. The classification criteria is based on the definition of a confidence threshold.

The final results range from zero to one, where zero indicates total dissimilarity and one indicates complete identity. In order to define the threshold, the precision and recall ratios defined in [Baeza-Yates and Ribeiro-Neto, 1999] are used. Precision and recall ratios are frequently used when evaluating search strategies and are usually expressed as percentages.

Recall defines the *ratio* between the number of relevant records retrieved and the total number of relevant records in the database. The Recall ratio formula is:

$$Recall = \frac{A}{A+B}100\% \tag{6.5}$$

where A is the number of relevant records retrieved and B is the number of relevant records not retrieved.

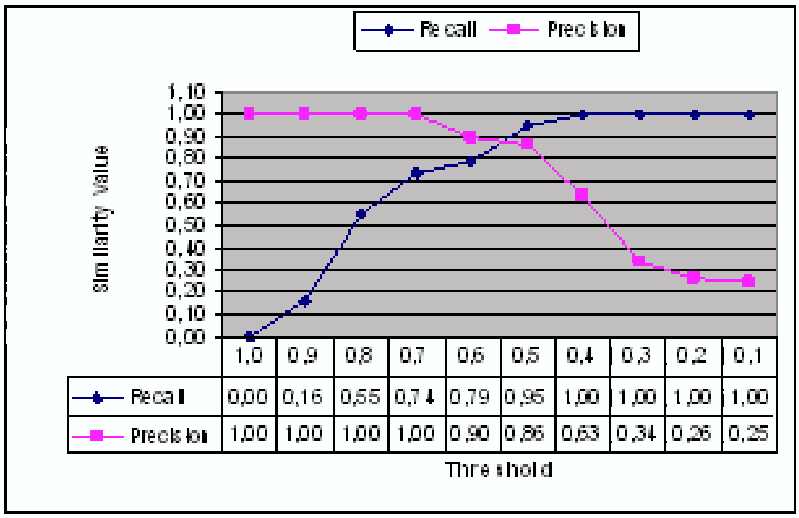
Precision defines the *ratio* between the number of relevant records retrieved and the number of irrelevant and relevant records retrieved. The Precision formula is given by the expression:

$$Precision = \frac{A}{A+C}100\% \tag{6.6}$$

where A is again the number of relevant records retrieved and C is the number of irrelevant records retrieved.

The values for precision and recall are considered excellent if they are equal to one, meaning that all retrieved records are relevant.

The curves used to define the threshold are plotted and are showed in Figure 6.16. This graphic shows the quantity of information returned together with the correspondent precision level when, for each item retrieved from the target ontology, there was a match with the item from the source ontology.



	1,0	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
Recall	0,00	0,16	0,55	0,74	0,79	0,95	1,00	1,00	1,00	1,00
Precision	1,00	1,00	1,00	1,00	0,90	0,86	0,63	0,34	0,26	0,25

Figure 6.16: Recall and Precision for Defining Threshold

We may observe that, when the threshold is higher than 0.7, the applied methods retrieve more than 70% of all information, with a 100% confidence level. When the threshold decreases, precision also diminishes. The point where the precision and the recall curves overlap is approximately at a confidence value of 0.55. This determines the threshold used in the application. Lower values indicate insufficient similarity, *i.e.*, the system lacks confidence on the matching result. In these cases, the OSAg concludes that the applied methods are unable to find a suitable matching item.

If more than one comparison result is above the 0.55 threshold, the item with the highest value is proposed. The similarity values range from 0.55 to 1.0. Finally, the similarity values are classified into three categories: high-confidence, moderate-confidence and weak-confidence, as proposed in Table 6.8.

Once the similarity between terms from both ontologies has been established, the OSAg sends to the SEAg an INFORM performative containing the matching concept and the associated confidence category.

Classification	Description
weak-confidence	This level indicates that the two terms are slightly similar. It means that the information available was insufficient to establish the similarity with higher confidence between terms. The range is between 0.55 and 0.59.
moderate-confidence	This level indicates that the two terms match with moderate confidence (with some degree of uncertainty). Referring to the graphic, it is exactly the point where the confidence in the match increases. The range for this level is between 0.6 and 0.69.
high-confidence	This level indicates that the two terms match with high confidence, meaning that the agents can fully trust this result. The range is between 0.7 and 1.0.

Table 6.8: Final Similarity Classification

6.6.1.5 Basic Learning Mechanism

Depending on the application field, there are several definitions for learning. According to the Cambridge Dictionary a definition for learning is "the activity of obtaining knowledge". This definition is related with our intuitive notion of learning as "the process of acquiring new information/expertise/skills or restructuring some previously acquired knowledge". Usually, the acquired knowledge can be used to improve a future activity.

Herbert Simon, in 1983 stated that learning means "... changes in the system that are adaptive in the sense that enable the system to do the same task or tasks drawn from the same population more efficiently and more effectively next time" [Simon, 1993].

Although machine learning techniques are useful in ontology alignment, they are not applicable to our case. Machine learning techniques take advantage of well-known methods as formal-concept analysis, Bayes learning, neural networks, *etc.* The main machine learning techniques used for ontology alignment are [Euzenat *et al.*, 2004]:

- **Supervised learning**, in which the ontology alignment algorithm learns how to work through a data set made of good alignments (positive examples) and bad alignments (negative examples). The drawback of this approach is that it is difficult to know which techniques work well for which ontological features. As a result, an ontology alignment algorithm learned from several pairs of ontologies might not necessarily work well for a new pair of ontologies.

- **Learning from data**, in which a data set of instances is communicated to the algorithm together with their relations and the classes they belong to. From this data, the algorithm learns the relations between classes and the alignment of properties.

In our approach, learning is applied with the purpose of improving the performance of the MTS and, consequently, the negotiation process. Once the OSAg has established the similarity between a pair of terms from different ontologies, this knowledge is stored in order to be available for future negotiation rounds. The performance improvement occurs with time: as the number of negotiations rounds increases, so does the amount of matched terms memorised.

We have implemented a Basic Learning Mechanism (BLM) that does not imply any complex machine learning technique. In our scenario, the learning process occurs during negotiation when agents need to match private and communicated product representations.

The OSAg stores in XML format the pairs of agents involved in each negotiation round, the respective negotiated concepts and their final confidence classification (according with Table 6.8). In addition, another XML file stores a global list holding the pairs of concepts successfully matched by different pairs of agents. The version of the involved ontologies is also stored since ontologies are dynamic and may change between negotiation rounds. This BLM is triggered before the pre-selection process.

The algorithm starts with both CEAg and SEAg sending to OSAg the date of their last ontology update. Based on this date, the OSAg verifies if the last ontology update is more recent than the last negotiation date in which both agents were involved. If any one of the ontologies has changed, it will avoid using concepts previously learned that are no longer defined in the CEAg or in the SEAg ontologies. The BLM applies the following steps (presented in Figure 6.17):

1. If any of the entreprise agents (CEAg or SEAg) has performed an ontology update since their last negotiation, then it is necessary to perform the matching between all terms, starting from the pre-selection process.

2. If the last ontology update date is previous to the last negotiation date, then it means there is no modification in the ontologies and the learned concepts may still be considered in this negotiation process.

3. If the pair of agents has negotiated the concept beforehand, it means that the OSAg has already the result. The OSAg sends to the SEAg the concept that best matches the requested concept together with its confidence level without reinvoking the MTS.

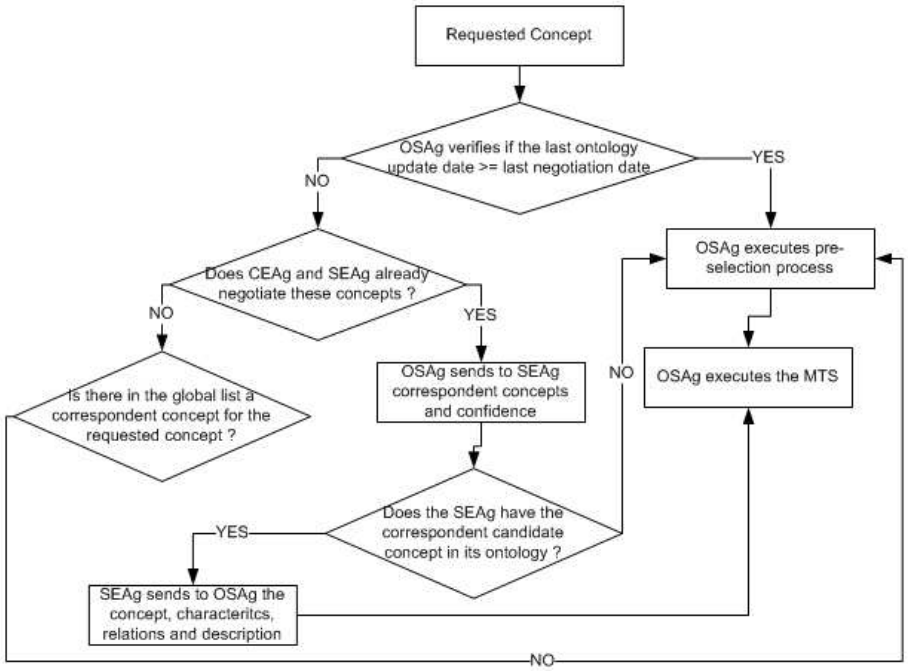


Figure 6.17: Basic-Learning Algorithm

4. If CEAg and SEAg have never negotiated the concept under consideration, the OSAg may still try to improve the MTS result. The OSAg verifies if there is in the global list a similar concept already learned from previous negotiation rounds involving other SEAgs.

5. If the OSAg finds a similar concept in the global list, the OSAg sends its correspondent concept to the SEAg. The SEAg looks for this correspondent concept in its private ontology before starting the pre-selection process. If the SEAg contains the correspondent concept, the OSAg executes the MTS taking it into consideration, reducing the number of matching concept candidates that are selected in the pre-selection process. The MTS is still invoked to confirm if the concept in the global list is identical to the one in private ontology of the SEAg.

6. If the OSAg finds a similar concept in the global list that is absent from the SEAg ontology, then it is necessary to perform the pre-selection process, followed by the invocation of the MTS for all matching concept candidates.

Some experiments illustrating the learning process are presented in the next Chapter.

6.6.2 Units Conversion Services

Interoperability problems between agents in a B2B context do not result only from different knowledge entities or distinct types of entities. During a negotiation process, agents may use different currency or measurement units, generating additional interoperability problems.

The use of different currency units poses a serious problem in the B2B e-commerce domain. When one agent analyses a product offer and the price is represented using a different currency, the agent is incapable of making a reasonable proposal. One solution could be to define a standard currency for the negotiation process. This would require all agents converting their price lists into a standard currency. Alternatively, the agents could negotiate in their own currency and use an external service to perform this conversion whenever necessary. This way, an agent using the "Japanese Yen" could negotiate with an agent using "Euro".

Similar problems happen when agents use different measurement units to represent the attributes of products. It is necessary to avoid requesting a product with some characteristics and receiving a product with different features because there was a measurement units misunderstanding during the negotiation.

To provide our MAS with the ability to solve these simpler types of interoperability problems, we implemented and integrated in the system the UCS, *i.e.*, a set of currency and measurement units conversion services. These services are implemented as Web services and they are exemplified in the next Chapter.

The enterprise agents request these services to the OSAg by sending a FIPA-ACL message. In return, they receive a message with the exchange rate between the specified currency units or the conversion value between the detailed measurement units.

6.7 Conclusions

Four types of agents are participating in the system:

- Customer Enterprise Agents representing the entities interested in purchasing components according to specific requirements.
- Supplier Enterprise Agents representing entities willing to sell their products or services whenever possible, *i.e.*, whenever the requested good is in stock or the service is available. In order to make offers, supplier agents need to understand the requests of customer agents.

- Facilitator Agent that provides a yellow-pages service and renders possible the negotiation process by bringing potential business partners together.

- Ontology-based Services Agent which is responsible for providing services to support negotiation.

Agents use ontologies to ascribe meanings to the terms they represent. Every agent in our MAS commits to a top-level ontology that defines an e-commerce specific vocabulary made of terms used during the negotiation process. This ensures that all agents uniformly interpret the senders intention to negotiate. However, this does not imply the correct identification of the requested product or service since each agent represents this domain-specific information in a private ontology. The private domain ontologies are stored in OWL format, while the E-Commerce Ontology is represented through Java classes. Both types of ontologies were created with the Protégé editor.

The OSAg provides services to help supplier agents understanding the contents of the CFP messages issued by customer agents since the requested products may be undefined in their private ontologies. The OSAg exchanges messages with both supplier and customer agents in order to gather the information required to successfully match terms from the two ontologies. Furthermore, the supplier agents implement a pre-selection process, which is based on the price of the requested product, that aims at narrowing the set of potential corresponding candidates.

The OSAg was enhanced with a basic learning mechanism that enables the learning of the concepts already compared and matched, avoiding repeating previous similarity matching processes.

The proposed solution for solving the interoperability problems applies methods from linguistic data processing, including the detection of lexical similarities, with the n-grams algorithm and semantic similarities with the LCH WordNet-based measure technique. The lexical measure techniques compare the attributes, relations, name and description of a pair of concepts, which are expressed in natural language. The implementation groups the attributes by data types leading to meaningful attribute comparisons. LCH takes into account the names of the concepts and scales them according to the WordNet's *is-a* hierarchy. A weighted average is performed to combine the similarity values delivered by the three matching techniques into a unique similarity result.

The accuracy of the matching techniques implemented depends on the quantity of information (attributes and descriptions) contained in the ontologies and in WordNet. Although WordNet is a very large lexical database containing more than 120 000 words, it does not contain all terms. Lack of information can either

cause the system to conclude its inability to propose any matching result or that the result is not well-founded.

The UCS as well as the MTS services are detailed in the next Chapter.

Chapter 7

Implementation and Experiments

In this Chapter we describe the implementation of our test-bed and the evaluation of the developed methodologies. The agent based ontological services, which are internal server-side services, are tested and the results obtained both with the units conversion services and the matching terms service are discussed. The WordNet-based similarity measure technique adopted, the weighted combination of the individual similarity results and the basic learning algorithm are also illustrated. Finally, we present the roles of JADE, JENA and of the BeanGenerator plug-in in the creation of ontologies.

7.1 Introduction

As described in Chapter 1, the aim of this thesis is the development of agent-based methodologies capable of supporting the automatic negotiation required by an EI during the formation of a VE.

Several problems are involved in the VE formation process. The most important issue that must be addressed is the lack of understanding that may occur between agents due both to structural and semantic differences in the representation of concepts. This problem can also occur during the identification of needs stage, when it is necessary to describe a needed product or service so that it can be understood by all enterprise agents.

In our implementation, the developed services are essential since our agents have different individual views of the world. These distinct perspectives are explicitly defined by different ontologies. During negotiation, the different views need to be reconciliated to ensure effective communication. While in a tradi-

tional setup, the agents can commit to use some common ontology, in open environments, populated by heterogeneous, autonomous agents, it is unlikely to have a common ontology. Each agent will typically use a different private ontology since enterprises will not consider converting the content of their ontologies unless the new ontology is considered as a *de facto* standard, which is not currently the case.

As a result, we created the OSAg which finds the correspondence (similarity) between concepts (products) from two ontologies based on their names, characteristics, relations and descriptions. Furthermore, the interoperability problems that may happen due to the use of different currency or measurement units are also solved by means of dedicated conversion Web services.

7.2 Agents

In Chapter 6 we described the system architecture and the roles of the four types of agents we developed: the Facilitator Agent (FAg), the Customer Enterprise Agent (CEAg), the Supplier Enterprise Agent (SEAg) and the Ontology-based Services Agent (OSAg). In addition to these agents and since JADE is the adopted agent development platform, there is also a set of default JADE agents: the Directory Facilitator (DF), the Agent Management System (AMS) and the Agent Communication Channel (ACC).

7.2.1 Facilitator Agent

The FAg interacts with the Directory Facilitator (DF) and is responsible for matching the enterprise agents. By matching agents we mean finding prospective business partners, *i.e.*, customer or supplier enterprise agents in the automobile assembling domain that are willing to negotiate together. There is a simple GUI for agents to register and communicate. The agents can be selected from the list of the enterprise agents present in the platform. Figure 7.1 shows this interface.

7.2.2 Customer Enterprise Agent

The CEAg represents enterprises interested in buying components to assemble some final product. In order to make an announcement, the user has to specify the component (product), quantity and currency in which the enterprise, represented by the CEAg, requires the proposals.

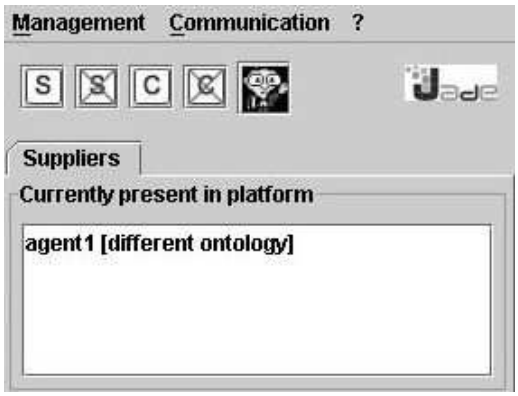


Figure 7.1: Interface to Create Enterprise Agents

Figure 7.2 shows the GUI used to trigger the process. This action results in the posting of an announcement specifying the component required to build the automobile. In Figure 7.2, the requested product is a "Motor", the quantity is "100" units and the currency is "Euro".

Figure 7.2: Interface for the CEAg

Figure 7.3 illustrates the available currency options. Whenever a CEAg sends an announcement to the FAg, the FAg forwards the announcement message to all registered SEAg agents that may provide the requested product.

Figure 7.3: Currency Options

7.2.3 Supplier Enterprise Agent

A SEAg represents an enterprise interested in providing some kind of product. Figure 7.4 shows the GUI for creating a SEAg. It is necessary to provide a name and choose the agent's ontology type (a common or a private ontology).

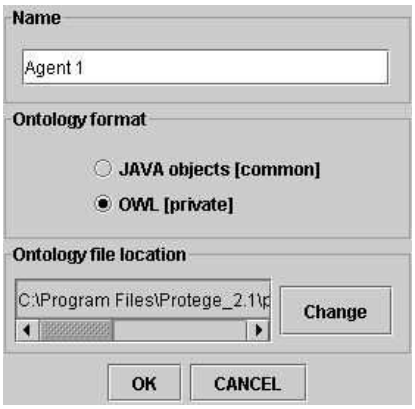


Figure 7.4: Interface for the SEAg Creation

We decided to allow the creation of two types of SEAg agents:

- Supplier agents that share a common domain ontology, *i.e.*, share an identical view of the domain.
- Supplier agents that use distinct private domain ontologies, *i.e.*, hold different domain perspectives.

As a consequence, two different kinds of supplier enterprise agents can be created. However, the focus of this thesis is concentrated on the second type of agents called "OWL [private]". In this case, the new supplier enterprise agent uses an OWL defined ontology that may be different from the CEAg's own ontology.

If a "OWL [private]" ontology is chosen, the button labelled "Change" is enabled, allowing the user to specify the path to the OWL file that defines the desired ontology. A default path is always predetermined. Figure 7.5 shows the dialog interface window that allows the selection of the ontology (*e.g.*, "AutomobileAssemblingOntology.owl").

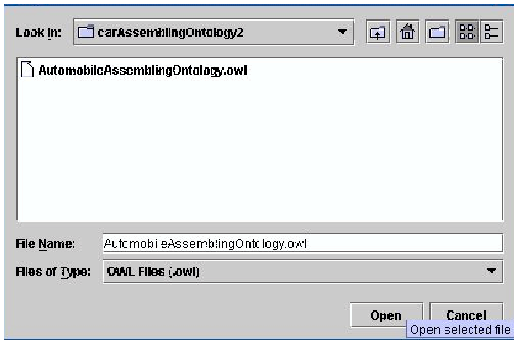


Figure 7.5: Interface for Ontology Selection

Figure 7.6 presents the SEAg's GUI created to visualise the messages exchanged with other agents. This figure presents a snapshot of the messages exchanged with the OSAg regarding the acquisition of 100 "Motor" units. Since the SEAg is unaware of the product name, it requests the assistance of the OSAg. Based on the similarity matching results obtained, the OSAg concludes that the SEAg's candidate-product that best matches "Motor" is "Engine" and informs the SEAg about its finding.

Once the SEAg understands the content of the announcement, if the announced product is available, it makes an offer. At least one SEAg has to be present in the platform so that communication can be established.

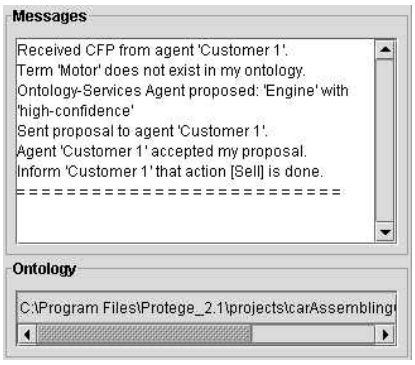


Figure 7.6: SEAg Interface for Message Monitoring

7.2.4 Ontology-based Services Agent

The OSAg is responsible for providing the enterprise agents with services that enable effective negotiations. In the example presented on the previous figures, the SEAg requests the assistance of the OSAg to find a suitable matching term for "Motor". The OSAg then exchanges messages with the CEAg, to gather information about the product "Motor", and with the SEAg, to receive the list of candidate-products obtained through the pre-selection process. The pre-selection of candidate-products is price-based.

Figure 7.7 shows the results of the similarity matchings calculated for each candidate-product submitted together with the final result obtained.

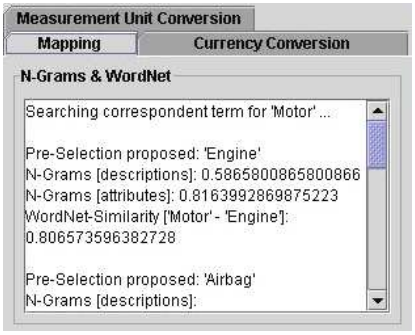


Figure 7.7: OSAg Processing the MTS

7.2.4.1 Matching Terms Service

Table 7.1 displays the MTS results for the product "Motor". The list of candidate-products submitted by the SEAg contains "Engine", "Cylinder_Block", "Transmitting_Aerial", "Wheel" and "Push_Rod".

The first column in Table 7.1 presents the individual similarity values calculated by the OSAg for each pair of concepts, *i.e.*, the requested product and the candidate-product. The second column presents the final result, that ranges from zero to one, with the correspondent classification already explained in Chapter 6. The OSAg establishes, with a high level of confidence, that "Engine" matches "Motor".

Individual Similarity Values	**Final result**
Pre-Selection proposed: Engine N-Grams [descriptions]: 0.58 N-Grams [attributes]: 0.82 Semantic-Similarity: 0.80	0.77 **high-confidence**
Pre-Selection proposed: Cylinder_Block N-Grams [descriptions]: 0.20 N-Grams [attributes]: 0.5 Semantic-Similarity: 0.36	0.38
Pre-Selection proposed: Transmitting_Aerial N-Grams [descriptions]: 0.06 N-Grams [attributes]: 0.07 Semantic -Similarity: 0.55	0.31
Pre-Selection proposed: Wheel N-Grams [descriptions]: 0.01 N-Grams [attributes]: 0.03 Semantic -Similarity: 0.45	0.24
Pre-Selection proposed: Push_Rod N-Grams [descriptions]: 0.14 N-Grams [attributes]: 0.03 Motor or Push_Rod not found in WordNet	0.06

Table 7.1: Similarity Matching for "Motor" with Pre-selection

In this example, the CEAg's ontology has a total of 27 concepts and the SEAg's ontology has 29 concepts. The ontologies specifications include, for each concept (product), its characteristics (attributes) with the correspondent data types, a natural language description explaining the meaning of the concept and a set of relationships connecting the represented concepts.

Due to the pre-selection process performed by the SEAg, the OSAg has only to calculate the similarities between the requested product "Motor" and five candidate-products. In the absence of this pre-selection process, the OSAg would have to calculate the similarity matchings between "Motor" and the 29 concepts contained in the SEAg's ontology. Table 7.2 shows the results of comparing "Motor" with all products represented in the SEAg ontology.

Requested Product	**Candidate-product**	**Final Similarity**
Motor	Windscreen	0.31
	Speed_Indicator	0.36
	Oil_Sump	0.00
	Parking_Brake	0.39
	Engine	0.77
	Car_Window	0.33
	Rearview_Mirror	0.36
	Accelerator	0.41
	Long_Distance_Light	0.00
	Foglamp	0.35
	Suspension_System	0.37
	Suspension_System	0.38
	Airbag	0.07
	Stoplight	0.25
	Electric_Battery	0.45
	Cylinder_Block	0.38
	Transmitting_Aerial	0.31
	Sparking_Plug	0.44
	Wheel	0.24
	Engine_Block	0.32
	Headlight	0.39
	Clutch	0.39
	Belt	0.34
	V-Belt	0.04
	Parking_Light	0.10
	Low_Beam_Light	0.00
	Push_Rod	0.06
	Car_Horn	0.43
	Gear_Lever	0.38

Table 7.2: Similarity Matching for “Motor” without Pre-selection

In order to build Table 7.2, the CEAg asked sequentially for all the items listed in its ontology. Whenever the SEAg was unable to understand the requested item, it queried the OSAg. The OSAg performed the matching and informed the SEAg of the final result.

Appendix A presents Table A.2 with all the experiments performed using different ontologies. These experiments include the weighted combination algorithm described in Table 6.7.

The SEAg's ontology has 19 concepts with correspondent terms in the CEAg's ontology. These pairs of concepts, which differ in name, attributes, relations or description, were successfully matched by the OSAg. There are also eight concepts without any correspondence. When the similarity between all concepts was performed using the established confidence threshold of 0.55, the OSAg found 17 correspondent concepts out of the 19 with correspondence. In summary, the OSAg failed to return two matching results. The remaining eight concepts were correctly unmatched.

The proposed method was 89% accurate. Nevertheless, if we consider the correctly unmatched terms, the accuracy rises to 92%. The efficiency of the method depends on the quantity of information available and on the quality of the concepts description. Obviously, when the item to be matched is represented in the WordNet database, the chances of obtaining a correct result increases.

Some examples presented in Appendix A, Table A.2, show that even corresponding products that are registered in WordNet may not be found because of weak matchings between attributes and poor concept descriptions. It is the case of the following correspondent pairs: ("bumper", "push_rod") and ("claxon", "car_horn").

Weighted Combination

As explained in Chapter 6, we are performing a weighted average (Formula (6.3)) to determine the final similarity value between each pair of terms. Alternatively, we can calculate the standard average using Formula (7.1):

$$sim_{term1/term2} = \frac{\sum_{i=1}^{n} result_{methods}}{n} \tag{7.1}$$

where n can be two or three depending on the number of single results to be considered. In this case, the result is the average of the individual similarity matching results.

Comparing the results obtained with Formula (6.3) and Formula (7.1), we concluded that the weighted average performed better than the standard average. The individual weights used in Formula (6.3) were established experimentally.

Table 7.3 presents in the first two columns the matched concepts, followed by the matching terms final result calculated with the standard and the weighted average formulas and, in the last column, their difference. The results for the concepts without correspondence can be found in Appendix B, Table B.1. The weighted average produces results that are higher, both for matching and non-matching terms, than the ones obtained with the standard formula.

The 15% value presented at the end of the Table 7.3 is the average difference between the results obtained with both formulas. Considering the comparison between "Throttle" and "Accelerator", while the final result using the standard average is 0.54, with the weighted average it increases to 0.77.

Requested Concept	**Correspondent Concept**	**Standard Average**	**Weighted Average**	**Difference**
Suspension	Suspension_System	0.62	0.81	19%
Throttle	Accelerator	0.54	0.77	23%
Clutch_pedal	Clutch	0.62	0.81	19%
Antenna	Transmitting_Aerial	0.74	0.84	9%
Hand_Brake	Parking_Brake	0.61	0.80	20%
Mirror	Rearview_Mirror	0.41	0.55	14%
Seat_Belt	Belt	0.41	0.55	14%
Claxon	Car_Horn	0.35	0.48	13%
Gearshift	Gear_Lever	0.80	0.90	10%
Windshield	Windscreen	0.57	0.78	22%
Headlamp	Headlight	0.66	0.83	17%
Taillight	Rear_Lamp	0.82	0.91	9%
Spark_Plug	Sparking_Plug	0.66	0.83	17%
Window	Car_Window	0.44	0.62	18%
Battery	Electric_Battery	0.69	0.85	15%
Speedometer	Speed_Indicator	0.73	0.87	13%
Motor	Engine	0.74	0.77	4%
				15%

Table 7.3: Final Matching Results - Standard *vs.* Weighted Average

Selection of the Semantic Similarity Measure Algorithm

As explained in Chapter 6, in order to select the WordNet-based similarity measure algorithm most appropriate to our scenario, we conducted a series of experiments to evaluate the algorithms of Resnik (RES), Jiang-Conrath (JCN), Leacock-Chodorow (LCH), Hirst-St.Onge (HSO) and Wu-Palmer (WUP).

We made several tests, that are presented in Appendix C, between automobile component names using the referred algorithms. To evaluate the similarity measure algorithms we calculated their precision and recall ratios (see Chapter 6).

In Figure 7.8, the precision and recall ratios are combined to show the retrieved information and their respective precision. The y-axis represents the precision ratio while the x-axis shows the percentage of information retrieved (recall ratio).

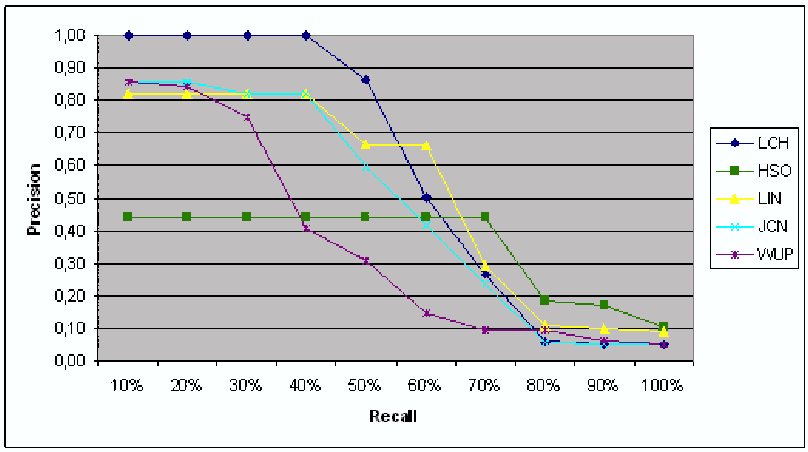


Figure 7.8: Evaluation of Similarity Measures

The LCH method presented the best results: 40% of instances are retrieved with no false positives. Even when we augmented the recall level and retrieved half of the data set, we still obtained a precision level above 80%, *i.e.*, LCH produces high quality results in our scenario.

WUP, JCN and LIN can be considered as second options since they produce values which are, in general, 20% below. With these algorithms, the recall/precision curve decreases considerably when the recall ratio is higher than 50%.

Since the invocation of LCH takes approximately 20 seconds per comparison, the results are stored on the server side for future negotiation rounds. As a result, if the same pair of concepts needs to be compared again, there is no need to repeat the whole matching process.

WordNet and LCH Remote Access

Common Gateway Interface (CGI) is a standard used to interface external programs with HTTP information servers[1]. It allows a Web page or any other kind

[1]CGI, http://hoohoo.ncsa.uiuc.edu/cgi/, October, 2004

of software program to invoke and run an application on the Web server's platform. CGI does not dictate the programming language and, consequently, any software program can be a CGI program as long as it handles input and output according to the CGI standard.

Both WordNet and the Perl implementation of LCH are installed on the HTTP server's platform. Furthermore, the LCH algorithm is a CGI compliant application. This distributed approach avoids replicating WordNet on all platforms where agents run.

In this case, the LCH algorithm, which is implemented in Perl, is transparently invoked by the OSAg. The following source code extracted shows how the OSAg interacts with the CGI script. The data, prior to be sent to the server, has to be converted into a Multipurpose Internet Mail Extensions (MIME) encoded `java.lang.String` object.

```
String data = URLEncoder.encode("method", "UTF-8") + "=" +
URLEncoder.encode(methodName, "UTF-8");
data += "&" + URLEncoder.encode("word1", "UTF-8") + "=" +
URLEncoder.encode(word1 + "#n#" + sense1, "UTF-8");
data += "&" + URLEncoder.encode("word2", "UTF-8") + "=" +
URLEncoder.encode(word2 + "#n#" + sense2, "UTF-8");
URL url = new URL("http://127.0.0.1/cgi-bin/server.pl");
URLConnection conn = url.openConnection();
OutputStreamWriter wr = new OutputStreamWriter(conn.getOutputStream());
wr.write(data);
```

The Universal Resource Locator (URL) specifies the location of the CGI script, including the Internet Protocol (IP) address (`127.0.0.1`) of the HTTP server platform and the path to the CGI script within the HTTP server (`server.pl`).

The script reads the client's input once it is submitted. The arguments passed to the script use the method GET, which is defined in the HyperText Transfer Protocol (HTTP). The parameters required by the method are included in the URL. The script invokes the desired Perl method, depending on the value of the `method` parameter. Once the CGI script receives a response from the invoked program, it integrates the result in an HTML page, creates an HTTP response message holding the page and dispatches the message to the client.

Since the information is returned in an HTML page, the incoming stream has to be parsed in order to extract the relevant information, *i.e.*, the result of semantic similarity matching between the two words submitted.

7.2.4.2 Basic Learning Mechanism

In Chapter 6 we described the Basic Learning Mechanism (BLM) developed to improve the performance of the MTS. In this subsection we exemplify the use of this algorithm.

Consider that a pair of agents, *e.g.*, Agent_1 and Agent_2, is negotiating a product, *e.g.*, "Motor". If the product is unknown by one of the agents and if it was previously negotiated by the current pair of agents, then the OSAg can, upon request, immediately send the stored matching term.

The example presented in Table 7.1 illustrates what happens when a CEAg requests for a product "Motor". The registered SEAg, which is unaware of such product, contacts the OSAg. The OSAg asks the SEAg for a list of candidate-products based on the CEAg's "Motor" price and receives a list of five candidate-products. The OSAg then matches "Motor" with each candidate-product to determine the best match and, finally, informs the SEAg that it is highly confident that "Engine" and "Motor" are corresponding terms. To improve this time and resource consuming process, the concepts compared (the five pairs of terms) as well as the agents involved (Agent_1 and Agent_2) are stored. Table 7.4 presents the list of concepts negotiated between Agent_1 and Agent_2 that required the help of the OSAg and that were, consequently, memorised by the BLM.

CEAg	**SEAg**	**Requested Concept**	**Correspondent Concept**	**Confidence Level**
Agent_1	Agent_2	Suspension	suspension_system	High
		Window	Car_Window	Moderate
		Throttle	Accelerator	High
		Clutch_pedal	Clutch	High
		Antenna	Transmitting_Aerial	High
		Hand_Brake	Parking_Brake	High
		Mirror	Rearview_Mirror	Weak
		Seat_Belt	Belt	Weak
		Gearshift	Gear_Lever	High
		Windshield	Windscreen	High
		Headlamp	Headlight	Strong
		Taillight	Rear_Lamp	High
		Spark_Plug	Sparking_Plug	High
		Battery	Electric_Battery	High
		Speedometer	Speed_Indicator	High
		Motor	**Engine**	**High**

Table 7.4: Concepts Comparison between Agent_1 and Agent_2

Using this knowledge, it is neither necessary for the SEAg ($Agent_2$) to perform the pre-selection process nor for the OSAg to invoke the MTS. The OSAg uses this list to find the correspondent concept for "Motor", which, in this case, is "Engine", and sends it with the respective confidence level to the SEAg.

Table D.1, in Appendix D, presents, for each pair of matched concepts, the established confidence level together with the enterprise agents involved.

Consider now the case of another SEAg, *e.g.*, $Agent_4$, that contacts the OSAg in order to find a suitable match for the announced product "Motor". The OSAg, before requesting $Agent_4$ for the list of candidate-products, suggests $Agent_4$ concept "Engine". This suggestion is based on the global list of matches performed which is generated and maintained by the OSAg. This process improves the efficiency of the MTS and, consequently, the negotiation process.

The list below presents the concepts memorised during a set of negotiations involving six agents presented in Appendix D, Table D.1.

Suspension : suspension_system
Window : car_window : auto_window
Throttle : accelerator
Clutch_pedal : clutch : car_pedal
Antenna : transmitting_aerial : aerial
Hand_brake : parking_brake : emergency_brake
Mirror : Rearview_mirror : car_mirror
Seat_belt : belt: Safety_belt: life_belt: Safety_belt
Gearshift: Gear_Lever: Gear_change: Gear
Windshield : Windscreen
Headlamp : Headlight
Tail_lamp : Rear_lamp : Tail_lamp : Rear_light : Tail_light
Spark_Plug : Sparking_plug : Electrical_Spark
Battery : Electric_Battery : Auto_battery
Speedometer : Speed_indicator : Speedo : Speed_controller
Motor : Engine : Auto_motor
Claxon : Klaxon : Horn
Bumper : Bumper_Guard
Parking_light : Side_light : Parking_lamp
Wheel : Auto_wheel
Tyre : Tire : Auto_tire
Fog_lamp : Fog_light
Brake_lamp : Brake_light
Handwheel : Steering_Wheel

As already explained in the previous Chapter, the performance improvement of the MTS occurs with time and depends on the quantity of the negotiations and on the number of matchings performed.

7.2.4.3 Units Conversion Services

The UCS are useful whenever the agents use different currency or measurement units. The currency and measurement units conversion services, which are provided as Web services, are explained below.

Units Currency Conversion

The currency conversion service helps converting prices whenever a (CEAg, SEAg) pair uses different currencies. The SEAg asks for the exchange rate between the two relevant currencies by sending an FIPA-ACL message to the OSAg. The obtained exchange rate can then be used to calculate the price that will be part of the SEAg's proposal.

This currency conversion service is implemented as a Web service. A Web service is a software system identified by a Uniform Resource Identifier (URI) whose public interfaces and bindings are defined and described using XML. It supports direct interaction with other software programs using XML-based messages exchanged via Internet-based protocols.

The Universal Description Discovery and Integration framework (UDDI) helps to find Web Services available on the Web. It is a lookup system that allows consumers to locate Web Services; it is comparable to a global yellow pages directory. The services description is public, can be downloaded and is stored in Web Services Description Language (WSDL) files. WSDL is the standard format for describing a Web service and specifies how to access a Web service and what operations it will perform. The information is expressed in XML.

A plug-in for Eclipse[2] called WSDL to Java Remote Interface (RMI) Compiler (WSDL2Java) generates client-side bindings to a Web Service. Once the WSDL file is imported into the agent project, the plug-in automatically generates the Java classes that are necessary to interact with the Web service.

From the WSDL file called `CurrencyConvertor.wsdl`, the plug-in generates the following files: `Currency.java`, `CurrencyConvertor.java`, `CurrencyConvertorLocator.java`, `CurrencyConvertorSoap.java` and `CurrencyConvertorSoapStub.java`.

The following code illustrates how to use the generated classes:

```
CurrencyConvertorSoapStub stub = new CurrencyConvertorSoapStub(new
URL("http:/www.webservicex.netCurrencyConvertor.asmx"), null);
double d = stub.conversionRate(Currency.fromValue(currency1), Cur-
rency.fromValue(currency2));
```

[2]"Eclipse is an open source community whose projects are focused on providing an extensible development platform and application frameworks for building software." <http://www.eclipse.org >

This straightforward solution allows the communication between a client application and a Web service. A stub object of the class `CurrencyConvertorSoapStub` is created on the client-side to support the communication. The URL passed in the constructor is defined in the WSDL file. The exchange rate between two currencies can be obtained using the method `conversionRate`. The implemented Web service delivers the current, up-to-date exchange rate between the two currencies specified.

Figure 7.9 presents an example of a currency conversion request made to the OSAg. In this example, the CEAg currency is Euro (EUR) while the SEAg currency is Dollar (USD).

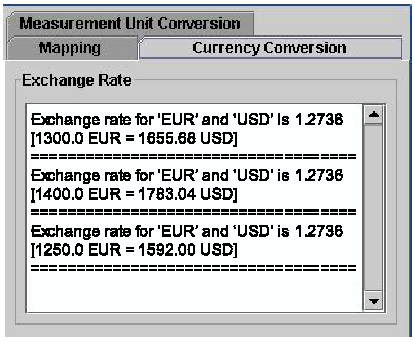


Figure 7.9: Currency Conversion - Dollar (USD) *vs.* Euro (EUR)

Measurement Units Conversion

Similarly to the currency conversion service, the assistance of the measurement units conversion service is required whenever a (CEAg, SEAg) pair uses different measurement units to represent the same/corresponding attributes. The SEAg requests the conversion between two measurement units by sending the appropriate FIPA-ACL message to the OSAg. The measurement units conversion service is also implemented as a Web service.

The service converts two types of measurement units: length and weight units. Their difference is strictly functional. Depending on the measure attribute submitted, the service performs the specified conversion.

From the WSDL file called `WeightConvertor.wsdl`, the plug-in generates the following files: `ConvertWeights.java`, `WeightUnit.java`, `ConvertWeightsLocator.java`, `ConvertWeightsSoap.java` and `ConvertWeightsSoapStub.java`.

The following code illustrates how to use the generated classes:

```
ConvertWeightsSoapStub stub = new ConvertWeightsSoapStub(new
URL("http://www.webservicex.net/ConvertWeight.asmx"),null);
double d = stub.convertWeight(value, new WeightUnit(fromUnit),new WeightUnit(toUnit));
```

A stub object of the class `ConvertWeightsSoapStub` is created on the client-side. The URL passed in the constructor is defined in the WSDL file. The units conversion ratio is obtained by invoking method `MeasureConvertor`. Since it is a conversion between two weight units, the `MeasureConvertor` method calls method `convertWeight`. The implemented Web service delivers the conversion ratio between the two weight units specified.

To invoke the length conversion service, an identical procedure is applied. From the WSDL file called `Length.wsdl`, the plug-in generates the following files: `Lengths.java`, `LengthUnit.java`, `LengthUnitLocator.java`, `LengthUnitSoap.java` and `LengthUnitSoapStub.java`.

The following code illustrates how to use the generated classes:

```
LengthUnitSoapStub stub = new LengthUnitSoapStub(new
URL("http://www.webservicex.net/length.asmx"),null);
double d = stub.changeLengthUnit(value, new Lengths(fromUnit),new Lengths(toUnit));
```

A stub object of the class `LengthUnitSoapStub` is created. The URL passed in the constructor is defined in the WSDL file. The units conversion ratio is obtained by invoking method `MeasureConvertor`. Since it is a conversion between two length units, the `MeasureConvertor` method calls method `changeLengthUnit`. The implemented Web service delivers the conversion ratio between two length units specified.

The implementation is identical to the currency conversion case. Figure 7.10 presents an example of the use of the measurement units conversion service. In this case, during the negotiation of a product, the CEAg uses "inches" to represent an attribute, while the SEAg uses "centimeters" to describe the same attribute.

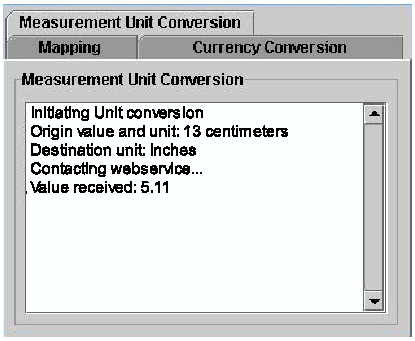


Figure 7.10: Measurement Units Conversion - inches *vs.* centimeters

7.3 Ontology Development

Each enterprise agent (Supplier or Customer) has its own private ontology. Since there are no ontologies defined for the automobile assembling domain, we created several automobile assembling ontologies[3] in order to test our services.

We built the ontologies using the Protégé ontology editor and stored them in OWL format. For testing purposes, some ontologies were created also using OntoEdit and stored in OXML. However, these ontologies defined in OXML were not used in the final experiments. The only difference resides in how to access the information, since the ontology is in a different format. The idea was to perform tests with agents that use not only different ontologies but also diverse ontology representation languages.

7.3.1 Web Ontology Language

Web Ontology Language (OWL) [Antoniou and van Harmelen, 2003] is a semantic markup language, developed as a vocabulary extension of RDF, for defining, publishing and sharing ontologies in the Web. OWL is written in XML and, therefore, inherits all the advantages of XML: the information can easily be exchanged between computers using different types of operating system and application languages. OWL differs from RDF since it enables greater machine interpretability of Web content by providing additional vocabulary along with formal semantics.

An OWL ontology mainly contains classes, properties, instances of classes and relationships between instances. Additionally, it defines a namespace declaration and an ontology header. Therefore, it can be used directly to publish and share sets of terms.

In Appendix F, we provide the structure of OWL documents and refer to the structures used in this work. A complete overview of OWL may be found in [Bechhofer *et al.*, 2004].

7.4 JADE

JADE is the agent development platform used to build our system. The main reason for choosing JADE lays on the fact that JADE implements the FIPA standards. JADE encloses implementations of the FIPA library of behaviours and interaction protocols. Furthermore, JADE comprehends the basic FIPA specifications which include FIPA-ACL, content languages, encoding schemes,

[3]One complete example of an OWL ontology in the automobile assembling domain can be found in Appendix E.

ontologies and transport protocols. Based on these specifications FIPA agents exist, operate and communicate.

JADE is a software framework fully implemented in Java which simplifies the implementation of a MAS through a middleware. Additionally, a set of graphical tools supports the debugging and deployment phases.

The agent platform can be distributed across machines that do not need to share the same operating system. Only one Java application and, therefore, only one Java Virtual Machine (JVM), is executed on each host. Each JVM is basically a container of agents that provides a complete runtime environment for agent execution and allows several agents to run concurrently on the same host.

The actual tasks that a JADE agent performs are typically carried out as "behaviours". Behaviours are created by extending the class `jade.core.behaviours.Behaviour`. To make an agent execute a certain task, an instance of the corresponding behaviour subclass has to be created and the `addBehaviour()` method of the `jade.core.Agent` class has to be called. One agent can implement and coordinate numerous behaviours in order to fulfil its goal.

A JADE agent is not a typical object-oriented programming object since its methods and attributes cannot be accessed directly. Instead, the JADE agent is message-driven, *i.e.*, reacts to requests.

By deafult, a thread is implemented per agent. Whenever an agent needs to execute concurrent tasks, JADE provides not only standard the multi-thread solution, which is supported directly by Java, but is also able to schedule the tasks of cooperative behaviours. However, behaviour scheduling rather than being preemptive is cooperative and everything occurs within a single Java thread.

The message transportation mechanism is adaptive, choosing the best available protocol depending on the underlying situation. For JADE platforms, communication relies on Java Remote Method Invocation (RMI). In case of different platforms, Internet Inter-ORB Protocol (IIOP) is used. IIOP is an object-oriented protocol that allows distributed programs written in different programming languages to communicate over the Internet. It is a critical part of the Common Object Request Broker Architecture (CORBA) specification. If necessary, more protocols can be added to JADE via interfaces.

Since version 1.3 (February 2000), JADE is an Open Source Software released under the General Public License (GNU) Lesser General Public License (LGPL). The version used in this work is 3.2, which was released on the 26th of July 2004. The minimal system requirement is version 1.4 of the Java Runtime Environment (JRE) or of the Java Development Kit (JDK). JADE ensures standard compliance with the FIPA specifications.

At start-up, JADE automatically activates some platform agents. This includes all mandatory components for managing the platform, *i.e.*, a naming service, a yellow pages service and a message transport and parsing service. These supporting agents are:

- The Agent Management System (AMS) provides the naming service, ensuring that each agent in the platform has a unique name. It also performs several "management tasks", *e.g.*, creating and killing agents on containers.

- The Director Facilitator (DF) provides a yellow pages service. The agents running on the platform register with DF. By querying the DF, the JADE agents can trace counterparts capable of supplying the services they need.

- The Agent Communication Channel (ACC) supports the communication between agents inside and outside the platform by listening to remote invocations. When it receives an FIPA-ACL message encoded as a string, which is usually the case of non-JADE agents, it parses the message and converts it into a Java `jade.lang.acl.ACLMessage` object used by all agents in the JADE platform.

Furthermore, JADE provides another "default agent" named Remote Monitoring Agent (RMA). This agent is not part of the FIPA model. The RMA offers a graphical interface to administrate the platform. It displays the state of the JADE agent platform where it resides, *e.g.*, lists all agents in the platform, and offers various tools for requesting administrative actions to the AMS agent, *e.g.*, alteration of the system configuration at run-time by moving agents from one machine to another.

FIPA/JADE ACL Messaging

In JADE, the agent communication is performed through message passing. FIPA-ACL is the language used to represent messages. The idea is to help ensure interoperability by providing a standard set of FIPA-ACL message structures as well as a well-defined process for maintaining this set[4].

According to the FIPA-ACL language format, a FIPA-ACL message consists of intention, attendees, content of the message, content description and conversation control:

[4]FIPA Communicative Act Library Specification, SC00037J, 12/03/2002. http://www.fipa.org/specs/fipa00037/SC00037J.html, April, 2005

- Intention is the type of the communicative act. The corresponding parameters are called performatives. They are linguistically-motivated and domain independent, corresponding to semantically-distinct message categories. Examples of FIPA-ACL performatives are REQUEST, INFORM, QUERY_IF.

- Attendees are the participants in a communication, *i.e.*, the sender and the set of receivers. Contributing parameters are "sender", "receiver" and "reply-to".

- Content of a message is the actual information that is exchanged, the parameter is "content".

- Content description contains an indication of the content language used to express the content ("language" parameter), the encoding of the content language expression ("encoding" parameter) and the content ontology that ascribes meanings to the symbols in the content expression for both sender and receiver(s) ("ontology" parameter).

- Conversation control includes the parameters "protocol", "conversation-id", "reply-with", "in-reply-to" and "reply-by". The "protocol" is the interaction protocol that the sending agent is employing with this FIPA-ACL message; "conversation-id" identifies the ongoing sequence of communicative acts that form a conversation; "reply-with" is used by the responding agent to identify this message; "in-reply-to" references an earlier action to which this message is a reply; and "reply-by" contains the latest time by which a sending agent accepts a reply.

Similarly to the KQML, FIPA-ACL is based on the speech act theory. A speaker "utters" speech acts, which are also known as performatives or as communicative acts. One agent takes the role of the initiator and starts conversation, while the receiver(s) of the message respond. Speech acts may be understood in terms of an intentional level description of an agent, which makes references to the Belief-Desire-Intention (BDI) model.

An example of a FIPA-ACL message:

```
(INFORM
:sender (agent-identifier
:name supplier@sutton:1099JADE
:addresses (sequence http:/sutton:7778acc))
:receiver (set (agent-identifier
:name customer@sutton:1099JADE
:addresses (sequence http:/sutton:7778acc)))
:content "((done (action
(agent-identifier :name supplier@sutton:1099JADE
:addresses (sequence http:/sutton:7778acc))
(Sell :quantity 1 :delivery_time 5))))"
:reply-with customer@sutton:1099JADE1107949238914
:in-reply-to R11904736_0
:language fipa-sl
:ontology eCommerce
:protocol fipa-contract-net
:conversation-id 1107949210593)
```

In this example, the supplier agent informs the customer agent about some content. The content is expressed in the FIPA-SL and the e-commerce ontology is used. Furthermore, a "conversation-id" number identifies the ongoing sequence of communicative acts that together form a conversation, "protocol" denotes the interaction protocol that the sending agent is employing with this FIPA-ACL message, the parameter "reply-with" is used by the responding agent to identify this message and "in-reply-to" refers to an earlier action to which this message is a reply.

Content Language SL

Content languages specify the syntax of messages and are domain independent. The JADE toolkit already implements the FIPA-SL[5], which is a human-readable, string-encoded language. Being human-readable is an advantage when debugging, testing and reproducing the message flow of an application. The syntax and its associated semantics are suggested as a candidate content language for use in conjunction with the FIPA-ACL.

For the semantics of content expressions, a content ontology has to be expressed in the ontology parameter of the message.

[5]SL Content Language Specification, SC000081, 12/03/2002, http://www.fipa.org/specs/fipa00008/SC00008I.html, April, 2005

7.5 BeanGenerator

The BeanGenerator associates each schema in an ontology with a Java class or interface. The sum of all created objects that implement the interfaces `jade.content.Concept`, `jade.content.Predicate` or `jade.content.AgentAction` represent the ontology. Furthermore, a "common" class defining the vocabulary of all classes, registering "Predicates", "Concepts" and "AgentActions", storing name mappings, *etc.*, is created.

The following code is generated by the BeanGenerator tool. It is the correspondent Java class for the "Concept" "Wheel_Rim".

```
/**
* the outer part of a wheel to which the tire is attached
* Protege name: Wheel_Rim
* @author ontology bean generator
*/
public class Wheel_Rim extends Automobile_Part
/**
* Protege name: manufacturer
*/
private String manufacturer;
public void setManufacturer(String value)
this.manufacturer=value;
public String getManufacturer()
return this.manufacturer;
private float diameter;
public void setDiameter(float value)
this.diameter=value;
public float getDiameter()
return this.diameter;
/**
* Protege name: wheel_rim_material
*/
private String wheel_rim_material;
public void setWheel_rim_material(String value)
this.wheel_rim_material=value;
public String getWheel_rim_material()
return this.wheel_rim_material;
```

Like all classes generated by the BeanGenerator plug-in, this class consist of the attributes specified in the Protégé ontology editor and the correspondent set-/get-methods. Since "Wheel_Rim" is an "Automobile_Part", it extends this class. "Automobile_Part", in turn, to comply with the CRM, implements the interface `jade.content.Concept`. In order to be used by JADE agents, the generated ontology files have to be imported into the agent project and the ontology has to be registered inside the agent's code. Using the Java keyword `import`,

the classes are bound to the agent. In other words, the ontology is hard-coded into the agent and the imported ontological classes have to be available for troubleshooting messaging.

7.6 JENA

JENA is an open source Java framework for writing Semantic Web applications. Initially developed at the HP Labs for Semantic Web Research, it provides a programmatic environment for RDF, RDF(S) and OWL, and includes a rule-based inference engine. Among other features it contains an RDF/XML parser and the Jena2 ontology API. It is intended to work with ontology data bases in RDF, which includes supporting OWL and RDF(S). Using the above languages, a set of Java abstractions extends the generic "RDF Resource" and "Property" classes to model directly the class and property expressions found in the ontologies as well as the relationships between these classes and properties. In other words, the Jena2 ontology API allows navigating within the structure of the ontology file and accessing the stored information.

Suppose an RDF model that is represented as a set of statements. The JENA model interface defines a method called `listStatements()` which returns all statements of a model in a `com.hp.hpl.jena.rdf.model.StmtIterator`. Furthermore, the `com.hp.hpl.jena.rdf.model.Statement` interface provides methods to access the subject, predicate and object of the statements.

```
StmtIterator iter = model.listStatements();
while (iter.hasNext())
Statement stmt = iter.nextStatement(); / get next statement
Resource subject = stmt.getSubject();  // get the subject
Property predicate = stmt.getPredicate(); / get the predicate
RDFNode object = stmt.getObject();  // get the object
```

Since the object of a statement can either be a resource or a literal, the `getObject()` method returns an object typed as `com.hp.hpl.jena.rdf.model.RDFNode`, which is a common superclass of both `com.hp.hpl.jena.rdf.model.Resource` and `com.hp.hpl.jena.rdf.model.Literal`. Casting has to be used to detect the type of the subclass.

In the developed MAS, the JENA model interface is used to get attributes, instances and values of a class (such as the price of items), to determine subclass-relationships and to check if a class is abstract.

Since ontologies changes are inevitable, the Protégé tool is integrated in the interface. This allows the user to define new classes, instances of classes and

values in the existing ontologies. After editing the ontology, the ontology file is stored in the same location again, so that the correspondent agent detects and considers the new changes.

Compared to the hard-coding ontology approach based in Java objects, this methodology has the advantage of being more flexible as far as ontology updates are concerned. Updates can be easily made using Protégé and then submitted to the parser. Otherwise, changing an ontology is an intricate process: the ontology has to be edited in Protégé, the Java sources have to be generated with the BeanGenerator tool, compiled and the agent's code altered.

Since homogeneity in the domain ontologies is not enforced, semantic and syntactic interoperability problems consequently occur.

7.7 Conclusions

In this Chapter we illustrate the methodologies described in Chapter 6. Most of the implemented techniques are services and do not require user interaction. The user intervenes only to start the process, registering the agent in the platform and, in the case of the customer enterprise agents, announcing a request for a product.

We presented some of the GUI implemented to illustrate the use of the ontology-based services. The experiments described demonstrate how the different services are invoked and the type of results obtained.

We show that the results of the MTS depend on the quantity and the quality of the information represented in the ontology. Additionally, similarity evaluations among two ontologies can only be achieved if both knowledge representations share some components, *i.e.*, it is impossible to match two ontologies if the name, characteristics, description and relations of the concepts under analysis are totally unrelated.

To choose an appropriate similarity matching algorithm we evaluated the Resnik, Jiang-Conrath, Leacock-Chodorow, Hirst-St-Onge and Wu-Palmer WordNet-based similarity measure algorithms. After comparing several concepts in the automobile domain, the LCH provided the best results. We also concluded that it is necessary to perform a large number of comparisons between concepts from different ontologies to establish which is the best algorithm for a given case. Alternatively, we could try to detect some behaviour patterns that would allow the OSAg to choose automatically the best algorithm.

In order to increase the performance of the similarity matching between concepts, the similarity results for each pair of compared concepts are stored and kept for the future negotiation rounds. Although this approach increased the

performance of the matching process, the MTS was still slow. To overcome this inefficiency we implemented a basic learning mechanism that can be seen as a combination of rote-learning and Case-Based Reasoning (CBR). We claim that it is rote learning because it memorises knowledge, avoiding repeating the same computations in the future. In addition, as in the CBR approach, we use specific knowledge from previously experienced situations (negotiations) to solve similar new problems by finding identical past cases and reusing them in a new problem situation. Consequently, we increase the performance of the MTS whenever the agents involved have already participate in some former negotiation round or when the product has been previously compared. As the number of negotiation rounds grows, the quantity of the memorised knowledge augments and the performance of the service increases.

We are applying a weighted average to combine the individual similarity results into a final similarity matching result. The weights used were established *via* experimentation, which showed that the most trustworthy results are the ones from the semantic comparison of concepts. This is caused by the fact that the semantic comparison is based on the lexicon database WordNet. The weights that are applied vary according to the results available so far.

All the services presented in this Chapter were implemented and tested when agents are negotiating items in the same domain and using the same language (English). The OSAg attempts to solve the detected interoperability problem but cannot, beforehand, guarantee whether its effort will be successful or not.

The majority of the experiences reported in the literature involving the same type of interoperability issues are applied to toy problems. We were unable to find results regarding experiments conducted with ontology matching and ontology-based information integration in real world applications. The real-world ontologies available on the Web and the reported experiences involving real-world data sources are not applicable to our domain.

The UCS are implemented as Web services. The enterprise agents (customer or supplier) request these services by sending an appropriate FIPA-ACL message to the OSAg. Some illustrative examples were presented.

Protégé, with the help of JENA and the Beangenerator plug-in, was successfully integrated with the JADE development platform. This enriched JADE platform allows the seamless creation, maintenance and update of multiple domain ontologies.

Chapter 8

Conclusions and Future Work

This Chapter summarises and presents the main conclusions of the research work described throughout this thesis. We identify the original contributions as well as the limitations, and propose future research directions that may emerge from this work.

8.1 Summary of this Thesis

The agent technology roadmap [AgentLink], written by the AgentLink III network of excellence researchers, identifies as one of the key problem areas the development of infrastructures for open agent communities. Electronic institutions (EI), together with ontologies and related services, address the development of such types of infrastructures. Within this framework, we propose the creation of a MAS to facilitate the formation of virtual enterprises (VE). In B2B e-commerce, the formation of the VE is an essential preliminary step where autonomous agents from different origins are expected to negotiate their future binding business contracts. Our work is motivated by the need to develop services to assist the coordination efforts between these autonomous agents that, although representing different real-world enterprises and using different ontologies, must interact in order to establish new business relationships.

The problem of handling information from agents that use different ontologies has only recently been addressed. To fully understand the exchanged utterances, agents cooperating in a MAS must share a common ontology. Since our MAS is an open system composed of heterogeneous, autonomous agents, it is impossible to define beforehand a common domain knowledge ontology. Moreover, in a B2B

domain, ontologies are likely to contain strategic business knowledge that not only should remain private, but also is expected to evolve with time.

The majority of the approaches that address the interoperability problems in MAS found in the literature neither adopts a common ontology language nor uses ontology editors to build ontologies. The ontology languages differ and no standard ontology languages are defined - usually each system uses its own ontology language. Furthermore, there are few approaches concerning agents using different ontologies in the B2B domain.

In a computational B2B context, as the one presented in this thesis, the heterogeneity among supplier and customer enterprise agents generates inevitably interoperability problems. Each supplier and customer may use its own formalism, concepts and characteristics to represent the same products. And, even when both supplier and customer agents use the same formalism to represent their ontologies, their content may differ significantly either syntactically or semantically. Whenever different ontologies are used, the different representations and terminologies prevail unless a formal mapping between the ontologies is established.

Simultaneously, proposals regarding standards and joint initiatives for the classification of products are proliferating. However, large and consensual knowledge models for e-commerce applications are difficult and expensive to build. Several e-commerce ontologies have been proposed in the last years to ease the information exchange between customer and supplier agents. The main problem is that, when companies represent products, they tend to adopt the terminology they use in their private catalogues or ontologies and do not invest in coding products according to some external standard. Based on this premise, we created several ontology-based services which just take into account the products' information and disregard any standard coding classification.

Our MAS models the activity that precedes the actual formation of a VE. In this stage, the enterprise agents are eager to negotiate. To support the negotiation of products, we provide an E-Commerce Ontology which is shared by all agents and defines a domain-independent business vocabulary. This ontology ensures a meaningful e-commerce communication since all agents will uniformly interpret the intention of the messages exchanged and the generic business terms used. Besides this E-Commerce Ontology, each agent has its own private car assembling domain ontology built with Protégé development tool and stored in OWL format. Since in a real-world business context, the enterprises will use different ontologies and adopt diverse ontology representation formalisms, our approach also envisages this possibility: the ontologies may be developed by any ontology development tool and stored in any standard format.

Although the main interoperability problems are caused by the use of different knowledge representations, interoperability problems may also occur when agents are negotiating using different units.

8.2 Contributions

We have implemented an Ontology-based Services Agent (OSAg), which is responsible for providing ontological services to the enterprise agents, in order to solve the identified interoperability problems.

The OSAg provides:

- **Units Conversion Services** (UCS) through Web services interfaces to all enterprise agents that need to convert between different currency or measurement units.
- **Matching Terms Service** (MTS) that assesses the semantic and syntactic similarity between concepts from two different ontologies.

The algorithms implemented compare the names, attributes, relations and descriptions of the pair of concepts under scrutiny. The comparison between the names is based on the Leacock-Chodorow (LCH) WordNet-based semantic similarity measure algorithm. The syntactic analysis is performed over the attributes, relations and descriptions. The attributes are grouped by data type (*integer, string, float and boolean*) and then compared using the n-grams algorithm. The relation *has-part* is also compared using also the n-grams algorithm. The attributes and relations (AR) comparison results are combined into a single result. Finally, the descriptions (D), which are also submitted to the n-grams algorithm, are first pre-processed before being compared in order to obtain comprehensive rather than verbose representations. At the end of the similarity matching process, there are up to three individual similarity values (AR, D and LCH).

To establish a final similarity value, a weighted average is performed. The weights that are applied vary and depend on the results obtained so far by the MTS. The results range between zero and one, where zero means that the two concepts are totally unrelated and one that they are identical.

The performance of the OSAg was enhanced with a Basic Learning Mechanism (BLM) that memorises the concepts already compared and matched. This avoids repeating the similarity matching process for already compared pairs of concepts.

Since JADE, which is our agent development platform, does not allow the creation of agents using different ontologies, we adopted a new methodology that

combines the CRM of JADE and OWL ontologies. Our methodology was integrated into the JADE environment and provides automatic support to the creation of MAS composed of agents with different ontologies. Furthermore, it also supports ontology maintenance and updating.

Finally, we combine two communication protocols: the FIPA Contract Net Protocol (FIPA-CNP) and the Ontology Interaction Protocol (OIP). The OIP was specifically developed to support the resolution of the interoperability problems that arise during the agents interaction.

A test-bed was implemented to demonstrate and validate our approach.

8.3 Limitations

The MTS is intended for agents that negotiate in the same domain and use the same language (English). The service was neither tested in a multiple domains scenario nor with different languages.

The efficiency of the MTS depends on the quality of the information represented in the ontologies and also on the completeness of the WordNet repository.

The ontologies are created taking into consideration a limited set of data types and relations. In order to extract the information and compare the characteristics between two concepts, we defined a set of data types and relations that must be used in the ontology construction.

The UCS are implemented as Web services and, as a result, depend of the Web service availability.

Since the OSAg functionalities are implemented as services, they only execute when solicited/asked, *i.e.*, the OSAg reacts to requests made by the enterprise agents. Alternatively, the OSAg could be monitoring all negotiation rounds in order to identify forthcoming interoperability problems, suggest possible solutions as well as learning the models of the enterprise agents.

8.4 Future Work

The future work is closely related with the limitations presented in the previous Section.

The ontologies created were based on real information. However, it would be interesting to observe the performance of the matching methods in a real-world scenario. We obtained promising results with ontologies based on real information in the automobile assembly domain, but we also would like to test our approach in other application domains.

We plan to enhance the OSAg with new capabilities so that it monitors the communication and negotiation between agents in order to suggest potential solutions without the explicit request of the enterprise agents.

A mapping between different data types and different relations also needs to be addressed. The OSAg could learn from data types and relations never used before.

An important enhancement would be to automatically perform experiments with the different WordNet-based similarity algorithms in the current scenario. Based on the analysis of the obtained results, the OSAg would choose the best algorithm to apply.

The implementation of a feedback mechanism, where the enterprise agents report the accurary of the suggested concept matches back to the OSAg, would provide additional information to improve the accuracy of the services.

Another possible improvement would be to create a new ontology based on the information exchanged. This new ontology could then be used by enterprises that do not have an ontology and also need to use the MTS.

We would also like to integrate a measurement units ontology to enhance the measurement units conversion service, namely, when different unit names are used.

Finally, we would like to integrate the proposed services in the Electronic Institution MAS under development at LIACC, which includes negotiation mediation, contract validation/registration and contract monitoring and enforcement.

Bibliography

[AgentLink] AgentLink III, *Agent Technology Roadmap: Overview and Consultation Report*. http://www.agentlink.org/roadmap/index.html, December 2004.

[Antoniou and van Harmelen, 2003] Antoniou, G., van Harmelen, F. *Web Ontology Language: OWL*. In Staab, S., Studer, R. (eds). Handbook on Ontologies in Information Systems, Springer-Verlag, 2003.

[Arpírez *et al.*, 2001] Arpírez, J. C., Corcho, O., Fernández-López, M., Gómez-Pérez, A. *WebODE: a scalable ontological engineering workbench*. In: Gil, Y., Musen, M., Shavlik, J. (eds.) First International Conference on Knowledge Capture (KCAP'01), Victoria, Canada, ACM Press, New York, pp. 6-13, 2001.

[Baader *et al.*, 2003] Baader, F., McGuinness, D., Nardi, D., Patel-Scheider, P. *The Description Logic Handbook: Theory, implementation and application*. Cambridge University Press, Cambridge, United Kingdom. 2003

[Baeza-Yates and Ribeiro-Neto, 1999] Baeza-Yates, R.A., Ribeiro-Neto, B.*Modern Information Retrieval*, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1999.

[Bailin and Truszkowski, 2002] Bailin, S. C., Truszkowski, W. *Ontology negotiation between intelligent information agents*. The Knowledge Engineering Review, Vol.17(1), pp. 7-19. 2002.

[Barnes-Vieyra and Claycomb, 2001] Barners-Vieyra, P., Claycomb, C. *Business-to-Business E-Commerce: Models and Managerial Decisions. Business Horizons*, pp. 13-20, 2001.

[Bechhofer *et al.*, 2001] Bechhofer, S., Horrocks, I., Goble, C., Stevens, R. *OilEd: a Reason-able Ontology Editor for the Semantic Web*, In Proceedings of 14th International Workshop on Description Logics, Stanford, USA, August, 2001.

[Bechhofer *et al.*, 2004] Bechhofer, S., van Harmelen, F., Hendler, J. Horrocks, I., McGuinness, D.L., Patel-Schneider, P.F., Stein, L.A. *OWL Web Ontology Language Reference*. In: Dean, M., Schreiber, G. (eds.) W3C Recommendation. 10 February 2004. http://www.w3.org/TR/2004/REC-owl-ref-20040210

[Bergamaschi *et al.*, 1999] Bergamaschi, S., Castano, S., Vincini, M. *Semantic Integration of Semistructured and Structured Data Sources*, SIGMOD Record, Vol.28(1):54-59, 1999.

[Borst, 1997] Borst, W. N. *Construction of Engineering Ontologies*. University of Tweenty. Enschede, The Netherlands - Centre for Telematica and Information Technology, 1997.

[Bray *et al.*, 2004] Bray, T., Paoli, J., Sperberg-McQueen, C. M., Maler, E., Yergeau, F. *Extensible Markup Language (XML)* 1.0 (Third Edition), W3C Recommendation, February, 2004. http://www.w3.org/TR/REC-xml

[Brickley and Guha, 2002] Brickley D., Guha R. V. *RDF Vocabulary Description Language 1.0: RDF Schema*, W3C Working Draft, 2002. http://www.w3.org/TR/PR-rdf-schema

[Bruijn *et al.*, 2004] Bruijn, J., Martín-Recuerda, F., Manov, D., Ehrig, M. *D4.2.1 State-of-the-art survey on Ontology Merging and Aligning* V1. EU-IST Integrated Project (IP) IST-2003-506826 SEKT, Digital Enterprise Research Institute, University of Innsbruck, 2004.

[Budanitsky, 1999] Budanitsky, A., Hirst, G. *Lexical Semantic Relatedness and Its Application in Natural Language Processing*. Technical Report CSRG-390, Computer Systems Research Group, University of Toronto, August, 1999.

[Budanitsky and Hirst, 2005] Budanitsky, A., Hirst, G. *Evaluating WordNet-based Measures of Lexical Semantic Relatedness. Association for Computational Linguistics*. Vol.32(1), 2005. http://ftp.cs.toronto.edu/pub/gh/Budanitsky+Hirst-2006.pdf

[Burnstein *et al.*, 2003] Burnstein, M., McDermott, D., Smith, D.R., Westfold, S.J., *Derivation of glue code for agent interoperation.* Autonomous Agents and Multi-Agent Systems, Vol.6(3):265-286, 2003.

[Caire and Cabanillas, 2004] Caire, G., Cabanillas, D. *JADE Tutorial – Application-defined content languages and ontologies*, Technical Report, TILab, 2004.

[Camarinha-Matos and Afsarmanesh, 1999] Camarinha-Matos, L. M., Afsarmanesh, H., *The Virtual Enterprise Concept.* In: Camarinha-Matos, L., Afsarmanesh, H. (eds.) Infrastructures for Virtual Enterprises: Networking Industrial Enterprises, Kluwer Academic Publishers, pp. 3-14, 1999.

[Cardoso and Oliveira, 2005] Cardoso, H. L., Oliveira, E. *Virtual Enterprise Normative Framework within Electronic Institutions*, In: Gleizes, M-P., A. Omicini, A., Zambonelli, F. (eds.), Engineering Societies in the Agents World V, LNAI 3451, Springer, pp.14-32, 2005.

[Cardoso *et al.*, 2005] Cardoso, H. L., Malucelli, A., Rocha, A.P., Oliveira, E. *Institutional Services for Dynamic Virtual Organizations.* In: Camarinha-Matos, L. M., Afsarmanesh, H., Ortiz, A. (eds.) Collaborative Networks and Their Breeding Environments - 6th IFIP Working Conference on Virtual Enterprises, Springer, pp. 521-528, 2005.

[Castano *et al.*, 2000] Castano, S., De Antonellis, V., De Capitán di Vimercati, S., *Global viewing of heterogeneous data sources*, IEEE Transactions on Knowledge and Data Engineering, 2000.

[Cologne Institute, 2005] Cologne Institute for Business Research Consult Ltd. *The leading classification system*, White Paper, June, 2005. http://www.eclass-online.com

[Corcho and Gómez-Pérez, 2001] Corcho, O., Gómez-Pérez, A. *Solving Integration Problems of E-Commerce Standards and Initiatives through Ontological Mappings*, In Proceedings of the Workshop on E-Business & the Intelligent Web, IJCAI, Seattle, USA, 2001.

[Cranefield and Purvis, 1999] Cranefield, S. and Purvis, M. *UML as an ontology modelling language*, In Proceedings of the

Workshop on Intelligent Information Integration, 16th International Joint Conference on Artificial Intelligence (IJCAI-99), 1999. http://hcs.science.uva.nl/usr/richard/workshops/ijcai99/UML_Ontology_Modelling.pdf

[Damashek, 1995] Damashek, M. *Gauging Similarity via N-Grams: Language-independent Sorting, Categorization, and Retrieval of Text*, Science 267, pp. 843-848, 1995.

[Decker *et al.*, 1997] Decker, K., Sycara, K., Williamson, M. *Middle-agents for the internet*, In Proceedings of the 15th International Joint Conference on Artificial Intelligence, Nagoya, Japan, 1997.

[Dignum, 2001] Dignum, F. *Agents, markets, institutions and protocols.* In: Dignum, F., Sierra, C. (eds.), Agent Mediated Electronic Commerce: The European Agentlink perspective, LNAI 1991, Springer, pp. 98-114, 2001.

[Dignum and Dignum, 2001] Dignum, V., Dignum, F. *Modelling agent societies: co-ordination frameworks and institutions.* In: Brazdil, P., Jorge, A. (eds.) Progress in Artificial Intelligence: Knowledge Extraction, Multi-agent Systems, Logic Programming, and Constraint Solving, LNAI 2258, Springer, pp. 191-204, 2001.

[Do and Rahm, 2002] Do, H-H, Rahm, E. *Coma - a system for flexible combination of schema matching approaches*, In Proceedings VLDB, pp. 610-621, 2002.

[Doan, 2002] Doan, A-H. *Learning to map between structured representations of data*, PhD thesis, University of Washington, Seattle, US, 2002. http://anhai.cs.uiuc.edu/home/thesis.html

[Doherty *et al.*, 2005] Doherty, P., Szalas, A. Lukaszewicz, W. *Approximative Query Techniques for Agents with Heterogeneous Ontologies and Perceptive Capabilities*, In Proceedings of the 9th International Conference on Principles of Knowledge Representation and Reasoning, 2004.

[Domingue, 1998] Domingue, J. *Tadzebao and WebOnto: Discussing, Browsing, and Editing Ontologies on the Web.* In: Gaines, B.R, Musen, M.A. (eds.) 11th International Workshop on Knowledge Management Support, IEEE Intelligent

Systems & their applications, Vol.15(3):26-32, 1998.

[Domingue *et al.*, 1999] Domingue, J., Motta, E., Corcho Garcia, O. *Knowledge Modelling in WebOnto and OCML: A User Guide.* Version 2.4. Knowledge Media Institute. The Open University. 1999. http://kmi.open.ac.uk/projects/ocml/ocml-webonto-guide.zip

[Dou *et al.*, 2003] Dou, D., Mcdermott, D., Qi, P. *Ontology Translation on the Semantic Web.* Proceedings of International Conference on Ontologies, Databases and Applications of Semantics (ODBASE 2003), LNCS 2888, Springer-Verlag. Berlin Heidelberg, pp. 952-969, 2003.

[Duineveld *et al.*, 1999] Duineveld, A. J., Stoter, R., Weiden, M. R., Kenepa, B., Benjamins, V. R. *WonderTools ? A comparative study of ontological engineering tools*, In Proceedings of the 12th Banff Knowledge Acquisition for Knowledge-Based Systems Workshop. 1999.

[Ehrig and Staab, 2004] Ehrig, M., Staab, S. *QOM - Quick Ontology Mapping*, In: Proceedings of the 3rd International Semantic Web Conference (ISWC2004), Hiroshima, Japan. LNCS, Springer, 2004.

[Ehrig and Sure, 2004] Ehrig, M., Sure, Y., *Ontology mapping - an integrated approach*, In Proceedings of the European Semantic Web Conference (ESWS), pp. 76-91, 2004.

[Euzenat and Valtchev, 2003] Euzenat, J., Valtchev, P. *An integrative proximity measure for ontology alignment*, In Proceedings of the Workshop on semantic information integration, Sanibel Island, US, pp. 33-38, 2003.

[Euzenat *et al.*, 2004] Euzenat, J., Bach, T.L., Barrasa, J., Bouquet, P., Bo, J.D., Dieng, R., Ehrig, M., Hauswirth, M., Jarrar, M., Lara, R., Maynard, D., Napoli, A. Stamou, G., Stuckenschmidt, H., Shvaiko, P., Tessaris, S., Van Acker, S., Zaihrayeu, I. *D.2.2.3: State of the art on ontology alignment*, Knowledge Web, August, 2004. http://dit.unitn.it/ p2p/RelatedWork/Matching/kweb-223.pdf.

[Falasconi *et al.*, 1996] Falasconi, S., Lanzola, G., Stefanelli, M. *Using ontologies in multi-agent systems.* In Proceedings of Tenth Knowledge Acquisition for Knowledge-Based Systems Workshop (KAW), University of Calgary, Banff,

Alberta, Canada, 1996.

[Farquhar *et al.*, 1997] Farquhar A., Fikes R., Rice J. *The Ontolingua Server: A Tool for Collaborative Ontology Construction*, International Journal of Human Computer Studies, Vol.46(6):707-727, 1997.

[Fensel *et al.*, 2000] Fensel, D., Horrocks, I., van Harmelen, F., Decker, S., Erdmann, M., Klein, M. *OIL in a nutshell.* In: Dieng, R., *et al.* (eds.) Knowledge Acquisition, Modeling, and Management, In Proceedings of the European Knowledge Acquisition Conference (EKAW-2000), LNAI, Springer-Verlag, 2000.

[Fensel, 2001] Fensel, D. *Ontologies and Electronic Commerce.* IEEE Intelligent Systems, January/February, 8, 2001.

[Fensel, 2004] Fensel, D. *Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce*, Second Edition, Springer, 2004.

[Ferber, 1999] Gerber, J. *Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence*, Addison-Wesley Longman, New York, 1999.

[Fisher *et al.*, 1996] Fischer, K., Muller, J., Heimig, I., Scheer, A. *Intelligent Agents in Virtual Enterprises*, Proceedings of the 1th International Conference on the Pratical Application of Intelligent Agents and Multi-Agent Technology, London, UK. 1996.

[Foss, 1999] Foss, J. *Brokering Automated Enterprises.* In Proceedings of the 9th Annual Conference of the Internet Society, INET'99, San Jose, California, USA, June, 1999. http://www.isoc.org/inet99/proceedings/1d/1d_3.htm

[Genesereth and Fikes, 1992] Genesereth, M. R., Fikes, R. E. *Knowledge Interchange Format*, Version 3.0, Reference Manual, Technical Report 92-1. Computer Science Department. Stanford University, California, 1992. http://logic.stanford.edu/kif/Hypertext/kif-manual.html

[Giunchiglia *et al.*, 2004] Giunchiglia, F., Shvaiko, P., Yatskevich, M. *S-Match: an algorithm and an implementation of semantic matching*, In Proceedings of ESWS 2004, Heraklio, GR, pp. 61-75, 2004.

[Gómez-Pérez *et al.*, 2004] Gómes-Pérez, A., Fernández-López, M., Corcho, O. *Ontological Engineering: with examples from the areas of knowledge management, e-commerce and the semanctic web*, Springer-Verlag London, 2004.

[Goodchild *et al.*, 2000] Goodchild, A., Herring, C., Milosevic, Z. *Business contracts for B2B*, In Ludwig, H., Hoffner, Y., Bussler, C., Bicher, M. (eds), Workshop on Infrastructure for Dynamic Business-to-Business Service Outsourcing (ISDO'00), CEUR Workshop Proceedings, pp. 63-74, 2000.

[Granada Research, 2001] Granada Research. *Using the UNSPSC: why coding and classifying products is critical to success in electronic commerce*, White Paper, October, 2001. http://www.unspsc.org/documentation.asp

[Gruber, 1993] Gruber, T. R. *Toward Principles for Design of Ontologies Used for Knowledge Sharing*, In: Guarino, N. and Poli, R. (eds.) Formal Ontology in Conceptual Analysis and Knowledge Representation, Kluwer Academic Publishers, 1993.

[Gruber, 1993a] Gruber, T. R. *A Translation approach to portable ontology specification*, Knowledge Acquisition, Vol.5(2):199-200, 1993.

[Grüninger and Fox, 1995] Grüninger, M., Fox, M. S. *Methodology for the design and evaluation of ontologies*, In: IJCAI95 Workshop on Basic Ontological Issues in Knowledge Sharing, Montreal, Canada.

[Guarino, 1998] Guarino, N. *Formal ontology in information systems*, IOS press, Amsterdan, NL, 1998.

[Guth, 1976] Guth, G. J. A. *Surname Spellings and Computerized Record Linkage*. Historical Methods Newsletter, Vol.10(1):10-19, 1976.

[Hakimpour and Geppert, 2001] Hakimpour, F., Geppert, A. *Resolving Semantic Heterogeneity in Schema Integration: an Ontology Based Approach*, In the Proceedings of the International Conference on Formal Ontology in Information Systems (FOIS 2001) Maine, USA, pp. 297-308, 2001.

[Hameed *et al.*, 2001] Hameed, A., Sleeman, D. Preece. A. D*etecting Mismatches Among Experts' Ontologies Acquired through Knowledge Elicitation*, Research and Development in Intelligent Systems, BCS Conference Series, Springer, pp.

9-22, 2001.

[He *et al.*, 2003] He, M., Jennings, N.R., Leung. H.-F. *On Agent-Mediated Electronic Commerce*, IEEE Transactions on Knowledge and Data Engineering, Vol.15(4):985-1003, 2003.

[Horrocks *et al.*, 2002] Horrocks, I., Patel-Schneider, P. F., van Harmelen, F. *Reviewing the design of DAML+OIL: An ontology language for the semantic web*, In Proceedings of the 18th National Conference on Artificial Intelligence, pp. 792-797, AAAI Press, 2002.

[Hugns, 2004] Hugns, M. *Software Development with Objects, Agents, and Services.* In Proceedings of the Third International Workshop on Agent-Oriented Methodologies, Vancouver, Canada, October, 2004.

[Huhns and Singh, 1997] Huhns, M. N., Singh, M. P. *Readings in Agents.* Morgan Kaufmann Publishers, INC. San Francisco, California, 1997.

[Jiang and Conrath, 1997] Jiang, J. J., Conrath, D. W. *Semantic Similarity Based on Corpus Statistics and Lexical Taxonomy*, In Proceedings of International Conference Research on Computational Linguistics, Taiwan, 1997.

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

[Kalfoglou and Schorlemmer, 2003a] Kalfoglou, Y., Schorlemmer, M. *If-map: an ontology mapping method based on information flow theory*, Journal of data semantics, 1:98-127, 2003.

[Karp *et al.*, 1999] Karp, P. D., Chaudhri, V. K., Thomere, J. *XOL: An XML-based ontology exchange language*, Version 0.4, 1999. http://www.ai.sri.com/pkarp/xol/xol.html.

[Kavouras, 2003] Kavouras, M. *A Unified Ontological Framework for Semantic Integration*, In Proceeding of the International Workshop on Next Generation Geospatial Information, Cambridge (Boston), Massachusetts, USA, October, pp. 19-21, 2003.

[Kifer *et al.*, 1995] Kifer, M., Lausen, G, Wu, J. *Logical Foundations of Object-Oriented and Frame-based Languages.* Journal of the ACM, Vol.42(4):741-843, 1995.

[Klein, 2001] Klein, M. *Combining and relating ontologies: an analysis of problems and solutions*, In Proceedings of Workshop on Ontologies and Information Sharing, IJCAI'01, Seattle, USA, 2001.

[Klein *et al.*, 2002] Klein, M., Kiryakov, A., Ognyanoff, D. Fensel, D. *Finding and specifying relations between ontology versions.* In Proceedings of the ECAI-02 Workshop on Ontologies and Semantic Interoperability. Lyon, July, 2002.

[Kozima and Furugori, 1993] Kozima, H., Furugori, T. *Similatiry between words computed by spreading activation on an English dictionary.* In Proceedings of the 6th Conference of the European Chapter of the Association for Computational Linguistics. Utrecht, The Netherlands, pp. 232-239, 1993.

[Lassila and McGuinness, 2001] Lassila, O., McGuinness, D. L. *The Role of Frame-Based Representation on the Semantic Web*, Knowledge Systems Laboratory, Report KSL-01-02, January, 2001.

[Lassila and Swick, 1999] Lassila, O., Swick, R. *Resource Description Framework (RDF) Model and Syntax Specification*, W3C Recommendation, 1999. http://www.w3.org/TR/REC-rdf-syntax

[Lenat and Guha, 1990] Lenat, D. B., Guha, R. V. *Building Large Knowledge-based Systems: Representation and Inference in the Cyc Project.* Menlo Park, CA: Addison-Wesley, 1990.

[Levenshtein, 1966] Levenshtein, I. V. *Binary Codes capable of correcting deletions, insertions, and reversals.* Cybernetcs and Control Theory, Vol.10(8):707-710, 1966.

[Lomuscio *et al.*, 2000] Lomuscio, A., Wooldridge, M., Jennings, N.R. *Automated negotiation in agent-mediated electronic commerce.* Book on European perspectives on AMEC, 2000. http://www.iiia.csic.es/AMEC/BOOK/Book.html

[Luke and Heflin, 2000] Luke, S., Heflin, J. D., *SHOE 1.01. Proposed Specification*, Technical Report, Parallel Understanding Systems

Group, Department of Computer Science, University of Maryland, 2000. http://www.cs.umd.edu/projects/plus/SHOE/spec.html

[MacGregor, 1991] MacGregor, R. *Inside the LOOM Description Classifier.* ACM SIGART Bulletin, Vol.2(3):88-92, ACM Press, New York, USA, 1991.

[Madhavan *et al.*, 2001] Madhavan, J., Bernstein, P., Rahm, E. *Generic schema matching using Cupid.* In Proceedings 27th VLDB, Roma, Italy, pp 48-58, 2001. http://research.microsoft.com/∼philbe/CupidVLDB01.pdf

[Mäedche and Staab, 2001] Mäedche, A., Staab, S. *Ontology Learning for the Semantic Web*, IEEE Intelligent Systems, Vol.16(2), 2001.

[Mäedche *et al.*, 2002] Maedche, A., Motik, B., Silva, N., Volz, R. *MAFRA - A Mapping Framewoek for Distributed Ontologies*, In Proceedings of 13th European Conference on Knowledge Engineering and Knowledge Management (EKAW), Siquenca, Spain, 2002.

[Malucelli and Oliveira, 2004a] Malucelli, A., Oliveira, E. *Towards to Similarity Identification to help in the Agents' Negotiation*, In: Bazzan, A. L. C., Labidi, S. (eds.) Advances in Artificial Intelligence - SBIA 2004, LNAI, Springer, pp. 536-545, 2004.

[Malucelli and Oliveira, 2004b] Malucelli, A., Oliveira, E. *Ontology-Services Agent to Help in the Structural and Semantic Heterogeneity*, In: Camarinha-Matos, L. (eds.) Virtual Enterprises and Collaborative Networks, Kluwer Academic Publishers, pp. 175-182, 2004.

[Malucelli and Oliveira, 2005] Malucelli, A., Oliveira, E. *Using Similarity Measures for an Efficient Business Information-Exchange*, In Proceddings of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT 2005), IEEE Computer Society, Compiègne, France, pp. 234-237, 2005.

[Malucelli *et al.*, 2005a] Malucelli, A., Cardoso, H. L. Oliveira, E. *Enriching a MAS Environment with Institutional Services.* In: Weyns, D., Parunak, V., Michel, F. (eds.) Environments for Multiagent Systems II (E4MAS), LNCS 3830, Springer-Verlag, Berlin, 2005.

[Malucelli *et al.*, 2005b] Malucelli, A., Palzer, D., Oliveira, E. *Combining Ontologies and Agents to Help in Solving the Heterogeneity Problem in E-Commerce Negotiations.* In Proceedings of the International Workshop on Data Engineering Issues in E-Commerce (DEEC 2005), IEEE Computer Society, Tokyo, Japan, pp. 26-35, 2005.

[Malucelli *et al.*, 2006] Malucelli, A., Palzer, D., Oliveira, E. *Ontology-based Services to help solving the heterogeneity problem in e-commerce negotiations.* To be published in Journal of Electronic Commerce Research and Applications - Special Issue Electronic data engineering: the next frontier in e-commerce, Vol.5(3), Elsevier, 2006.

[McGuinness and van Harmelen, 2004] McGuinness, D., van Harmelen, F. *OWL Web Ontology Language Overview*, W3C Recommendation, 10 February 2004, http://www.w3.org/TR/owl-features

[McGuinness *et al.*, 2000] McGuinness, D. L., Fikes, R., Rice, J. and Wilder, S. *An environment for merging and testing large ontologies*, In Proceedings of 7th International Conference on Principles of Knowledge Representation and Reasoning. Colorado, USA, April, 2000.

[Melnik *et al.*, 2002] Melnik, S., Garcia-Molina, H., Rahm, E. *Similarity flooding: A versatile graph matching algorithm*, In Proceedings of the International Conference on Data Engineering (ICDE), pp. 117-128, 2002.

[Melnik *et al.*, 2003] Melnik, S., Rahm, E., Bernstein, P. A., *Rondo: A Programming Platform for Model Management*, In Proceedings ACM SIGMOD 2003, San Diego, June, 2003 http://www-db.stanford.edu/∼melnik/pub/melnik_SIGMOD03.pdf

[Mena *et al.*, 1999] Mena, E., Illarramendi, A., Kashyap, V., Sheth, A. *OBSERVER: An Approach for Query Processing in Global Information Systems based on Interoperation across Pre-existing Ontologies*, Distributed and Parallel Databases Journal, 1999.

[Miller, 1995] Miller, G. *WordNet:A Lexical Database for English*, Communication of ACM, 38(11):39-41, 1995.

[Mitra and Wiederhold, 2002] Mitra, P., Wiederhold, G. *Resolving Terminological Heterogeneity in Ontologies*, In Proceedings of the ECAI-02, Workshop on Ontologies and Semantic Interoperability, Lyon, 2002.

[Morris and Hirst, 1991] Morris, J., Hirst, G. *Lexical cohesion computes by thesaural relations as an indicator of the structure of text*, Computational Linguistics, Vol.17(1):21-48, March, 1991.

[Natrajan *et al.*, 1997] Natrajan, A., Powell, A., French, J. C. *Using N-grams to Process Hindi Queries with Transliteration Variations*, Technical Report No. CS-97-17, University of Virginia, July 16, 1997.

[Neches *et al.*, 1991] Neches, R., Fikes, R. E., Finin, T., Gruber, T. R., Senator, T., Swartout, W. R. *Enabling technology for knowledge sharing*, AI Magazine, Vol.12(3):36-56, 1991.

[Ng and Lim, 2001] Ng, W. K., Lim, E. P. *Standardization and Integration in Business-to-Business Electronic Commerce*, IEEE Intelligent Systems, January/February, pp. 12-14, 2001.

[Noy and McGuinness, 2001] Noy, N. F., McGuinness, D. L. *Ontology Development 101: A Guide to Creating your First Ontology*, Stanford Knowledge Systems Laboratory Technical Report KSL-01-05 and Stanford Medical Informatics Technical Report SMI-2001-0880, March, 2001.

[Noy and Musen, 2000] Noy, N. F. Musen, M. A. *PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment*, In Proceedings of 17th National Conference on Artificial Intelligence, Austin, Texas, 2000.

[Noy *et al.*, 2000] Noy, N. F., Fergerson, R. W., Musen, M. A. *The knowledge model of Protégé-2000: Combining interoperability and flexibility.* In: Dieng, R., Corby, O. (eds.), Knowledge Engineering and Knowledge Management. Methods, Models, and Tools: 12th International Conference, LNCS, pp.17, 2000.

[Noy and Musen, 2001] Noy, N. F., Musen, M. A. *Anchor-PROMPT: Using non-local context for semantic matching.* In Proceedings of IJCAI 2001 - Workshop on ontology and information sharing, Seattle, US, pp 63-70, 2001. http://dit.unitn.it/∼accord/RelatedWork/Matching/noy.pdf

[O'Leary *et al.*, 1997] O'Leary, D. E., Kuokka, D., Plant, R. *Artificial Intelligence and Virtual Organizations*, Communications of the ACM, Vol.40(1):52-59, January, 1997.

[Okumura and Honda, 1994] Okumura, M., Honda, T. *Word sense disambiguation and text segmentation based on lexical cohesion*, In Proceedings of the Fifteenth International Conference on Computational Linguistics, Vol.(2):755-761, Kyoto, Japan, August, 1994.

[Oliveira *et al.*, 1999] Oliveira, E., Fisher, K., Stepankova, O. *Multi-Agent Systems: Which Research for which Application*, Journal of Robotics and Autonomous Systems, Vol.27(1-2):91-106, Elsevier, 1999.

[Palzer, 2005] Palzer, D. *Ontology-based Services in Multi-Agent Systems*, Diploma Thesis, University of Applied Sciences Trier and Faculty of Engineering, University of Porto, February, 2005.

[Pedersen *et al.*, 2003] Pedersen, T., Banerjee, S., Patwardhan, S. *Maximizing Semantic Relatedness to Perform Word Sense Disambiguation*, Elsevier Science, October, 2003.

[Petrie and Bussler, 2003] Petrie, C., Bussler, C. *Service Agents and Virtual Enterprises: A Survey.* IEEE Internet Computing, IEEE Computer Society, pp. 2-12, July/August 2003.

[Pinto, 1999] Pinto, H. S. *Towards Ontology Reuse.* In Proceedings of the AAAI99's Workshop on Ontology Management, WS-99-13, AAAI Press, pp.67-73, 1999.

[Pinto *et al.*, 1999] Pinto, H. S., Gómez-Pérez, A. and Martins, J. P. *Some issues on ontology integration.* In: Benjamins, V. R., Goméz-Pérez, A. (eds.) Proceedings of the IJCAI'99 - Workshop on Ontology and Problem-Solving Methods: Lesson learned and Future Trends, CEUR Publications, Amsterdam, pp. 7.1-7.11, 1999.

[Pinto, 2000] Pinto, H. S. *Ontology Integration: Characterization of the Process and a Methodology to Perform it*, Tese de Doutoramento, Instituto Superior Técnico, Universidade Técnica de Lisboa, 2000.

[Rahm and Bernstein, 2001] Rahm, E., Bernstein, P. A. *A Survey of Approaches to Automatic Schema Matching*, The VLDB Journal, 10(4):334-350, 2001.

[Resnik, 1995] Resnik, P. *Using information content to evaluate semantic similarity*, In Proceedings of the 14th International Joint Conference on Artificial Intelligence, Montreal, Canada, pp. 448-453. 1995.

[Rocha and Oliveira, 2001] Rocha, A. P., Oliveira, E. *Electronic Institutions as a framework for Agents' Negotiation and mutual Commitment*, In: Brazdil, P., Jorge, A. (eds.), Progress in Artificial Intelligence: Knowledge Extraction, Multi-agent Systems, Logic Programming, and Constraint Solving, LNAI 2258, Springer, pp.232-245, 2001.

[Rodríguez and Egenhofer, 2003] Rodríguez, M. A., Egenhofer, M. J. *Determining Semantic Similarity Among Entity Classes from Different Ontologies*. IEEE Transactions on Knowledge and Data Engineering, Vol.15(2):442-456, 2003.

[Shvaiko and Euzenat, 2005] Shvaiko, P., Euzenat, J. *A Survey of Schema-based Matching Approaches*, Journal on Data Semantics, 2005.

[Simon, 1983] Simon, H. *Why should machines learn ? Machine Learning: an artificial intelligence approach.* Michalski, R., Carbonell, J. and Mitchell, T. (eds.), Tioga Publishing Company, pp. 25-37, 1983.

[Singh *et al.*, 2001] Singh, T., Jayashankar, J. V., Singh, J., *E-Commerce in the U.S. and Europe – Is Europe Ready to Compete?* Business Horizons. March-April, pp.6-16, 2001.

[Sowa, 2000] Sowa, J. F. *Knowledge Representation: Logical, Philosophical, and Computational Foundations*, Brooks Cole Publishing Co., Pacific Grove, CA, 2000.

[Steels, 1998] Steels, L. *The origins of ontologies and communication conventions in multi-agent systems.* Autonomous Agents and Multi-Agent Systems, Vol.1(2):169-194, 1998.

[Studer *et al.*, 1998] Studer, R., Benjamins, V.R., Fensel, D. Knowledge *Engineering: Principles and Methods*, IEEE Transactions on Data and Knowledge

Engineering, Vol.25(1-2):161-197, 1998

[Stumme and Mäedche, 2001] Stumme, G. Mäedche, A. *Ontology Merging for Federated Ontologies on the Semantic Web.* In Proceedings of the Workshop on Ontologies and Information Sharing, IJCAI'01, Seattle, USA, 2001.

[Stumme and Mäedche, 2001a] Stumme, G. Maedche, A. *FCA-merge: bottom-up merging of ontologies*, In Proceedings of 17th IJCAI, Seattle, US, pp.225-230, 2001.

[Subramani and Walden, 2000] Subramani, M. Walden, E. *Economic returns to firms from business-to-business electronic commerce initiatives: an empirical examination*, Proceedings of the 21th international conference on Information systems, pp. 229-241, 2000.

[Suguri *et al.*, 2001] Suguri, H., Kodama, E., Miyazaki, M., Nunokawa, H., Noguchi, S. *Implementation of FIPA Ontology Service.* In Proceedings of the Workshop on Ontologies in Agent Systems, AAMAS, Montreal, Canada, 2001.

[Sure *et al.*, 2002] Sure, Y., Erdmann, M., Angele, J., Staab, S., Studer, R., Wenke, D. *OntoEdit: Collaborative Ontology Engineering for the Semantic Web*, In: Horrocks I., Hendler J. (eds.), First International Semantic Web Conference (ISWC'02), Sardinia, Italy, Springer Verlag, LNCS 2342, Berlin, Germany, pp. 221-235, 2002.

[Swartout *et al.*, 1997] Swartout, B., Ramesh, P., Knight, K., Russ, T. *Toward Distributed Use of Large-Scale Ontologies.* In: Farquhar, A., Gruninger, M., Gómez-Pérez, A., Uschold, M., van der Vet, P. (eds), AAAI'97 Spring Symposium on Ontological Engineering, Stanford University, California, pp. 138-148, 1997.

[Thalheim, 2000] Thalheim, B. *Entity-relationship modelling*, Foundation of Database Technology, Springer-Verlag, Berlin, Germany, 2000.

[Trastour *et al.*, 2002] Trastour, D., Bartolini, C., Preist, C. *Semantic Web Support for the Business-to-Business E-Commerce Lifecycle*, In Proceedings of the International WWW Conference, Honolulu, Hawaii, USA, 2002.

[Tzitzikas and Meghini, 2003] Tzitzikas, Y., Meghini, C. *Ostensive automatic schema mapping for taxonomybased peer-to-peer systems.* In Proceedings of the 7th International Workshop on Cooperative Information Agents, Helsinki, Finland, 2003.

[Uschold, 2001] Uschold, M. *Barriers to Effective Agent Communication*, Position Statement, Workshop on Ontologies in Agent Systems, Montreal, Canada, 2001.

[Ushold and Gruninger, 1996] Uschold, M., Grüninger, M. *Ontologies: Principles, Methods and Applications*, Knowledge Engineering Review, Vol.11(2):93-155, 1996.

[Uschold and Jasper, 1999] Uschold, M., Jasper, R. *A Framework for Understanding and Classifying Ontology Applications*, In: Benjamins VR (ed) IJCAI'99 Workshop on Ontology and Problem Solving Methods: Lessons Learned and Future Trends, Stockholm, Sweden, CEUR Workshop Proceedings 18:11.1-11.12. Amsterdam, The Netherlands. 1999. http://CEUR-WS.org/Vol-18

[van Diggelen *et al.*, 2005] van Diggelen, J., Jan Beun, R., Dignum, F, van Eijk, R. M., Meyer, J-J. *Optimal Communication Vocabularies and Heterogeneous Ontologies*, In: van Eijk, R. M., Huget, M. P., Dignum, F. (eds.), Developments in Agent Communication, LNAI 3396, Springer Verlag, Berlin Heidelberg, pp. 76-90, 2005.

[van Eijk *et al.*, 2001] van Eijk, R. M., Boer, F. S., van der Hoek, W., Meyer, J-J Ch. *On Dynamically Generated Ontology Translators in Agent Communication*, International Journal of Intelligent Systems, Vol.16, pp.587-607, 2001.

[van Harmelen *et al.*, 2003] van Harmelen, F., Hendler, J., Horrocks, I., McGuinness, D., Patel-Schneider, P. F., Stein, L. A. *OWL Web Ontology Language Reference: W3C Working* Draft 31, March 2003. http://www.w3.org/TR/2003/WS-owl-ref-20030331.

[van Heijst *et al.*, 1997] van Heijst, G., Schreiber, A. Th., Wielinga, B. J. *Using explicit ontologies in kbs development*, International Journal of Human-Computer Studies, Vol.45:184-292, 1997.

[Vázquez-Salceda and Dignum, 2003] Vázquez-Salceda, J., Dignum. F. *Modelling Electronic Organizations*, In: Marik, V., Muller, J., Pechoucek, M. (eds.),

Multi-Agent Systems and Applications III: 3rd. International/Central and Eastern European Conference on Multi-Agent Systems, LNAI 2691, Springer, pp. 584-593, 2003.

[Veres, 2004] Veres, C. *On the use of WordNet for semantic interoperability: towards "cognitive computing"*, In Grundspenkis, J., Kirikova, M. (eds) CAiSE'04 Workshops in connection with The 16th Conference on Advanced Information Systems Engineering, Knowledge and Model Driven Information Systems Engineering for Networked Organisations, Proceedings, Vol.3. Riga, Latvia, 2004.

[Visser *et al.*, 1997] Visser, P. R. S., Jones, D. M., Bench-Capon, T. J. M., Shave, M. J. R. *An Analysis of Ontology Mismatches; Heterogeneity vs. Interoperability*, In Proceedings of AAAI'97 Spring Symposium on Ontological Engineering, Stanford, 1997.

[Wache *et al.*, 2001] Wache, H., Vögele, T., Visser, U., Stuckenschmidt, H., Schuster, G., Neumann, H., Hübner, S. *Ontology-Based Integration of Information-A Survey of Existing Approaches*, In Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI-01), Workshop Ontologies and Information Sharing, Seattle, USA, 2001.

[Weiss, 1999] Weiss, G. Multiagent Systems. *A Modern Approach to Distributed Artificial Intelligence*, The MIT Press Cambridge, Massachusetts, London, England, 1999.

[Welty and Guarino, 2001] Welty, C., Guarino, N.: *Supporting ontological analysis of taxonomic relationships*, Data&Knowledge Engineering, Vol.39, pp. 51-74, 2001.

[Wiesman and Roos, 2004] Wiesman, F., Roos, N. *Domain independent learning of ontology mappings*, In: Jennings, N., Sierra, C., Sonenbergm, L., Tamble, M. (eds.), AAMAS, ACM Press, New York, USA, pp.846-853, 2004.

[Williams *et al.*, 2003] Williams, A., Padmanabhan, A., Blake, M. B.*Local Consensus Ontologies for B2B-Oriented Service Composition*, In: Rosenschein, J., Sandholm, T., Wooldridge, M., Yokoo, M. (eds.), AAMAS, pp. 647-654. ACM Press, Melborne, 2003.

[Willmott, 2001] Willmott, S., Constantinescu, I., Calisti, M. *Multilingual Agents: Ontologies, Languages and Abstractions*, In Proceedings of the Workshop on Ontologies in Agent Systems, 5th International Conference on Autonomous Agents, Montreal, Canada, 2001.

[Wooldridge, 2002] Wooldridge, M. *An Introduction to Multi-Agent Systems.* John Wiley&Sons Ltd, 2002. http://www-cdr.stanford.edu/ProcessLink/papers/JATL.html#WhatIsJATLite

Appendices

Appendix A

Matching Terms Service Experiments

In this Appendix we present experiments when applying the MTS. Table A.1 presents the similarity comparison for concept's description. The first column presents the respective description for the requested product. The second column presents the descriptions for each candidate and the last column points out the final result of the similarity comparison (SimCom) between the description of the requested product with each description of the candidates.

Requested Product Description	**Product-Candidate Description**	**SimCom**
Lighting_System: all components that make up the illumination of a car	**Brake_Light**: a red light on the rear of a motor vehicle that signals when the brakes are applied to slow or stop	0.07
	Light: any device serving as a source of car illumination	0.42
	Blinker: a light that flashes on and off; used as a signal or to send messages	0.06
	Motor: machine that converts other forms of energy into mechanical energy and so imparts motion	0.17
	Tire: a tire consisting of a rubber ring around the rim of an automobile wheel	0.05
Wheel_Rim: the outer part of a wheel to which the tire is attached	**Wheel_Rim**: Steel or aluminium part of wheel that holds tire	0.62

Requested Product Description	Product-Candidate Description	SimCom
Wheel: able to rotate about a central axle or pivot, with a durable but elastic rim or with regular teeth cut on the rim and for lightness often supported by spokes joined to the hub instead of being left solid	**Wheel**: a wheel that has a tire and rim and hubcap; used to propel the car	0.32
	Wheel: a circular rotating device, thin in relation to its face area	0.32
Gearing: wheelwork consisting of a connected set of rotating gears by which force is transmitted or motion or torque is changed	**Gears**: wheelwork consisting of a connected set of rotating gears by which force is transmitted or motion or torque is changed	1.0
	Gears: A wheel with teeth that, when meshed with the teeth of another wheel, is able to transmit, modify, or change the direction of an applied force.	0.32
Battery: a device that produces electricity	**Battery**: A device that stores energy and produces electric current by chemical action	0.88
	Mirror: polished surface that forms images by reflecting light	0.13
	Thermostat: The thermostat is a valve located in the cooling system of a vehicle that automatically regulates the coolant flow through the radiator and engine based on the coolant temperature.	0.06
	Thermostat: The thermostat is a device and produces whatever.	0.69
Klaxon: a kind of loud horn formerly used on motor vehicles	**Horn**: a device on a vehicle that is used to make a loud noise as a warning or signal to other people	0.42
Seat_belt: a belt which fastens around someone travelling in a vehicle or aircraft, and which holds them in their seat in order to reduce the risk of them being injured in an accident	**Seat_belt**: a safety belt used in a car or plane to hold you in your seat in case of an accident	0.39
	Belt: belt attaching you to some object as a restraint in order to prevent you from getting hurt	0.40

Requested Product Description	Product-Candidate Description	SimCom
Speedometer: a meter fixed to a vehicle that measures and displays its speed	**Speedometer**: a device in a vehicle which shows how fast the vehicle is moving	0.34
Cylinder_Block: a metal casting containing the cylinders and cooling ducts of an engine	**Cylinder_Block**: The basic part of the engine to which other engine parts are attached. It is usually a casting and includes engine cylinders and the upper part of the crankcase.	0.54
	Tachometer: a device for measuring the rate at which something turns	0.23
Bumper: a mechanical device consisting of bars at either end of a vehicle to absorb shock and prevent serious damage	**Bumper**: a horizontal bar along the lower front and lower back part of a motor vehicle to help protect it if there is an accident	0.22
	Parking_Brake: Used to prevent the car from rolling when not being driven, the parking brake uses cables to mechanically apply brakes, usually the rear brakes	0.21
Parking_Brake: An auxiliary brake attached to a rear wheel or to the transmission that keeps the car from moving accidentally	**Hand_Brake**: A separate brake system in a vehicle for use in case of failure of the regular brakes and commonly used as a parking brake	0.22
	Hand_Brake: a device operated by hand which locks into position and prevents a vehicle from moving	0.17

Table A.1: Comparison between Concept's Description

Table A.2 presents the final results for the similarity matching terms service when CEAg requests different products (first column) and the SEAg proposes some products as candidate (second column). The last column presents the final results for the MTS. The candidate product found as correspondent to the requested one is marked in bold as well as its final result.

Requested Product [CEAg's ontology]	**Product proposed through Pre-selection [SEAg's ontology]**	**Final Similarity**
suspension	**Suspension_System**	**0.81**
throttle	Rearview_Mirror	0.29
	Accelerator	**0.77**
	Foglamp	0.30
	Rear_Lamp	0.33
	Transmitting_Aerial	0.37
	Sparking_Plug	0.36
	Headlight	0.33
	Clutch	0.53
	Belt	0.28
	Car_Horn	0.35
clutch_pedal	Rearview_Mirror	0.29
	Accelerator	0.61
	Foglamp	0.27
	Rear_Lamp	0.31
	Transmitting_Aerial	0.36
	Sparking_Plug	0.34
	Headlight	0.29
	Clutch	**0.81**
	Belt	0.27
	Car_Horn	0.35
antenna	Rearview_Mirror	0.40
	Accelerator	0.42
	Foglamp	0.36
	Rear_Lamp	0.44
	Transmitting_Aerial	**0.84**
	Headlight	0.43
	Clutch	0.40

Requested Product [CEAg s ontology]	**Product proposed through Pre-selection [SEAg s ontology]**	**Final Similarity**
foglight	Rearview_Mirror	0.33
	Accelerator	0.14
	Foglamp	**0.91**
	Rear_Lamp	0.57
	Transmitting_Aerial	0.02
	Sparking_Plug	0.10
	Headlight	0.25
	Clutch	0.11
	Belt	0.14
hand_brake	Windscreen	0.28
	Speed_Indicator	0.34
	Parking_Brake	**0.80**
	Push_Rod	0.07
bumper	Windscreen	0.37
	Speed_Indicator	0.40
	Parking_Brake	0.40
	Car_Window	0.35
	Push_Rod	0.51
mirror	**Rearview_Mirror**	**0.55**
	Accelerator	0.40
	Foglamp	0.38
	Rear_Lamp	0.44
	Transmitting_Aerial	0.44
	Headlight	0.46
	Clutch	0.37
claxon	Sparking_Plug	0.39
	Belt	0.29
	Car_Horn	0.48
gearshift	Electric_Battery	0.41
	Wheel	0.47
	V-Belt	0.28
	Gear_Lever	**0.90**
windshield	Windscreen	**0.78**
	Speed_Indicator	**0.27**
	Parking_Brake	**0.29**
	Push_Rod	**0.09**

Requested Product [CEAg s ontology]	**Product proposed through Pre-selection [SEAg s ontology]**	**Final Similarity**
headlamp	Rearview_Mirror	0.33
	Accelerator	0.36
	Foglamp	**0.63**
	Rear_Lamp	0.46
	Transmitting_Aerial	0.25
	Sparking_Plug	0.38
	Headlight	0.83
	Clutch	0.33
	Belt	0.27
taillight	Rearview_Mirror	0.39
	Accelerator	0.36
	Foglamp	0.41
	Rear_Lamp	**0.91**
	Transmitting_Aerial	0.27
	Sparking_Plug	0.40
	Headlight	0.47
	Clutch	0.33
	Belt	0.31
spark_plug	**Sparking_Plug**	**0.83**
	Belt	0.30
	Car_Horn	0.47
window	Speed_Indicator	0.37
	Parking_Brake	0.38
	Car_Window	**0.62**
	Push_Rod	0.18
battery	**Electric_Battery**	**0.85**
	Wheel	0.36
	V-Belt	0.25
	Gear_Lever	0.41
speedometer	Windscreen	0.30
	Speed_Indicator	**0.87**
	Parking_Brake	0.36
motor	**Engine**	**0.77**
	Airbag	0.07
	Cylinder_Block	0.38
	Engine_Block	0.32

Requested Product [CEAg s ontology]	**Product proposed through Pre-selection [SEAg s ontology]**	**Final Similarity**
steering_wheel	Rearview_Mirror	0.29
	Accelerator	0.42
	Foglamp	0.33
	Rear_Lamp	0.34
	Transmitting_Aerial	0.26
	Sparking_Plug	0.35
	Headlight	0.32
	Clutch	0.37
	Belt	0.26
hub_cap	Accelerator	0.29
	Foglamp	0.26
	Rear_Lamp	0.29
	Transmitting_Aerial	0.29
	Sparking_Plug	0.30
	Headlight	0.27
	Clutch	0.25
	Belt	0.34
	Car_Horn	0.30
wheel_rim	Oil_Sump	0.08
	Electric_Battery	0.04
	Wheel	0.33
tire	Electric_Battery	0.30
	Wheel	0.41
	V-Belt	0.19
	Gear_Lever	0.28
thermostat	Rearview_Mirror	0.33
	Accelerator	0.50
	Foglamp	0.32
	Rear_Lamp	0.36
	Transmitting_Aerial	0.39
	Sparking_Plug	0.39
	Headlight	0.36
	Clutch	0.38
	Belt	0.30
	Car_Horn	0.39
water_pump	Windscreen	0.27
	Speed_Indicator	0.32
	Parking_Brake	0.36
	Push_Rod	0.13
heating_system	Suspension_System	0.38

Table A.2: Similarity Matching Results

Appendix B

Weighted Combination Results

This Appendix presents experiments applying weights. The list do not have correspondent concepts. Table B.1 presents in the first two columns the pair of compared concepts, followed by the matching terms final result calculated with the standard (without using weights) and the weighted average formulas and, in the last column their difference.

Requested Product	**Candidate Product**	**Standard**	**Weighted Average**	**Difference**
Throttle	Rearview_window	0.19	0.29	9.67%
	Foglamp	0.22	0.30	8.41%
	Rear_lamp	0.24	0.33	8.88%
	Trnasmitting_aerial	0.28	0.37	8.91%
	Sparking_plug	0.27	0.36	9.30%
	Headlight	0.25	0.33	8.65%
	Clutch	0.37	0.53	16.02%
	Belt	0.22	0.28	5.35%
	Carn_horn	0.24	0.35	10.97%
Clutch_pedal	Rearview_window	0.25	0.29	4.02%
	Accelerator	0.52	0.61	8.51%
	Foglamp	0.20	0.27	6.57%
	Rear_lamp	0.26	0.31	4.74%
	Transmitting_aerial	0.34	0.36	2.33%
	Sparking_plug	0.28	0.34	5.18%
	Headlight	0.22	0.29	7.05%
	Belt	0.21	0.27	6.13%
	Carn_horn	0.32	0.35	3.38%

Requested Product	**Candidate Product**	**Standard**	**Weighted Average**	**Difference**
Antenna	Rearview_window	0.34	0.40	5.74%
	Accelerator	0.34	0.42	7.96%
	Foglamp	0.26	0.36	9.71%
	Rear_lamp	0.39	0.44	5.71%
	Headlight	0.36	0.43	6.97%
	Clutch	0.33	0.40	6.15%
Hand_brake	Windscreen	0.20	0.28	8.00%
	Speed_indicator	0.25	0.34	8.41%
Bumper	Windscreen	0.36	0.37	1.23%
	Speed_indicator	0.38	0.40	1.80%
	Parking_Brake	0.34	0.40	6.02%
	Car_window	0.27	0.35	7.37%
Mirror	Accelerator	0.31	0.40	9.60%
	Foglamp	0.30	0.38	7.96%
	Rear_lamp	0.39	0.44	5.68%
	Transmitting_aerial	0.33	0.44	10.81%
	Headlight	0.42	0.46	4.15%
	Clutch	0.28	0.37	8.62%
	Sparking_Plug	0.37	0.43	6.57%
	Clutch	0.24	0.33	8.97%
	Carn_horn	0.31	0.41	9.37%
Claxon	Sparking_plug	0.29	0.39	10.59%
	Belt	0.22	0.29	7.00%
Gearshift	Electtric_battery	0.32	0.41	9.11%
	Wheel	0.34	0.47	13.90%
Windshield	Speed_indicator	0.22	0.27	5.59%
	Parking_brake	0.22	0.29	6.83%
Headlamp	Rearview_mirror	0.23	0.33	9.38%
	Accelerator	0.26	0.36	9.69%
	Foglamp	0.45	0.63	17.63%
	Rear_lamp	0.38	0.46	8.77%
	Transmitting_aerial	0.17	0.25	8.33%
	Sparking_plug	0.26	0.38	11.89%
	Clutch	0.24	0.33	9.05%
	Belt	0.18	0.27	8.94%

Requested Product	**Candidate Product**	**Standard**	**Weighted Average**	**Difference**
Taillight	Rearview_mirror	0.35	0.39	3.45%
	Accelerator	0.26	0.36	10.02%
	Foglamp	0.33	0.41	8.65%
	Transmitting_aerial	0.20	0.27	6.50%
	Sparking_plug	0.29	0.40	10.32%
	Headlight	0.40	0.47	7.65%
	Clutch	0.24	0.33	9.10%
	Belt	0.27	0.31	4.32%
Spark_plug	Belt	0.21	0.30	8.78%
	Carn_horn	0.39	0.47	8.06%
	Speed_indicator	0.32	0.37	5.17%
	Parking_brake	0.30	0.38	7.75%
Battery	Wheel	0.26	0.36	9.65%
	Gear_level	0.32	0.41	8.88%
Speedometer	Windscreen	0.27	0.30	2.89%
	Parking_brake	0.30	0.36	5.86%
Motor	Cylinder_block	0.35	0.38	2.65%
	Engine_blocl	0.28	0.32	3.97%
				7.62%

Table B.1: Similarity Matching with and without Weighting

Appendix C

WordNet-based Similarity Measures Comparison

This Appendix presents the experiments to select the WordNet-based Semantic Similarity Algorithm. Table C.1 presents the compared concepts and the original result for each one of the methods: Hirst-St.Onge (HSO), Jiang-Conrath (JCN), Leacock-Chodorow (LCH), Resnik (RES), and Wu-Palmer (WUP).

The higher (the best case, when concepts are considered identical) value varies for each method and the lower value (meaning no similarity) is zero.

The acronym "NFW" means "Not found in WordNet". It happens when the concept is not found in WordNet. "w#p#s" means "word#pos#sense". As explained in subsection 6.6.1.3, the algorithms require two words as input. The format is word#pos#sense, where **word** is a term, **pos** signifies the part of speech (**n** for noun, **v** for verb, **a** for adjective and **r** for relation), and **sense** is a positive integer and represents the sense of the word in WordNet.

Concept 1	Concept 2	HSO	JCN	LCH	RES	WUP
automobile	car	16	29590099.4292139	3.58351893845611	6.10857528657738	1.1
automobile	engine	6	0.126058375935633	1.28093384546206	3.73884299593615	0.526315789473684
automobile	wheel	5	0.118258094357266	1.18562366565774	3.73884299593615	0.5
automobile	long_distance_light	w#p#s	NFW	NFW	NFW	NFW
automobile	stoplight	5	0	0.750305594399894	0	0.111111111111111
automobile	parking_light	w#p#s	NFW	NFW	NFW	NFW
automobile	headlight	5	0.10596688909857	1.28093384546206	3.73884299593615	0.526315789473684
automobile	low_beam_light	w#p#s	NFW	NFW	NFW	NFW
automobile	v-belt	0	NFW	NFW	NFW	NFW
automobile	oil_sump	w#p#s	NFW	NFW	NFW	NFW
automobile	engine_block	w#p#s	0	1.09861228866811	2.55801504707402	0.421052631578947
automobile	cylinder_block	w#p#s	0	1.09861228866811	2.55801504707402	0.421052631578947
automobile	speed_indicator	w#p#s	0	1.18562366565774	3.73884299593615	0.5
automobile	rotation_speed	w#p#s	NFW	NFW	NFW	NFW
automobile	speedometer	6	0	1.18562366565774	3.73884299593615	0.5
automobile	paint	2	0.0952445994738899	1.28093384546206	2.55801504707402	0.5
automobile	brake	5	0.107726588704472	1.38629436111989	3.73884299593615	0.555555555555556
automobile	battery	0	0.0638494865663311	0.693147180559945	0	0.105263157894737
automobile	window	2	0.108148795573475	1.18562366565774	2.55801504707402	0.444444444444444
automobile	bolt	0	0.0555148190160038	0.750305594399894	0	0.111111111111111
automobile	mirror	6	0.10929883954991	1.38629436111989	3.73884299593615	0.555555555555556
automobile	car	16	29590099.4292139	3.58351893845611	6.10857528657738	1.1
automobile	automobile	16	29590099.4292139	3.58351893845611	6.10857528657738	1.1

Concept 1	**Concept 2**	**HSO**	**JCN**	LCH	**RES**	**WUP**
automobile	motor	5	0.130801871107812	1.38629436111989	3.73884299593615	0.555555555555556
automobile	break_light	w#p#s	NFW	NFW	NFW	NFW
automobile	rear_fog_light	w#p#s	NFW	NFW	NFW	NFW
automobile	steering_wheel	w#p#s	0.10596688909857	1.18562366565774	3.73884299593615	0.5
automobile	tire	4	0.0877724172425366	1.18562366565774	2.55801504707402	0.444444444444444
automobile	tyre	0	0	0.944461608840852	0.855458788752587	0.235294117647059
automobile	wheel_rim	w#p#s	NFW	NFW	NFW	NFW
automobile	fuel_injector	w#p#s	NFW	NFW	NFW	NFW
automobile	spark_plug	w#p#s	0	1.38629436111989	3.73884299593615	0.555555555555556
engine	car	6	0.126058375935633	1.28093384546206	3.73884299593615	0.526315789473684
engine	wheel	2	0.102496605608936	1.50407739677627	4.68535709642899	0.631578947368421
engine	long_distance_light	w#p#s	NFW	NFW	NFW	NFW
engine	stoplight	0	0	0.810930216216329	0	0.117647058823529
engine	parking_light	w#p#s	NFW	NFW	NFW	NFW
engine	headlight	0	0.0931337175342702	1.6376087894008	4.68535709642899	0.666666666666667
engine	low_beam_light	w#p#s	NFW	NFW	NFW	NFW
engine	v-belt	0	NFW	NFW	NFW	NFW
engine	oil_sump	w#p#s	NFW	NFW	NFW	NFW
engine	engine_block	w#p#s	0	1.18562366565774	2.55801504707402	0.444444444444444
engine	cylinder_block	w#p#s	0	1.18562366565774	2.55801504707402	0.444444444444444
engine	speed_indicator	w#p#s	0	1.50407739677627	4.68535709642899	0.631578947368421
engine	rotation_speed	w#p#s	NFW	NFW	NFW	NFW
engine	speedometer	0	0	1.50407739677627	4.68535709642899	0.631578947368421

Concept 1	**Concept 2**	**HSO**	**JCN**	LCH	**RES**	**WUP**
engine	paint	0	0.0730318766155491	1.38629436111989	2.55801504707402	0.533333333333333
engine	brake	2	0.0944902791202106	1.79175946922806	4.68535709642899	0.705882352941177
engine	battery	0	0.0530357641371822	0.750305594399894	0	0.111111111111111
engine	window	0	0.0803865754800409	1.28093384546206	2.55801504707402	0.470588235294118
engine	bolt	0	0.0471551834508911	0.810930216216329	0	0.117647058823529
engine	mirror	2	0.0956977375299088	1.79175946922806	4.68535709642899	0.705882352941177
engine	car	6	0.126058375935633	1.28093384546206	3.73884299593615	0.526315789473684
engine	engine	16	29590099.4292139	3.58351893845611	9.30194335189837	1
engine	motor	4	3.47605949678218	2.89037175789616	9.01426127944659	0.941176470588235
engine	break_light	w#p#s	NFW	NFW	NFW	NFW
engine	rear_fog_light	w#p#s	NFW	NFW	NFW	NFW
engine	steering_wheel	w#p#s	0.0931337175342702	1.50407739677627	4.68535709642899	0.631578947368421
engine	tire	0	0.0685566881862268	1.28093384546206	2.55801504707402	0.470588235294118
engine	tyre	0	0	1.01856958099457	0.855458788752587	0.25
engine	wheel_rim	w#p#s	NFW	NFW	NFW	NFW
engine	fuel_injector	w#p#s	NFW	NFW	NFW	NFW
engine	spark_plug	w#p#s	0	1.79175946922806	4.68535709642899	0.705882352941177
wheel	stoplight	0	0	0.750305594399894	0	0.111111111111111
wheel	parking_light	w#p#s	NFW	NFW	NFW	NFW
wheel	headlight	0	0.0888060186528432	1.50407739677627	4.68535709642899	0.631578947368421
wheel	low_beam_light	w#p#s	NFW	NFW	NFW	NFW
wheel	v-belt	0	NFW	NFW	NFW	NFW
wheel	oil_sump	w#p#s	NFW	NFW	NFW	NFW

Concept 1	**Concept 2**	**HSO**	**JCN**	LCH	**RES**	**WUP**
wheel	wheel_block	w#p#s	NFW	NFW	NFW	NFW
wheel	cylinder_block	w#p#s	0	1.09861228866811	2.55801504707402	0.421052631578947
wheel	speed_indicator	w#p#s	0	1.38629436111989	4.68535709642899	0.6
wheel	rotation_speed	w#p#s	NFW	NFW	NFW	NFW
wheel	speedometer	0	0	1.38629436111989	4.68535709642899	0.6
wheel	paint	0	0.0703437741305538	1.28093384546206	2.55801504707402	0.5
wheel	brake	0	0.0900386028400648	1.6376087894008	4.68535709642899	0.666666666666667
wheel	battery	0	0.0516037163637563	0.693147180559945	0	0.105263157894737
wheel	window	0	0.0771418262339885	1.18562366565774	2.55801504707402	0.444444444444444
wheel	bolt	0	0.0460196997210488	0.750305594399894	0	0.111111111111111
wheel	mirror	0	0.091134308814737	1.6376087894008	4.68535709642899	0.666666666666667
wheel	car	5	0.118258094357266	1.18562366565774	3.73884299593615	0.5
wheel	wheel	16	29590099.4292139	3.58351893845611	9.82519149566292	1.1
wheel	motor	0	0.105610688405151	1.6376087894008	4.68535709642899	0.666666666666667
wheel	break_light	w#p#s	NFW	NFW	NFW	NFW
wheel	rear_fog_light	w#p#s	NFW	NFW	NFW	NFW
wheel	steering_wheel	w#p#s	0196997174444071	1.6376087894008	7.77749865229766	0.7
wheel	tire	4	0.0661825762332245	1.18562366565774	2.55801504707402	0.444444444444444
wheel	tyre	0	0	0.944461608840852	0.855458788752587	0.235294117647059
wheel	wheel_rim	w#p#s	NFW	NFW	NFW	NFW
wheel	fuel_injector	w#p#s	NFW	NFW	NFW	NFW
wheel	spark_plug	w#p#s	0	1.6376087894008	4.68535709642899	0.666666666666667
long_distance_light	stoplight	w#p#s	NFW	NFW	NFW	NFW

Concept 1	**Concept 2**	**HSO**	**JCN**	LCH	**RES**	**WUP**
long_distance_light	parking_light	w#p#s	NFW	NFW	NFW	NFW
long_distance_light	headlight	w#p#s	NFW	NFW	NFW	NFW
long_distance_light	low_beam_light	w#p#s	NFW	NFW	NFW	NFW
long_distance_light	v-belt	w#p#s	NFW	NFW	NFW	NFW
long_distance_light	oil_sump	w#p#s	NFW	NFW	NFW	NFW
long_distance_light	engine_block	w#p#s	NFW	NFW	NFW	NFW
long_distance_light	cylinder_block	w#p#s	NFW	NFW	NFW	NFW
long_distance_light	speed_indicator	w#p#s	NFW	NFW	NFW	NFW
long_distance_light	rotation_speed	w#p#s	NFW	NFW	NFW	NFW
long_distance_light	speedometer	w#p#s	NFW	NFW	NFW	NFW
long_distance_light	paint	w#p#s	NFW	NFW	NFW	NFW
long_distance_light	brake	w#p#s	NFW	NFW	NFW	NFW
long_distance_light	battery	w#p#s	NFW	NFW	NFW	NFW
long_distance_light	window	w#p#s	NFW	NFW	NFW	NFW
long_distance_light	bolt	w#p#s	NFW	NFW	NFW	NFW
long_distance_light	mirror	w#p#s	NFW	NFW	NFW	NFW
long_distance_light	car	w#p#s	NFW	NFW	NFW	NFW
long_distance_light	engine	w#p#s	NFW	NFW	NFW	NFW
long_distance_light	long_distance_light	w#p#s	NFW	NFW	NFW	NFW
long_distance_light	motor	w#p#s	NFW	NFW	NFW	NFW
long_distance_light	break_light	w#p#s	NFW	NFW	NFW	NFW
long_distance_light	rear_fog_light	w#p#s	NFW	NFW	NFW	NFW
long_distance_light	steering_wheel	w#p#s	NFW	NFW	NFW	NFW

Concept 1	**Concept 2**	**HSO**	**JCN**	LCH	**RES**	**WUP**
long_distance_light	tire	w#p#s	NFW	NFW	NFW	NFW
long_distance_light	tyre	w#p#s	NFW	NFW	NFW	NFW
long_distance_light	wheel_rim	w#p#s	NFW	NFW	NFW	NFW
long_distance_light	fuel_injector	w#p#s	NFW	NFW	NFW	NFW
long_distance_light	spark_plug	w#p#s	NFW	NFW	NFW	NFW
stoplight	stoplight	16	0	3.58351893845611	0	1
stoplight	parking_light	w#p#s	NFW	NFW	NFW	NFW
stoplight	headlight	0	0	0.810930216216329	0	0.117647058823529
stoplight	low_beam_light	w#p#s	NFW	NFW	NFW	NFW
stoplight	v-belt	0	NFW	NFW	NFW	NFW
stoplight	oil_sump	w#p#s	NFW	NFW	NFW	NFW
stoplight	engine_block	w#p#s	0	0.810930216216329	0	0.117647058823529
stoplight	cylinder_block	w#p#s	0	0.810930216216329	0	0.117647058823529
stoplight	speed_indicator	w#p#s	0	0.750305594399894	0	0.111111111111111
stoplight	rotation_speed	w#p#s	NFW	NFW	NFW	NFW
stoplight	speedometer	0	0	0.750305594399894	0	0.111111111111111
stoplight	paint	0	0	1.01856958099457	0	0.142857142857143
stoplight	brake	0	0	0.8754687373539	0	0.125
stoplight	battery	0	0	0.810930216216329	0	0.117647058823529
stoplight	window	0	0	0.8754687373539	0	0.125
stoplight	bolt	0	0	0.8754687373539	0	0.125
stoplight	mirror	0	0	0.8754687373539	0	0.125
stoplight	car	5	0	0.750305594399894	0	0.111111111111111

Concept 1	**Concept 2**	**HSO**	**JCN**	LCH	**RES**	**WUP**
stoplight	engine	0	0	0.810930216216329	0	0.117647058823529
stoplight	motor	0	0	0.8754687373539	0	0.125
stoplight	break_light	w#p#s	NFW	NFW	NFW	NFW
stoplight	rear_fog_light	w#p#s	NFW	NFW	NFW	NFW
stoplight	steering_wheel	w#p#s	0	0.750305594399894	0	0.111111111111111
stoplight	tire	0	0	0.8754687373539	0	0.125
stoplight	tyre	0	0	0.944461608840852	0	0.133333333333333
stoplight	wheel_rim	w#p#s	NFW	NFW	NFW	NFW
stoplight	fuel_injector	w#p#s	NFW	NFW	NFW	NFW
stoplight	spark_plug	w#p#s	0	0.8754687373539	0	0.125
parking_light	parking_light	w#p#s	NFW	NFW	NFW	NFW
parking_light	headlight	w#p#s	NFW	NFW	NFW	NFW
parking_light	low_beam_light	w#p#s	NFW	NFW	NFW	NFW
parking_light	v-belt	w#p#s	NFW	NFW	NFW	NFW
parking_light	oil_sump	w#p#s	NFW	NFW	NFW	NFW
parking_light	engine_block	w#p#s	NFW	NFW	NFW	NFW
parking_light	cylinder_block	w#p#s	NFW	NFW	NFW	NFW
parking_light	speed_indicator	w#p#s	NFW	NFW	NFW	NFW
parking_light	rotation_speed	w#p#s	NFW	NFW	NFW	NFW
parking_light	speedometer	w#p#s	NFW	NFW	NFW	NFW
parking_light	paint	w#p#s	NFW	NFW	NFW	NFW
parking_light	brake	w#p#s	NFW	NFW	NFW	NFW
parking_light	battery	w#p#s	NFW	NFW	NFW	NFW

Concept 1	**Concept 2**	**HSO**	**JCN**	LCH	**RES**	**WUP**
parking_light	window	w#p#s	NFW	NFW	NFW	NFW
parking_light	bolt	w#p#s	NFW	NFW	NFW	NFW
parking_light	mirror	w#p#s	NFW	NFW	NFW	NFW
parking_light	car	w#p#s	NFW	NFW	NFW	NFW
parking_light	engine	w#p#s	NFW	NFW	NFW	NFW
parking_light	motor	w#p#s	NFW	NFW	NFW	NFW
parking_light	break_light	w#p#s	NFW	NFW	NFW	NFW
parking_light	rear_fog_light	w#p#s	NFW	NFW	NFW	NFW
parking_light	steering_wheel	w#p#s	NFW	NFW	NFW	NFW
parking_light	tire	w#p#s	NFW	NFW	NFW	NFW
parking_light	tyre	w#p#s	NFW	NFW	NFW	NFW
parking_light	wheel_rim	w#p#s	NFW	NFW	NFW	NFW
parking_light	fuel_injector	w#p#s	NFW	NFW	NFW	NFW
parking_light	spark_plug	w#p#s	NFW	NFW	NFW	NFW
headlight	headlight	16	29590099.4292139	3.58351893845611	10.8060207486746	1
headlight	low_beam_light	w#p#s	NFW	NFW	NFW	NFW
headlight	v-belt	0	NFW	NFW	NFW	NFW
headlight	oil_sump	w#p#s	NFW	NFW	NFW	NFW
headlight	engine_block	w#p#s	0	1.18562366565774	2.55801504707402	0.444444444444444
headlight	cylinder_block	w#p#s	0	1.18562366565774	2.55801504707402	0.444444444444444
headlight	speed_indicator	w#p#s	0	1.50407739677627	4.68535709642899	0.631578947368421
headlight	rotation_speed	w#p#s	NFW	NFW	NFW	NFW
headlight	speedometer	0	0	1.50407739677627	4.68535709642899	0.631578947368421

Concept 1	**Concept 2**	**HSO**	**JCN**	LCH	**RES**	**WUP**
headlight	paint	0	0.0658036369686689	1.38629436111989	2.55801504707402	0.533333333333333
headlight	brake	2	0.0827323063981283	1.79175946922806	4.68535709642899	0.705882352941177
headlight	battery	0	0.0491176540751561	0.750305594399894	0	0.111111111111111
headlight	window	0	0.071715610852917	1.28093384546206	2.55801504707402	0.470588235294118
headlight	bolt	0	0.0440321978143881	0.810930216216329	0	0.117647058823529
headlight	mirror	2	0.0836564897642843	1.79175946922806	4.68535709642899	0.705882352941177
headlight	car	5	0.10596688909857	1.28093384546206	3.73884299593615	0.526315789473684
headlight	engine	0	0.0931337175342702	1.6376087894008	4.68535709642899	0.666666666666667
headlight	motor	2	0.0956977375299088	1.79175946922806	4.68535709642899	0.705882352941177
headlight	break_light	w#p#s	NFW	NFW	NFW	NFW
headlight	rear_fog_light	fog	NFW	NFW	NFW	NFW
headlight	steering_wheel	w#p#s	0.0816904878961208	1.50407739677627	4.68535709642899	0.631578947368421
headlight	tire	0	0.0621482938901184	1.28093384546206	2.55801504707402	0.470588235294118
headlight	tyre	0	0	1.01856958099457	0.855458788752587	0.25
headlight	wheel_rim	w#p#s	NFW	NFW	NFW	NFW
headlight	fuel_injector	w#p#s	NFW	NFW	NFW	NFW
headlight	spark_plug	w#p#s	0	1.79175946922806	4.68535709642899	0.705882352941177
low_beam_light	low_beam_light	w#p#s	NFW	NFW	NFW	NFW
low_beam_light	v-belt	w#p#s	NFW	NFW	NFW	NFW
low_beam_light	oil_sump	w#p#s	NFW	NFW	NFW	NFW
low_beam_light	engine_block	w#p#s	NFW	NFW	NFW	NFW
low_beam_light	cylinder_block	w#p#s	NFW	NFW	NFW	NFW
low_beam_light	speed_indicator	w#p#s	NFW	NFW	NFW	NFW

Concept 1	**Concept 2**	**HSO**	**JCN**	LCH	**RES**	**WUP**
low_beam_light	rotation_speed	w#p#s	NFW	NFW	NFW	NFW
low_beam_light	speedometer	w#p#s	NFW	NFW	NFW	NFW
low_beam_light	paint	w#p#s	NFW	NFW	NFW	NFW
low_beam_light	brake	w#p#s	NFW	NFW	NFW	NFW
low_beam_light	battery	w#p#s	NFW	NFW	NFW	NFW
low_beam_light	window	w#p#s	NFW	NFW	NFW	NFW
low_beam_light	bolt	w#p#s	NFW	NFW	NFW	NFW
low_beam_light	mirror	w#p#s	NFW	NFW	NFW	NFW
low_beam_light	car	w#p#s	NFW	NFW	NFW	NFW
low_beam_light	engine	w#p#s	NFW	NFW	NFW	NFW
low_beam_light	motor	w#p#s	NFW	NFW	NFW	NFW
low_beam_light	break_light	w#p#s	NFW	NFW	NFW	NFW
low_beam_light	rear_fog_light	w#p#s	NFW	NFW	NFW	NFW
low_beam_light	steering_wheel	w#p#s	NFW	NFW	NFW	NFW
low_beam_light	tire	w#p#s	NFW	NFW	NFW	NFW
low_beam_light	tyre	w#p#s	NFW	NFW	NFW	NFW
low_beam_light	wheel_rim	w#p#s	NFW	NFW	NFW	NFW
low_beam_light	fuel_injector	w#p#s	NFW	NFW	NFW	NFW
low_beam_light	spark_plug	w#p#s	NFW	NFW	NFW	NFW
v-belt	v-belt	16	NFW	NFW	NFW	NFW
v-belt	oil_sump	w#p#s	NFW	NFW	NFW	NFW
v-belt	engine_block	w#p#s	NFW	NFW	NFW	NFW
v-belt	cylinder_block	w#p#s	NFW	NFW	NFW	NFW

Concept 1	**Concept 2**	**HSO**	**JCN**	LCH	**RES**	**WUP**
v-belt	speed_indicator	w#p#s	NFW	NFW	NFW	NFW
v-belt	rotation_speed	w#p#s	NFW	NFW	NFW	NFW
v-belt	speedometer	0	NFW	NFW	NFW	NFW
v-belt	paint	0	NFW	NFW	NFW	NFW
v-belt	brake	0	NFW	NFW	NFW	NFW
v-belt	battery	0	NFW	NFW	NFW	NFW
v-belt	window	0	NFW	NFW	NFW	NFW
v-belt	bolt	0	NFW	NFW	NFW	NFW
v-belt	mirror	0	NFW	NFW	NFW	NFW
v-belt	car	0	NFW	NFW	NFW	NFW
v-belt	engine	0	NFW	NFW	NFW	NFW
v-belt	motor	0	NFW	NFW	NFW	NFW
v-belt	break_light	w#p#s	NFW	NFW	NFW	NFW
v-belt	rear_fog_light	w#p#s	NFW	NFW	NFW	NFW
v-belt	steering_wheel	w#p#s	NFW	NFW	NFW	NFW
v-belt	tire	0	NFW	NFW	NFW	NFW
v-belt	tyre	0	NFW	NFW	NFW	NFW
v-belt	wheel_rim	w#p#s	NFW	NFW	NFW	NFW
v-belt	fuel_injector	w#p#s	NFW	NFW	NFW	NFW
v-belt	spark_plug	w#p#s	NFW	NFW	NFW	NFW
oil_sump	oil_sump	w#p#s	NFW	NFW	NFW	NFW
oil_sump	engine_block	w#p#s	NFW	NFW	NFW	NFW
oil_sump	cylinder_block	w#p#s	NFW	NFW	NFW	NFW

Concept 1	**Concept 2**	**HSO**	**JCN**	LCH	**RES**	**WUP**
oil_sump	speed_indicator	w#p#s	NFW	NFW	NFW	NFW
oil_sump	rotation_speed	w#p#s	NFW	NFW	NFW	NFW
oil_sump	speedometer	w#p#s	NFW	NFW	NFW	NFW
oil_sump	paint	w#p#s	NFW	NFW	NFW	NFW
oil_sump	brake	w#p#s	NFW	NFW	NFW	NFW
oil_sump	battery	w#p#s	NFW	NFW	NFW	NFW
oil_sump	window	w#p#s	NFW	NFW	NFW	NFW
oil_sump	bolt	w#p#s	NFW	NFW	NFW	NFW
oil_sump	mirror	w#p#s	NFW	NFW	NFW	NFW
oil_sump	car	w#p#s	NFW	NFW	NFW	NFW
oil_sump	engine	w#p#s	NFW	NFW	NFW	NFW
oil_sump	motor	w#p#s	NFW	NFW	NFW	NFW
oil_sump	break_light	w#p#s	NFW	NFW	NFW	NFW
oil_sump	rear_fog_light	w#p#s	NFW	NFW	NFW	NFW
oil_sump	steering_wheel	w#p#s	NFW	NFW	NFW	NFW
oil_sump	tire	w#p#s	NFW	NFW	NFW	NFW
oil_sump	tyre	w#p#s	NFW	NFW	NFW	NFW
oil_sump	wheel_rim	w#p#s	NFW	NFW	NFW	NFW
oil_sump	fuel_injector	w#p#s	NFW	NFW	NFW	NFW
oil_sump	spark_plug	w#p#s	NFW	NFW	NFW	NFW
engine_block	engine_block	w#p#s	0	3.58351893845611	0	1
engine_block	cylinder_block	w#p#s	0	3.58351893845611	0	1
engine_block	speed_indicator	w#p#s	0	1.09861228866811	2.55801504707402	0.421052631578947

Concept 1	**Concept 2**	**HSO**	**JCN**	LCH	**RES**	**WUP**
engine_block	rotation_speed	w#p#s	NFW	NFW	NFW	NFW
engine_block	speedometer	w#p#s	0	1.09861228866811	2.55801504707402	0.421052631578947
engine_block	paint	w#p#s	0	1.38629436111989	2.55801504707402	0.533333333333333
engine_block	brake	w#p#s	0	1.28093384546206	2.55801504707402	0.470588235294118
engine_block	battery	w#p#s	0	0.750305594399894	0	0.111111111111111
engine_block	window	w#p#s	0	1.28093384546206	2.55801504707402	0.470588235294118
engine_block	bolt	w#p#s	0	0.810930216216329	0	0.117647058823529
engine_block	mirror	w#p#s	0	1.28093384546206	2.55801504707402	0.470588235294118
engine_block	car	w#p#s	0	1.09861228866811	2.55801504707402	0.421052631578947
engine_block	engine	w#p#s	0	1.18562366565774	2.55801504707402	0.444444444444444
engine_block	motor	w#p#s	0	1.28093384546206	2.55801504707402	0.470588235294118
engine_block	break_light	w#p#s	NFW	NFW	NFW	NFW
engine_block	rear_fog_light	w#p#s	NFW	NFW	NFW	NFW
engine_block	steering_wheel	w#p#s	0	1.09861228866811	2.55801504707402	0.421052631578947
engine_block	tire	w#p#s	0	1.28093384546206	2.55801504707402	0.470588235294118
engine_block	tyre	w#p#s	0	1.01856958099457	0.855458788752587	0.25
engine_block	wheel_rim	w#p#s	NFW	NFW	NFW	NFW
engine_block	fuel_injector	w#p#s	NFW	NFW	NFW	NFW
engine_block	spark_plug	w#p#s	0	1.28093384546206	2.55801504707402	0.470588235294118
cylinder_block	cylinder_block	w#p#s	0	3.58351893845611	0	1
cylinder_block	speed_indicator	w#p#s	0	1.09861228866811	2.55801504707402	0.421052631578947
cylinder_block	rotation_speed	w#p#s	NFW	NFW	NFW	NFW
cylinder_block	speedometer	w#p#s	0	1.09861228866811	2.55801504707402	0.421052631578947

Concept 1	**Concept 2**	**HSO**	**JCN**	LCH	**RES**	**WUP**
cylinder_block	paint	w#p#s	0	1.38629436111989	2.55801504707402	0.533333333333333
cylinder_block	brake	w#p#s	0	1.28093384546206	2.55801504707402	0.470588235294118
cylinder_block	battery	w#p#s	0	0.750305594399894	0	0.111111111111111
cylinder_block	window	w#p#s	0	1.28093384546206	2.55801504707402	0.470588235294118
cylinder_block	bolt	w#p#s	0	0.810930216216329	0	0.117647058823529
cylinder_block	mirror	w#p#s	0	1.28093384546206	2.55801504707402	0.470588235294118
cylinder_block	car	w#p#s	0	1.09861228866811	2.55801504707402	0.421052631578947
cylinder_block	engine	w#p#s	0	1.18562366565774	2.55801504707402	0.444444444444444
cylinder_block	motor	w#p#s	0	1.28093384546206	2.55801504707402	0.470588235294118
cylinder_block	break_light	w#p#s	NFW	NFW	NFW	NFW
cylinder_block	rear_fog_light	w#p#s	NFW	NFW	NFW	NFW
cylinder_block	steering_wheel	w#p#s	0	1.09861228866811	2.55801504707402	0.421052631578947
cylinder_block	tire	w#p#s	0	1.28093384546206	2.55801504707402	0.470588235294118
cylinder_block	tyre	w#p#s	0	1.01856958099457	0.855458788752587	0.25
cylinder_block	wheel_rim	w#p#s	NFW	NFW	NFW	NFW
cylinder_block	fuel_injector	w#p#s	NFW	NFW	NFW	NFW
cylinder_block	spark_plug	w#p#s	0	1.28093384546206	2.55801504707402	0.470588235294118
speed_indicator	speed_indicator	w#p#s	0	3.58351893845611	0	1.1
speed_indicator	rotation_speed	w#p#s	NFW	NFW	NFW	NFW
speed_indicator	speedometer	w#p#s	0	3.58351893845611	0	1.1
speed_indicator	paint	w#p#s	0	1.28093384546206	2.55801504707402	0.5
speed_indicator	brake	w#p#s	0	1.6376087894008	4.68535709642899	0.666666666666667
speed_indicator	battery	w#p#s	0	0.693147180559945	0	0.105263157894737

Concept 1	**Concept 2**	**HSO**	**JCN**	LCH	**RES**	**WUP**
speed_indicator	window	w#p#s	0	1.18562366565774	2.55801504707402	0.444444444444444
speed_indicator	bolt	w#p#s	0	0.750305594399894	0	0.111111111111111
speed_indicator	mirror	w#p#s	0	1.6376087894008	4.68535709642899	0.666666666666667
speed_indicator	car	w#p#s	0	1.18562366565774	3.73884299593615	0.5
speed_indicator	engine	w#p#s	0	1.50407739677627	4.68535709642899	0.631578947368421
speed_indicator	motor	w#p#s	0	1.6376087894008	4.68535709642899	0.666666666666667
speed_indicator	break_light	w#p#s	NFW	NFW	NFW	NFW
speed_indicator	rear_fog_light	w#p#s	NFW	NFW	NFW	NFW
speed_indicator	steering_wheel	w#p#s	0	1.38629436111989	4.68535709642899	0.6
speed_indicator	tire	w#p#s	0	1.18562366565774	2.55801504707402	0.444444444444444
speed_indicator	tyre	w#p#s	0	0.944461608840852	0.855458788752587	0.235294117647059
speed_indicator	wheel_rim	w#p#s	NFW	NFW	NFW	NFW
speed_indicator	fuel_injector	w#p#s	NFW	NFW	NFW	NFW
speed_indicator	spark_plug	w#p#s	0	1.6376087894008	4.68535709642899	0.666666666666667
rotation_speed	rotation_speed	w#p#s	NFW	NFW	NFW	NFW
rotation_speed	speedometer	w#p#s	NFW	NFW	NFW	NFW
rotation_speed	paint	w#p#s	NFW	NFW	NFW	NFW
rotation_speed	brake	w#p#s	NFW	NFW	NFW	NFW
rotation_speed	battery	w#p#s	NFW	NFW	NFW	NFW
rotation_speed	window	w#p#s	NFW	NFW	NFW	NFW
rotation_speed	bolt	w#p#s	NFW	NFW	NFW	NFW
rotation_speed	mirror	w#p#s	NFW	NFW	NFW	NFW
rotation_speed	car	w#p#s	NFW	NFW	NFW	NFW

Concept 1	**Concept 2**	**HSO**	**JCN**	LCH	**RES**	**WUP**
rotation_speed	engine	w#p#s	NFW	NFW	NFW	NFW
rotation_speed	motor	w#p#s	NFW	NFW	NFW	NFW
rotation_speed	break_light	w#p#s	NFW	NFW	NFW	NFW
rotation_speed	rear_fog_light	w#p#s	NFW	NFW	NFW	NFW
rotation_speed	steering_wheel	w#p#s	NFW	NFW	NFW	NFW
rotation_speed	tire	w#p#s	NFW	NFW	NFW	NFW
rotation_speed	tyre	w#p#s	NFW	NFW	NFW	NFW
rotation_speed	wheel_rim	w#p#s	NFW	NFW	NFW	NFW
rotation_speed	fuel_injector	w#p#s	NFW	NFW	NFW	NFW
rotation_speed	spark_plug	w#p#s	NFW	NFW	NFW	NFW
speedometer	speedometer	16	0	3.58351893845611	0	1.1
speedometer	paint	0	0	1.28093384546206	2.55801504707402	0.5
speedometer	brake	0	0	1.6376087894008	4.68535709642899	0.666666666666667
speedometer	battery	0	0	0.693147180559945	0	0.105263157894737
speedometer	window	0	0	1.18562366565774	2.55801504707402	0.444444444444444
speedometer	bolt	0	0	0.750305594399894	0	0.111111111111111
speedometer	mirror	0	0	1.6376087894008	4.68535709642899	0.666666666666667
speedometer	car	6	0	1.18562366565774	3.73884299593615	0.5
speedometer	engine	0	0	1.50407739677627	4.68535709642899	0.631578947368421
speedometer	motor	0	0	1.6376087894008	4.68535709642899	0.666666666666667
speedometer	break_light	w#p#s	NFW	NFW	NFW	NFW
speedometer	rear_fog_light	w#p#s	NFW	NFW	NFW	NFW
speedometer	steering_wheel	w#p#s	0	1.38629436111989	4.68535709642899	0.6

Concept 1	**Concept 2**	**HSO**	**JCN**	LCH	**RES**	**WUP**
speedometer	tire	0	0	1.18562366565774	2.55801504707402	0.444444444444444
speedometer	tyre	0	0	0.944461608840852	0.855458788752587	0.235294117647059
speedometer	wheel_rim	w#p#s	NFW	NFW	NFW	NFW
speedometer	fuel_injector	w#p#s	NFW	NFW	NFW	NFW
speedometer	spark_plug	w#p#s	0	1.6376087894008	4.68535709642899	0.666666666666667
paint	paint	16	29590099.4292139	3.58351893845611	9.50673776454438	1
paint	brake	0	0.0664779678945023	1.50407739677627	2.55801504707402	0.571428571428571
paint	battery	0	0.0524659094307516	0.944461608840852	0	0.133333333333333
paint	window	0	0.0790846272853159	1.50407739677627	2.55801504707402	0.571428571428571
paint	bolt	0	0.0467041559016792	1.01856958099457	0	0.142857142857143
paint	mirror	0	0.0670733712104102	1.50407739677627	2.55801504707402	0.571428571428571
paint	car	2	0.0952445994738899	1.28093384546206	2.55801504707402	0.5
paint	engine	0	0.0730318766155491	1.38629436111989	2.55801504707402	0.533333333333333
paint	motor	0	0.0745992030076069	1.50407739677627	2.55801504707402	0.571428571428571
paint	break_light	w#p#s	NFW	NFW	NFW	NFW
paint	rear_fog_light	w#p#s	NFW	NFW	NFW	NFW
paint	steering_wheel	w#p#s	0.0658036369686689	1.28093384546206	2.55801504707402	0.5
paint	tire	0	0.0676074773990962	1.50407739677627	2.55801504707402	0.571428571428571
paint	tyre	0	0	1.28093384546206	0.855458788752587	0.307692307692308
paint	wheel_rim	w#p#s	NFW	NFW	NFW	NFW
paint	fuel_injector	w#p#s	NFW	NFW	NFW	NFW
paint	spark_plug	w#p#s	0	1.50407739677627	2.55801504707402	0.571428571428571
brake	brake	16	29590099.4292139	3.58351893845611	10.6518700688474	1

Concept 1	**Concept 2**	**HSO**	**JCN**	LCH	**RES**	**WUP**
brake	battery	0	0.0494923866590715	0.810930216216329	0	0.117647058823529
brake	window	2	0.0725172902337492	1.38629436111989	2.55801504707402	0.5
brake	bolt	0	0.0443331129516069	0.8754687373539	0	0.125
brake	mirror	3	0.0847493929146747	1.97408102602201	4.68535709642899	0.75
brake	car	5	0.107726588704472	1.38629436111989	3.73884299593615	0.555555555555556
brake	engine	2	0.0944902791202106	1.79175946922806	4.68535709642899	0.705882352941177
brake	motor	3	0.0971305955749447	1.97408102602201	4.68535709642899	0.75
brake	break_light	w#p#s	NFW	NFW	NFW	NFW
brake	rear_fog_light	w#p#s	NFW	NFW	NFW	NFW
brake	steering_wheel	w#p#s	0.0827323063981283	1.6376087894008	4.68535709642899	0.666666666666667
brake	tire	0	0.0627494462450517	1.38629436111989	2.55801504707402	0.5
brake	tyre	0	0	1.09861228866811	0.855458788752587	0.266666666666667
brake	wheel_rim	w#p#s	NFW	NFW	NFW	NFW
brake	fuel_injector	w#p#s	NFW	NFW	NFW	NFW
brake	spark_plug	w#p#s	0	1.97408102602201	4.68535709642899	0.75
battery	battery	16	29590099.4292139	3.58351893845611	9.55325778017928	1
battery	window	0	0.0561569573328843	0.810930216216329	0	0.117647058823529
battery	bolt	0	0.046602902797111	0.810930216216329	0	0.117647058823529
battery	mirror	0	0.0498216473299529	0.810930216216329	0	0.117647058823529
battery	car	0	0.0638494865663311	0.693147180559945	0	0.105263157894737
battery	engine	0	0.0530357641371822	0.750305594399894	0	0.111111111111111
battery	motor	0	0.0538574915037761	0.810930216216329	0	0.117647058823529
battery	break_light	w#p#s	NFW	NFW	NFW	NFW

Concept 1	**Concept 2**	**HSO**	**JCN**	LCH	**RES**	**WUP**
battery	rear_fog_light	w#p#s	NFW	NFW	NFW	NFW
battery	steering_wheel	w#p#s	0.0491176540751561	0.693147180559945	0	0.105263157894737
battery	tire	0	0.0501157337153565	0.810930216216329	0	0.117647058823529
battery	tyre	0	0	0.8754687373539	0	0.125
battery	wheel_rim	w#p#s	NFW	NFW	NFW	NFW
battery	fuel_injector	w#p#s	NFW	NFW	NFW	NFW
battery	spark_plug	w#p#s	0	0.810930216216329	0	0.117647058823529
window	window	16	29590099.4292139	3.58351893845611	8.25397479604901	1
window	bolt	0	0.0496066002307733	0.8754687373539	0	0.125
window	mirror	0	0.0732263656451295	1.38629436111989	2.55801504707402	0.5
window	car	2	0.108148795573475	1.18562366565774	2.55801504707402	0.444444444444444
window	engine	0	0.0803865754800409	1.28093384546206	2.55801504707402	0.470588235294118
window	motor	0	0.0822895860665053	1.38629436111989	2.55801504707402	0.5
window	break_light	w#p#s	NFW	NFW	NFW	NFW
window	rear_fog_light	w#p#s	NFW	NFW	NFW	NFW
window	steering_wheel	w#p#s	0.071715610852917	1.18562366565774	2.55801504707402	0.444444444444444
window	tire	2	0.0738634246559206	1.38629436111989	2.55801504707402	0.5
window	tyre	0	0	1.09861228866811	0.855458788752587	0.266666666666667
window	wheel_rim	w#p#s	NFW	NFW	NFW	NFW
window	fuel_injector	w#p#s	NFW	NFW	NFW	NFW
window	spark_plug	w#p#s	0	1.38629436111989	2.55801504707402	0.5
bolt	bolt	16	29590099.4292139	3.58351893845611	11.9046330373428	1
bolt	mirror	0	0.0445971217720001	0.8754687373539	0	0.125

Concept 1	**Concept 2**	**HSO**	**JCN**	LCH	**RES**	**WUP**
bolt	car	0	0.0555148190160038	0.750305594399894	0	0.111111111111111
bolt	engine	0	0.0471551834508911	0.810930216216329	0	0.117647058823529
bolt	motor	0	0.047803673791564	0.8754687373539	0	0.125
bolt	break_light	w#p#s	NFW	NFW	NFW	NFW
bolt	rear_fog_light	w#p#s	NFW	NFW	NFW	NFW
bolt	steering_wheel	w#p#s	0.0440321978143881	0.750305594399894	0	0.111111111111111
bolt	tire	0	0.0448326178469132	0.8754687373539	0	0.125
bolt	tyre	0	0	0.944461608840852	0	0.133333333333333
bolt	wheel_rim	w#p#s	NFW	NFW	NFW	NFW
bolt	fuel_injector	w#p#s	NFW	NFW	NFW	NFW
bolt	spark_plug	w#p#s	0	0.8754687373539	0	0.125
mirror	mirror	16	29590099.4292139	3.58351893845611	10.5183386762229	1
mirror	car	6	0.10929883954991	1.38629436111989	3.73884299593615	0.555555555555556
mirror	engine	2	0.0956977375299088	1.79175946922806	4.68535709642899	0.705882352941177
mirror	motor	3	0.0984069318767201	1.97408102602201	4.68535709642899	0.75
mirror	break_light	w#p#s	NFW	NFW	NFW	NFW
mirror	rear_fog_light	w#p#s	NFW	NFW	NFW	NFW
mirror	steering_wheel	w#p#s	0.0836564897642843	1.6376087894008	4.68535709642899	0.666666666666667
mirror	tire	0	0.0632796679077497	1.38629436111989	2.55801504707402	0.5
mirror	tyre	0	0	1.09861228866811	0.855458788752587	0.266666666666667
mirror	wheel_rim	w#p#s	NFW	NFW	NFW	NFW
mirror	fuel_injector	w#p#s	NFW	NFW	NFW	NFW
mirror	spark_plug	w#p#s	0	1.97408102602201	4.68535709642899	0.75

Concept 1	**Concept 2**	**HSO**	**JCN**	LCH	**RES**	**WUP**
car	car	16	29590099.4292139	3.58351893845611	6.10857528657738	1.1
car	engine	6	0.126058375935633	1.28093384546206	3.73884299593615	0.526315789473684
car	motor	5	0.130801871107812	1.38629436111989	3.73884299593615	0.555555555555556
car	break_light	w#p#s	NFW	NFW	NFW	NFW
car	rear_fog_light	w#p#s	NFW	NFW	NFW	NFW
car	steering_wheel	w#p#s	0.10596688909857	1.18562366565774	3.73884299593615	0.5
car	tire	4	0.0877724172425366	1.18562366565774	2.55801504707402	0.444444444444444
car	tyre	0	0	0.944461608840852	0.855458788752587	0.235294117647059
car	wheel_rim	w#p#s	NFW	NFW	NFW	NFW
car	fuel_injector	w#p#s	NFW	NFW	NFW	NFW
car	spark_plug	w#p#s	0	1.38629436111989	3.73884299593615	0.555555555555556
motor	motor	16	29590099.4292139	3.58351893845611	9.01426127944659	1
motor	break_light	w#p#s	NFW	NFW	NFW	NFW
motor	rear_fog_light	w#p#s	NFW	NFW	NFW	NFW
motor	steering_wheel	w#p#s	0.0956977375299088	1.6376087894008	4.68535709642899	0.666666666666667
motor	tire	0	0.0699360031153904	1.38629436111989	2.55801504707402	0.5
motor	tyre	0	0	1.09861228866811	0.855458788752587	0.266666666666667
motor	wheel_rim	w#p#s	NFW	NFW	NFW	NFW
motor	fuel_injector	w#p#s	NFW	NFW	NFW	NFW
motor	spark_plug	w#p#s	0	1.97408102602201	4.68535709642899	0.75
break_light	break_light	w#p#s	NFW	NFW	NFW	NFW
break_light	rear_fog_light	w#p#s	NFW	NFW	NFW	NFW
break_light	steering_wheel	w#p#s	NFW	NFW	NFW	NFW

Concept 1	**Concept 2**	**HSO**	**JCN**	LCH	**RES**	**WUP**
break_light	tire	w#p#s	NFW	NFW	NFW	NFW
break_light	tyre	w#p#s	NFW	NFW	NFW	NFW
break_light	wheel_rim	w#p#s	NFW	NFW	NFW	NFW
break_light	fuel_injector	w#p#s	NFW	NFW	NFW	NFW
break_light	spark_plug	w#p#s	NFW	NFW	NFW	NFW
rear_fog_light	rear_fog_light	w#p#s	NFW	NFW	NFW	NFW
rear_fog_light	steering_wheel	w#p#s	NFW	NFW	NFW	NFW
rear_fog_light	tire	w#p#s	NFW	NFW	NFW	NFW
rear_fog_light	tyre	w#p#s	NFW	NFW	NFW	NFW
rear_fog_light	wheel_rim	w#p#s	NFW	NFW	NFW	NFW
rear_fog_light	fuel_injector	w#p#s	NFW	NFW	NFW	NFW
rear_fog_light	spark_plug	w#p#s	NFW	NFW	NFW	NFW
steering_wheel	steering_wheel	w#p#s	29590099.4292139	3.58351893845611	10.8060207486746	1.1
steering_wheel	tire	w#p#s	0.0621482938901184	1.18562366565774	2.55801504707402	0.444444444444444
steering_wheel	tyre	w#p#s	0	0.944461608840852	0.855458788752587	0.235294117647059
steering_wheel	wheel_rim	w#p#s	NFW	NFW	NFW	NFW
steering_wheel	fuel_injector	w#p#s	NFW	NFW	NFW	NFW
steering_wheel	spark_plug	w#p#s	0	1.6376087894008	4.68535709642899	0.666666666666667
tire	tire	16	29590099.4292139	3.58351893845611	10.4005556405665	1
tire	tyre	0	0	1.09861228866811	0.855458788752587	0.266666666666667
tire	wheel_rim	w#p#s	NFW	NFW	NFW	NFW
tire	fuel_injector	w#p#s	NFW	NFW	NFW	NFW
tire	spark_plug	w#p#s	0	1.38629436111989	2.55801504707402	0.5

Concept 1	**Concept 2**	**HSO**	**JCN**	LCH	**RES**	**WUP**
tyre	tyre	16	0	3.58351893845611	0	1
tyre	wheel_rim	w#p#s	NFW	NFW	NFW	NFW
tyre	fuel_injector	w#p#s	NFW	NFW	NFW	NFW
tyre	spark_plug	w#p#s	0	1.09861228866811	0.855458788752587	0.266666666666667
wheel_rim	wheel_rim	w#p#s	NFW	NFW	NFW	NFW
wheel_rim	fuel_injector	w#p#s	NFW	NFW	NFW	NFW
wheel_rim	spark_plug	w#p#s	NFW	NFW	NFW	NFW
fuel_injector	fuel_injector	w#p#s	NFW	NFW	NFW	NFW
fuel_injector	spark_plug	w#p#s	NFW	NFW	NFW	NFW
spark_plug	spark_plug	w#p#s	0	3.58351893845611	0	1

Table C.1: Comparison among LCH, WUP, JCN, HSO and RES WordNet-based Semantic Similarity Methods

Table C.2 presents the same results as table C.1, however in this table the concepts not found in WordNet were removed and the values were normalised in order to have a value between zero and one.

Palavra 1	**Palavra 2**	**LCH**	**WUP**	**JCN**	**LIN**	**HSO**	**RES**
automobile	car	1,00	1,00	1,00	1,00	1,00	1,00
automobile	engine	0,61	0,84	0,00	0,65	0,38	0,99
automobile	wheel	0,55	0,78	0,00	0,69	0,31	0,97
automobile	stoplight	0,21	0,11	0,00	0,00	0,31	0,00
automobile	headlight	0,36	0,53	0,00	0,44	1,00	0,61
automobile	engine_block	0,31	0,42	0,00	0,00	1,00	0,42
automobile	cylinder_block	0,31	0,42	0,00	0,00	1,00	0,42
automobile	speed_indicator	0,33	0,50	0,00	0,00	1,00	0,61
automobile	speedometer	0,33	0,50	0,00	0,00	1,00	0,61
automobile	paint	0,36	0,53	0,00	0,33	0,13	0,61
automobile	brake	0,39	0,56	0,00	0,45	0,31	0,61
automobile	battery	0,39	0,56	0,00	0,47	0,19	0,61
automobile	window	0,39	0,53	0,00	0,36	0,38	0,61
automobile	bolt	0,39	0,56	0,00	0,40	0,25	0,61
automobile	mirror	0,39	0,56	0,00	0,45	0,38	0,61
automobile	motor	0,39	0,56	0,00	0,49	0,31	0,61
automobile	steering_wheel	0,33	0,50	0,00	0,44	1,00	0,61
automobile	tire	0,33	0,44	0,00	0,31	1,00	0,42
automobile	tyre	0,33	0,44	0,00	0,31	0,25	0,42
automobile	spark_plug	0,39	0,56	0,00	0,00	1,00	0,61
engine	car	0,61	0,84	0,00	0,65	0,38	0,67
engine	automobile	0,61	0,84	0,00	0,65	0,13	0,97
engine	wheel	0,61	0,82	0,00	0,50	0,13	0,65

Palavra 1	**Palavra 2**	**LCH**	**WUP**	**JCN**	**LIN**	**HSO**	**RES**
engine	stoplight	0,33	0,17	0,00	0,00	0,00	0,00
engine	headlight	0,46	0,67	0,00	0,47	0,00	0,52
engine	engine_block	0,36	0,44	0,00	0,00	0,00	0,28
engine	cylinder_block	0,36	0,44	0,00	0,00	0,00	0,28
engine	speed_indicator	0,42	0,63	0,00	0,00	0,00	0,52
engine	speedometer	0,42	0,63	0,00	0,00	0,00	0,52
engine	paint	0,46	0,56	0,00	0,27	0,00	0,41
engine	brake	0,50	0,71	0,00	0,47	0,13	0,52
engine	battery	0,50	0,71	0,00	0,49	0,13	0,52
engine	window	0,46	0,67	0,00	0,29	0,00	0,52
engine	bolt	0,42	0,63	0,00	0,34	0,00	0,52
engine	mirror	0,50	0,71	0,00	0,47	0,13	0,52
engine	motor	0,81	0,94	1,00	0,98	0,25	1,00
engine	steering_wheel	0,42	0,63	0,00	0,47	0,00	0,52
engine	tire	0,39	0,47	0,00	0,26	0,00	0,28
engine	tyre	0,42	0,47	0,00	0,26	0,00	0,28
engine	spark_plug	0,50	0,71	0,00	0,00	0,13	0,52
wheel	automobile	0,42	0,78	0,00	0,69	0,00	0,77
wheel	engine	0,42	0,82	0,00	0,50	0,00	0,52
wheel	stoplight	0,24	0,13	0,00	0,00	0,00	0,00
wheel	headlight	0,42	0,63	0,00	0,45	0,00	0,43
wheel	cylinder_block	0,36	0,47	0,00	0,00	0,19	0,24
wheel	speed_indicator	0,46	0,70	0,00	0,00	0,00	0,54

Palavra 1	**Palavra 2**	**LCH**	**WUP**	**JCN**	**LIN**	**HSO**	**RES**
wheel	speedometer	0,46	0,70	0,00	0,00	0,00	0,54
wheel	paint	0,42	0,59	0,00	0,26	0,00	0,35
wheel	brake	0,46	0,67	0,00	0,46	0,00	0,43
wheel	battery	0,46	0,67	0,00	0,48	0,00	0,43
wheel	window	0,46	0,63	0,00	0,28	0,00	0,43
wheel	bolt	0,46	0,63	0,00	0,33	0,00	0,43
wheel	mirror	0,46	0,67	0,00	0,46	0,00	0,43
wheel	car	0,69	0,88	0,00	0,69	0,38	0,55
wheel	motor	0,46	0,67	0,00	0,50	0,00	0,43
wheel	steering_wheel	1,00	1,00	1,00	1,00	1,00	1,00
wheel	tire	0,39	0,50	0,00	0,25	0,25	0,24
wheel	tyre	0,39	0,50	0,00	0,25	0,25	0,24
wheel	spark_plug	0,46	0,67	0,00	0,00	0,00	0,43
stoplight	automobile	0,23	0,11	0,00	0,00	0,00	0,00
stoplight	wheel	0,23	0,13	0,00	0,00	0,00	0,00
stoplight	headlight	0,23	0,12	0,00	0,00	0,00	0,00
stoplight	engine_block	0,23	0,12	0,00	0,00	0,00	0,00
stoplight	cylinder_block	0,23	0,12	0,00	0,00	0,00	0,00
stoplight	speed_indicator	0,21	0,11	0,00	0,00	0,00	0,00
stoplight	speedometer	0,21	0,11	0,00	0,00	0,00	0,00
stoplight	paint	0,28	0,14	0,00	0,00	0,00	0,00
stoplight	brake	0,28	0,14	0,00	0,00	0,00	0,00
stoplight	battery	0,33	0,17	0,00	0,00	0,00	0,00

Palavra 1	**Palavra 2**	**LCH**	**WUP**	**JCN**	**LIN**	**HSO**	**RES**
stoplight	window	0,33	0,29	0,00	0,14	0,00	0,28
stoplight	bolt	0,31	0,25	0,00	0,00	0,00	0,28
stoplight	mirror	0,24	0,13	0,00	0,00	0,00	0,00
stoplight	car	0,24	0,13	0,00	0,00	0,31	0,00
stoplight	engine	0,33	0,17	0,00	0,00	0,00	0,00
stoplight	motor	0,31	0,15	0,00	0,00	0,00	0,00
stoplight	steering_wheel	0,21	0,11	0,00	0,00	0,00	0,00
stoplight	tire	0,24	0,13	0,00	0,00	0,00	0,00
stoplight	tyre	0,26	0,13	0,00	0,00	0,00	0,00
stoplight	spark_plug	0,24	0,13	0,00	0,00	0,00	0,00
healight	automobile	0,39	0,53	0,00	0,44	0,00	0,61
headlight	wheel	0,39	0,63	0,00	0,45	0,00	0,35
headlight	stoplight	0,42	0,12	0,00	0,00	1,00	0,77
headlight	engine_block	0,33	0,44	0,00	0,00	1,00	0,42
headlight	cylinder_block	0,33	0,44	0,00	0,00	1,00	0,42
headlight	speed_indicator	0,42	0,63	0,00	0,00	1,00	0,77
headlight	speedometer	0,42	0,63	0,00	0,00	1,00	0,77
headlight	paint	0,39	0,56	0,00	0,25	0,00	0,61
headlight	brake	0,50	0,71	0,00	0,44	0,13	0,77
headlight	battery	0,50	0,71	0,00	0,45	0,13	0,77
headlight	window	0,46	0,67	0,00	0,27	0,00	0,77
headlight	bolt	0,42	0,63	0,00	0,32	0,00	0,77
headlight	mirror	0,50	0,71	0,00	0,44	0,13	0,77

Palavra 1	**Palavra 2**	**LCH**	**WUP**	**JCN**	**LIN**	**HSO**	**RES**
headlight	car	0,42	0,59	0,00	0,44	0,31	0,61
headlight	engine	0,46	0,67	0,00	0,47	0,00	0,77
headlight	motor	0,50	0,71	0,00	0,47	0,13	0,77
headlight	steering_wheel	0,42	0,63	0,00	0,43	1,00	0,77
headlight	tire	0,36	0,47	0,00	0,24	1,00	0,42
headlight	tyre	0,36	0,47	0,00	0,24	0,00	0,42
headlight	spark_plug	0,50	0,71	0,00	0,00	1,00	0,77
engine_block	automobile	0,36	0,42	0,00	0,00	0,00	0,42
engine_block	stoplight	0,31	0,44	0,00	0,00	1,00	0,42
engine_block	headlight	0,31	0,44	0,00	0,00	1,00	0,42
engine_block	cylinder_block	1,00	1,00	0,00	0,00	1,00	0,00
engine_block	speed_indicator	0,31	0,42	0,00	0,00	1,00	0,42
engine_block	speedometer	0,31	0,42	0,00	0,00	1,00	0,42
engine_block	paint	0,39	0,50	0,00	0,00	0,00	0,42
engine_block	brake	0,36	0,47	0,00	0,00	0,00	0,42
engine_block	battery	0,36	0,47	0,00	0,00	0,00	0,42
engine_block	window	0,42	0,53	0,00	0,00	0,00	0,42
engine_block	bolt	0,36	0,47	0,00	0,00	0,00	0,42
engine_block	mirror	0,36	0,47	0,00	0,00	0,00	0,42
engine_block	car	0,36	0,47	0,00	0,00	0,31	0,42
engine_block	engine	0,36	0,44	0,00	0,00	0,00	0,42
engine_block	motor	0,36	0,47	0,00	0,00	0,00	0,42
engine_block	steering_wheel	0,31	0,42	0,00	0,00	1,00	0,42

Palavra 1	**Palavra 2**	**LCH**	**WUP**	**JCN**	**LIN**	**HSO**	**RES**
engine_block	tire	0,36	0,47	0,00	0,00	1,00	0,42
engine_block	tyre	0,36	0,47	0,00	0,00	0,00	0,42
engine_block	spark_plug	0,36	0,47	0,00	0,00	1,00	0,42
cylinder_block	automobile	0,36	0,42	0,00	0,00	0,00	0,42
cylinder_block	wheel	0,36	0,47	0,00	0,00	0,00	0,24
cylinder_block	stoplight	0,36	0,12	0,00	0,00	0,31	0,42
cylinder_block	headlight	0,36	0,44	0,00	0,00	0,00	0,42
cylinder_block	engine_block	0,36	1,00	0,00	0,00	0,00	0,42
cylinder_block	speed_indicator	0,31	0,42	0,00	0,00	1,00	0,42
cylinder_block	speedometer	0,31	0,42	0,00	0,00	1,00	0,42
cylinder_block	paint	0,39	0,50	0,00	0,00	0,00	0,42
cylinder_block	brake	0,36	0,47	0,00	0,00	0,00	0,42
cylinder_block	battery	0,36	0,47	0,00	0,00	0,00	0,42
cylinder_block	window	0,42	0,53	0,00	0,00	0,00	0,42
cylinder_block	bolt	0,36	0,47	0,00	0,00	0,00	0,42
cylinder_block	mirror	0,36	0,47	0,00	0,00	0,00	0,42
cylinder_block	car	0,36	0,47	0,00	0,00	0,31	0,42
cylinder_block	engine	0,36	0,44	0,00	0,00	0,00	0,42
cylinder_block	motor	0,36	0,47	0,00	0,00	0,00	0,42
cylinder_block	steering_wheel	0,31	0,42	0,00	0,00	1,00	0,42
cylinder_block	tire	0,36	0,47	0,00	0,00	1,00	0,42
cylinder_block	tyre	0,36	0,47	0,00	0,00	0,00	0,42
cylinder_block	spark_plug	0,36	0,47	0,00	0,00	1,00	0,42

Palavra 1	**Palavra 2**	**LCH**	**WUP**	**JCN**	**LIN**	**HSO**	**RES**
speed_indicator	automobile	0,46	0,50	0,00	0,00	0,00	0,77
speed_indicator	wheel	0,36	0,70	0,00	0,00	0,00	0,35
speed_indicator	stoplight	0,46	0,11	0,00	0,00	1,00	0,77
speed_indicator	headlight	0,33	0,63	0,00	0,00	1,00	0,42
speed_indicator	engine_block	0,46	0,42	0,00	0,00	0,00	0,77
speed_indicator	cylinder_block	0,39	0,42	0,00	0,00	0,00	0,77
speed_indicator	speedometer	1,00	1,00	0,00	0,00	1,00	0,00
speed_indicator	paint	0,36	0,53	0,00	0,00	0,00	0,61
speed_indicator	brake	0,46	0,67	0,00	0,00	0,00	0,77
speed_indicator	battery	0,46	0,67	0,00	0,00	0,00	0,77
speed_indicator	window	0,42	0,63	0,00	0,00	0,00	0,77
speed_indicator	bolt	0,39	0,60	0,00	0,00	0,00	0,77
speed_indicator	mirror	0,46	0,67	0,00	0,00	0,00	0,77
speed_indicator	car	0,39	0,56	0,00	0,00	0,38	0,61
speed_indicator	engine	0,42	0,63	0,00	0,00	0,00	0,77
speed_indicator	motor	0,46	0,67	0,00	0,00	0,00	0,77
speed_indicator	steering_wheel	0,39	0,60	0,00	0,00	1,00	0,77
speed_indicator	tire	0,33	0,44	0,00	0,00	1,00	0,42
speed_indicator	tyre	0,33	0,44	0,00	0,00	0,00	0,42
speed_indicator	spark_plug	0,46	0,67	0,00	0,00	1,00	0,77
speedometer	automobile	0,36	0,50	0,00	0,00	0,00	0,61
speedometer	wheel	0,42	0,70	0,00	0,00	0,00	0,35
speedometer	stoplight	0,46	0,11	0,00	0,00	0,00	0,61

Palavra 1	**Palavra 2**	**LCH**	**WUP**	**JCN**	**LIN**	**HSO**	**RES**
speedometer	headlight	0,42	0,63	0,00	0,00	0,00	0,61
speedometer	engine_block	0,50	0,42	0,00	0,00	0,13	0,61
speedometer	cylinder_block	0,33	0,42	0,00	0,00	0,00	0,42
speedometer	speed_indicator	0,39	1,00	0,00	0,00	0,38	0,61
speedometer	paint	0,36	0,53	0,00	0,00	0,00	0,61
speedometer	brake	0,46	0,67	0,00	0,00	0,00	0,77
speedometer	battery	0,46	0,67	0,00	0,00	0,00	0,77
speedometer	window	0,42	0,63	0,00	0,00	0,00	0,77
speedometer	bolt	0,39	0,60	0,00	0,00	0,00	0,77
speedometer	mirror	0,46	0,67	0,00	0,00	0,00	0,77
speedometer	car	0,39	0,56	0,00	0,00	0,38	0,61
speedometer	engine	0,42	0,63	0,00	0,00	0,00	0,77
speedometer	motor	0,46	0,67	0,00	0,00	0,00	0,77
speedometer	steering_wheel	0,39	0,60	0,00	0,00	1,00	0,77
speedometer	tire	0,33	0,44	0,00	0,00	1,00	0,42
speedometer	tyre	0,33	0,44	0,00	0,00	0,00	0,42
speedometer	spark_plug	0,46	0,67	0,00	0,00	1,00	0,77
paint	brake	0,42	0,59	0,00	0,25	0,00	0,61
paint	battery	0,42	0,59	0,00	0,26	0,00	0,61
paint	window	0,50	0,62	0,00	0,29	0,13	0,61
paint	bolt	0,42	0,59	0,00	0,23	0,00	0,61
paint	mirror	0,42	0,59	0,00	0,26	0,00	0,61
paint	car	0,42	0,59	0,00	0,33	0,13	0,61

Palavra 1	**Palavra 2**	**LCH**	**WUP**	**JCN**	**LIN**	**HSO**	**RES**
paint	engine	0,46	0,56	0,00	0,27	0,00	0,61
paint	motor	0,42	0,59	0,00	0,28	0,00	0,61
paint	steering_wheel	0,36	0,53	0,00	0,25	0,00	0,61
paint	tire	0,42	0,53	0,00	0,26	0,00	0,42
paint	tyre	0,42	0,53	0,00	0,26	0,00	0,61
paint	spark_plug	0,42	0,59	0,00	0,00	0,00	0,61
brake	battery	0,55	0,75	0,00	0,46	0,19	0,61
brake	window	0,50	0,71	0,00	0,27	0,13	0,56
brake	bolt	0,55	0,78	0,00	0,32	0,00	1,00
brake	mirror	0,55	0,75	0,00	0,44	0,19	0,56
brake	car	0,46	0,63	0,00	0,45	0,38	0,45
brake	engine	0,50	0,71	0,00	0,47	0,13	0,56
brake	motor	0,55	0,75	0,00	0,48	0,19	0,56
brake	steering_wheel	0,46	0,67	0,00	0,44	0,00	0,56
brake	tire	0,39	0,50	0,00	0,24	0,00	0,31
brake	tyre	0,39	0,50	0,00	0,24	0,00	0,31
brake	spark_plug	0,55	0,75	0,00	0,00	0,19	0,56
battery	window	0,50	0,71	0,00	0,28	0,00	0,60
battery	bolt	0,46	0,67	0,00	0,33	0,00	0,60
battery	mirror	0,55	0,75	0,00	0,46	0,19	0,60
battery	car	0,46	0,63	0,00	0,47	0,19	0,49
battery	engine	0,50	0,71	0,00	0,49	0,13	0,60
battery	motor	0,55	0,75	0,00	0,50	0,19	0,60

Palavra 1	**Palavra 2**	**LCH**	**WUP**	**JCN**	**LIN**	**HSO**	**RES**
battery	steering_wheel	0,46	0,67	0,00	0,45	0,00	0,60
battery	tire	0,39	0,50	0,00	0,25	0,00	0,33
battery	tyre	0,39	0,50	0,00	0,25	0,00	0,33
battery	spark_plug	0,69	0,88	0,00	0,00	0,31	1,00
window	bolt	0,46	0,63	0,00	0,25	0,13	0,76
window	mirror	0,50	0,71	0,00	0,27	0,00	0,60
window	car	0,46	0,59	0,00	0,37	0,38	0,49
window	engine	0,46	0,67	0,00	0,29	0,00	0,60
window	motor	0,50	0,71	0,00	0,30	0,00	0,60
window	steering_wheel	0,42	0,63	0,00	0,27	0,00	0,60
window	tire	0,46	0,57	0,00	0,27	0,13	0,33
window	tyre	0,46	0,57	0,00	0,27	0,13	0,41
window	spark_plug	0,50	0,71	0,00	0,00	0,00	0,60
bolt	mirror	0,46	0,67	0,00	0,32	0,00	0,77
bolt	car	0,46	0,63	0,00	0,40	0,25	0,61
bolt	engine	0,42	0,63	0,00	0,34	0,00	0,77
bolt	motor	0,46	0,67	0,00	0,35	0,00	0,77
bolt	steering_wheel	0,39	0,60	0,00	0,32	0,00	0,77
bolt	tire	0,39	0,50	0,00	0,22	0,00	0,42
bolt	tyre	0,39	0,50	0,00	0,22	0,00	0,42
bolt	spark_plug	0,46	0,67	0,00	0,00	0,00	0,77
mirror	car	0,46	0,63	0,00	0,45	0,38	0,61
mirror	engine	0,50	0,71	0,00	0,47	0,13	0,77

Palavra 1	**Palavra 2**	**LCH**	**WUP**	**JCN**	**LIN**	**HSO**	**RES**
mirror	motor	0,55	0,75	0,00	0,48	0,19	0,77
mirror	steering_wheel	0,46	0,67	0,00	0,44	0,00	0,77
mirror	tire	0,39	0,50	0,00	0,24	0,00	0,42
mirror	tyre	0,39	0,50	0,00	0,24	0,00	0,42
mirror	spark_plug	0,55	0,75	0,00	0,00	0,19	0,77
car	engine	0,61	0,84	0,00	0,65	0,38	0,99
car	motor	0,46	0,63	0,00	0,49	0,31	0,61
car	steering_wheel	0,39	0,56	0,00	0,44	0,00	0,61
car	tire	0,39	0,50	0,00	0,31	0,25	0,42
car	tyre	0,39	0,50	0,00	0,31	0,25	0,42
car	spark_plug	0,46	0,63	0,00	0,00	0,25	0,61
motor	steering_wheel	0,46	0,67	0,00	0,47	0,00	0,77
motor	tire	0,39	0,50	0,00	0,26	0,00	0,42
motor	tyre	0,39	0,50	0,00	0,26	0,00	0,42
motor	spark_plug	0,55	0,75	0,00	0,00	0,19	0,77
steering_wheel	tire	0,33	0,44	0,00	0,24	1,00	0,42
steering_wheel	tyre	0,33	0,44	0,00	0,24	0,00	0,42
steering_wheel	spark_plug	0,46	0,67	0,00	0,00	1,00	0,77
Tire	tyre	1,00	1,00	1,00	1,00	1,00	1,00
Tire	spark_plug	0,39	0,50	0,00	0,00	1,00	0,25
Tyre	spark_plug	0,39	0,50	0,00	0,00	0,00	0,25
spark_plug	spark_plug	1,00	1,00	0,00	0,00	1,00	0,00
suspension	suspension system	1,00	1,00	0,00	0,00	0,00	0,08

Palavra 1	**Palavra 2**	**LCH**	**WUP**	**JCN**	**LIN**	**HSO**	**RES**
throttle	accelerator	1,00	1,00	1,00	1,00	1,00	1,00
clutch	clutch pedal	1,00	1,00	0,00	0,00	0,00	0,33
antenna	aerial	1,00	1,00	1,00	1,00	1,00	1,00
antenna	transmitting aerial	1,00	1,00	1,00	1,00	1,00	1,00
aerial	transmitting aerial	1,00	1,00	1,00	1,00	1,00	1,00
foglamp	foglight	0,00	0,00	0,00	0,00	1,00	0,00
hand brake	parking brake	1,00	1,00	1,00	1,00	1,00	1,00
bumper	push rod	0,00	0,00	0,00	0,00	0,00	0,00
driving mirror	rearview mirror	0,00	0,00	0,00	0,00	1,00	0,00
seat belt	belt	0,69	0,89	0,00	0,00	0,38	1,00
safety belt	life belt	1,00	1,00	0,00	0,00	1,00	0,35
klaxon	horn	0,81	0,94	0,00	0,00	0,13	1,00
claxon	klaxon	1,00	1,00	0,00	0,00	1,00	0,00
claxon	car horn	0,61	0,82	0,00	0,00	1,00	0,00
gearshift	gear lever	1,00	1,00	0,00	0,00	1,00	0,00
windshield	windscreen	1,00	1,00	1,00	1,00	1,00	1,00
headlamp	headlight	1,00	1,00	1,00	1,00	1,00	1,00
taillight	taillamp	0,00	0,00	0,00	0,00	1,00	0,00
taillight	rear light	1,00	1,00	0,00	0,00	1,00	0,00
taillight	rear lamp	1,00	1,00	0,00	0,00	1,00	0,00
taillamp	rear light	0,00	0,00	0,00	0,00	1,00	0,00
taillamp	rear lamp	0,00	0,00	0,00	0,00	1,00	0,00

Palavra 1	**Palavra 2**	**LCH**	**WUP**	**JCN**	**LIN**	**HSO**	**RES**
rear light	rear lamp	1,00	1,00	0,00	0,00	1,00	0,00

Table C.2: Normalised Values among LCH, WUP, JCN, HSO and RES WordNet-based Semantic Similarity Algorithms

Appendix D

Memorised Concepts for the Basic Learning Mechanism

This Appendix presents the memorised concepts after the negotiation between enterprise agents, representing customers (CEAg) and suppliers (SEAg). Table D.1 has the requested concept and the respective correspondent concept together with its confidence.

CEAg	**SEAg**	**Requested Concept**	**Correspondent Concept**	**Confidence**
$Agent_1$	$Agent_2$	Suspension	Suspension_system	High
		Window	Car_Window	Moderate
		Throttle	Accelerator	High
		Clutch_pedal	Clutch	High
		Antenna	Transmitting_Aerial	High
		Hand_Brake	Parking_Brake	High
		Mirror	Rearview_Mirror	Weak
		Seat_Belt	Belt	Weak
		Gearshift	Gear_Lever	High
		Windshield	Windscreen	High
		Headlamp	Headlight	High
		Taillight	Rear_Lamp	High
		Spark_Plug	Sparking_Plug	High
		Battery	Electric_Battery	High
		Speedometer	Speed_Indicator	High
		Motor	Engine	High

CEAg	**SEAg**	**Requested Concept**	**Correspondent Concept**	**Confidence**
$Agent_1$	$Agent_3$	Suspension	Suspension	High
		Window	Window	High
		Throttle	Accelerator	High
		Clutch_pedal	Car_pedal	Moderate
		Antenna	Aerial	High
		Hand_Brake	Emergency_brake	Weak
		Mirror	Car_mirror	Moderate
		Seat_Belt	Safety_belt	Weak
		Gearshift	Gear_change	Moderate
		Windshield	Windshield	High
		Headlamp	Headlight	High
		Taillight	Tail_lamp	High
		Spark_Plug	Electrical_spark	Moderate
		Battery	Auto_battery	High
		Speedometer	Speedo	Weak
		Motor	Motor	High
$Agent_1$	$Agent_4$	Suspension	Suspension_system	Moderate
		Window	Auto_window	Moderate
		Throttle	Accelerator	Moderate
		Clutch_pedal	Clutch	Moderate
		Antenna	Aerial	High
		Hand_Brake	Parking_brake	Weak
		Mirror	Mirror	Weak
		Seat_Belt	Safety_belt	Moderate
		Gearshift	Gear	Moderate
		Windshield	Windshield	Moderate
		Headlamp	Headlight	Weak
		Taillight	Rear_light	Weak
		Spark_Plug	Spark_Plug	Weak
		Battery	Battery	High
		Speedometer	Speed_controller	Moderate
		Motor	Engine	Moderate

CEAg	**SEAg**	**Requested Concept**	**Correspondent Concept**	**Confidence**
$Agent_2$	$Agent_3$	Claxon	Klaxon	High
		Bumper	Bumper	Moderate
		Accelerator	Accelerator	High
		Clutch	Car_pedal	Moderate
		Transmitting_Aerial	aerial	Moderate
		Parking_Brake	Emergency_brake	Moderate
		Rearview_Mirror	Car_mirror	Weak
		Belt	Life_belt	High
		Gear_Lever	Gear_change	Moderate
		Windscreen	Windshield	Weak
		Headlight	Headlight	High
		Rear_Lamp	Rear_light	Weak
		Sparking_Plug	Electrical_spark	Weak
		Stoplight	Traffic light	High
		Wheel	Wheel	Weak
		tire	Tyre	High
$Agent_2$	$Agent_4$	Claxon	Horn	Weak
		Bumper	Bumper_guard	Moderate
		Accelerator	Throttle	Moderate
		Clutch	Clutch_pedal	Moderate
		Transmitting_Aerial	Aerial	High
		Parking_Brake	Parking_brake	Moderate
		Rearview_Mirror	Mirror	Weak
		Belt	Safety_belt	Weak
		Gear_Lever	Gear	Weak
		Windscreen	Windshield	Weak
		Headlight	Headlight	Weak
		Rear_Lamp	Tail_light	High
		Sparking_Plug	Spark_Plug	Moderate
		Stoplight	Stop_light	Moderate
		Wheel	Wheel	Moderate
		Tire	Tire	High

CEAg	**SEAg**	**Requested Concept**	**Correspondent Concept**	**Confidence**
$Agent_3$	$Agent_4$	Parking_light	Side light	High
		Klaxon	Horn	Moderate
		Bumper	Bumper_guard	Moderate
		Accelerator	Throttle	Weak
		Car_pedal	Clutch_pedal	Weak
		Aerial	Aerial	High
		Emergency_brake	Parking_brake	Weak
		Car_mirror	Mirror	Moderate
		Life_belt	Safety_belt	Moderate
		Gear_change	Gear	Moderate
		Windshield	Windshield	Weak
		Headlight	Headlight	Weak
		Rear_light	Tail_light	Weak
		Electrical_spark	Spark_Plug	Weak
		Wheel	Wheel	Moderate
		Tyre	Tire	High
$Agent_5$	$Agent_6$	Headlight	Headlamp	High
		Fog_lamp	Fog_light	High
		Parking_lamp	Parking_light	High
		Brake_lamp	Brake_light	High
		Tail_lamp	Tail_light	High
$Agent_5$	$Agent_1$	Headlight	Headlamp	High
		Tail_lamp	Taillight	High
$Agent_5$	$Agent_2$	Headlight	Headlight	Moderate
		Tail_lamp	Rear_lamp	Moderate
$Agent_5$	$Agent_3$	Headlight	Headlight	High
		Parking_lamp	Parking_light	Moderate
		Tail_lamp	Rear_light	High
$Agent_5$	$Agent_4$	Headlight	Headlight	Moderate
		Parking_lamp	Side light	Moderate
		Tail_lamp	Tail_light	Weak
$Agent_6$	$Agent_1$	Auto_Motor	Motor	High
$Agent_6$	$Agent_2$	Auto_Motor	Engine	Moderate
		Auto_Wheel	Wheel	High
		Auto_Tire	Tire	High
$Agent_6$	$Agent_3$	Auto_Motor	Motor	Moderate
		Auto_Wheel	Wheel	Moderate
		Auto_Tire	Tyre	High
$Agent_6$	$Agent_4$	Auto_Motor	Engine	Weak
		Auto_Wheel	Wheel	Weak
		Auto_Tire	Tire	Moderate

CEAg	**SEAg**	**Requested Concept**	**Correspondent Concept**	**Confidence**
$Agent_7$	$Agent_1$	Handwheel	Steering_wheel	High
$Agent_7$	$Agent_8$	Handwhell	Steering_wheel	Moderate

Table D.1: Memorised Concepts Using the BLM

Appendix E

OWL Ontology

This appendix presents an ontology represented in OWL format in the automobile assembling domain.

```
<rdf:RDF xml:base="http://www.owlontologies.com/unnamed.owl">
<owl:Ontology rdf:about="">
<owl:imports rdf:resource="http://protege.stanford.edu/plugins/owl/protege"/>
</owl:Ontology>
<owl:Class rdf:ID="Electric_Battery">
<rdfs:subClassOf>
<owl:Class rdf:ID="Item"/>
</rdfs:subClassOf>
<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
An electrical storage container designed to produce DC voltage by means of an
electrochemical reaction.
</rdfs:comment>
</owl:Class>
<owl:Class rdf:ID="Gear_Lever">
<rdfs:subClassOf rdf:resource="#Item"/>
<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
lever that allows you select the different gears of a transmission
</rdfs:comment>
</owl:Class>
<owl:Class rdf:ID="Car_Window">
<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">a window in a
car</rdfs:comment>
<rdfs:subClassOf rdf:resource="#Item"/>
```

```
</owl:Class>
<owl:Class rdf:ID="Brake">
<protege:abstract>true</protege:abstract>
<rdfs:subClassOf rdf:resource="#Item"/>
</owl:Class>
<owl:Class rdf:ID="Wheel">
<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
a wheel that has a tire and rim and hubcap; used to propel the car
</rdfs:comment>
<rdfs:subClassOf rdf:resource="#Item"/>
</owl:Class>
<owl:Class rdf:ID="Low_Beam_Light">
<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">dim light, dimmed
headlight</rdfs:comment>
<rdfs:subClassOf rdf:resource="#Item"/>
</owl:Class>
<owl:Class rdf:ID="VBelt">
<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
rubber belt that drives such things as the alternator, air conditioning compressor, power
steering pump and waterpump
</rdfs:comment>
<rdfs:subClassOf rdf:resource="#Item"/>
</owl:Class>
<owl:Class rdf:ID="Lightning_System">
<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
all components that make up the illumination of a car
</rdfs:comment>
<rdfs:subClassOf rdf:resource="#Item"/>
<protege:abstract>true</protege:abstract>
</owl:Class>
<owl:Class rdf:ID="Sparking_Plug">
<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
Part of the ignition system, it's an electrical device with a ground and center electrode
where a spark is created between the two by a high voltage current from the distributor.
</rdfs:comment>
<rdfs:subClassOf rdf:resource="#Item"/>
</owl:Class>
<owl:Class rdf:ID="Belt">
<rdfs:subClassOf rdf:resource="#Item"/>
<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">band to tie around
the body for safety</rdfs:comment>
</owl:Class>
<owl:Class rdf:ID="Suspension_System">
```

```
<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
The suspension connects the vehicle body to the frame, helps the vehicle to handle better,
while increasing comfort and isolating passengers from bumps and vibration.
</rdfs:comment>
<rdfs:subClassOf rdf:resource="#Item"/>
</owl:Class>
<owl:Class rdf:ID="Airbag">
<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
A supplemental safety system in vehicles that inflates to cushion an occupant during a
collision.
</rdfs:comment>
<rdfs:subClassOf rdf:resource="#Item"/>
</owl:Class>
<owl:Class rdf:ID="Parking_Light">
<rdfs:subClassOf rdf:resource="#Item"/>
<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">faint light on a
parked car</rdfs:comment>
</owl:Class>
<owl:Class rdf:ID="Car_Horn">
<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
a device on an automobile for making a warning noise
</rdfs:comment>
<rdfs:subClassOf rdf:resource="#Item"/>
</owl:Class>
<owl:Class rdf:ID="Transmitting_Aerial">
<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
an electrical device that sends or receives radio or television signals
</rdfs:comment>
<rdfs:subClassOf rdf:resource="#Item"/>
</owl:Class>
<owl:Class rdf:ID="Stoplight">
<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
a red light on the rear of a motor vehicle that flashes up during driver brakes
</rdfs:comment>
<rdfs:subClassOf rdf:resource="#Item"/>
</owl:Class>
<owl:Class rdf:ID="Speed_Indicator">
<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
a meter fixed to a vehicle that measures and displays its speed
</rdfs:comment>
<rdfs:subClassOf rdf:resource="#Item"/>
</owl:Class>
```

```
<owl:Class rdf:ID="Door">
<rdfs:subClassOf rdf:resource="#Item"/>
<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
a swinging or sliding barrier that will close the entrance to a room or building or
vehicle
</rdfs:comment>
</owl:Class>
<owl:Class rdf:ID="Oil_Sump">
<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">oil
reservoir</rdfs:comment>
<rdfs:subClassOf rdf:resource="#Item"/>
</owl:Class>
<owl:Class rdf:ID="Clutch">
<rdfs:subClassOf rdf:resource="#Item"/>
<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
A mechanical device which uses mechanical, magnetic, or friction type connections to
facilitate engaging or disengaging two rotating members.
</rdfs:comment>
</owl:Class>
<owl:Class rdf:ID="Foglamp">
<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
headlight that provides strong beam for use in foggy weather
</rdfs:comment>
<rdfs:subClassOf rdf:resource="#Item"/>
</owl:Class>
<owl:Class rdf:ID="Rearview_Mirror">
<rdfs:subClassOf rdf:resource="#Item"/>
<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
car mirror that reflects the view out of the rear window
</rdfs:comment>
</owl:Class>
<owl:Class rdf:ID="Parking_Brake">
<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
The parking brake is used when parking on an incline to prevent the car from rolling away.
The parking brake is usually cable operated and can be used as a backup if the regular
hydraulic brake system fails.
</rdfs:comment>
<rdfs:subClassOf rdf:resource="#Brake"/>
</owl:Class>
<owl:Class rdf:ID="Cylinder_Block">
<rdfs:subClassOf rdf:resource="#Item"/>
<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
a metal casting containing the cylinders and cooling ducts of an engine
```

```
</rdfs:comment>

</owl:Class>

<owl:Class rdf:ID="Rear_Lamp">

<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">usually red light at
the rear of a car</rdfs:comment>

<rdfs:subClassOf rdf:resource="#Item"/>

</owl:Class>

<owl:Class rdf:ID="Engine_Block">

<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">

a metal casting containing the cylinders and cooling ducts of an engine

</rdfs:comment>

<rdfs:subClassOf rdf:resource="#Item"/>

</owl:Class>

<owl:Class rdf:ID="Car_Body">

<rdfs:subClassOf rdf:resource="#Item"/>

<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">

The assemblage of components, including windows, doors, seats, etc., that provide
enclosures for passengers and / or cargo in a motor vehicle.

</rdfs:comment>

<protege:abstract>true</protege:abstract>

</owl:Class>

<owl:Class rdf:ID="Engine">

<rdfs:subClassOf rdf:resource="#Item"/>

<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">

motor that converts thermal energy to mechanical work

</rdfs:comment>

</owl:Class>

<owl:Class rdf:ID="Accelerator">

<rdfs:subClassOf rdf:resource="#Item"/>

<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">

A foot operated device which controls the flow of fuel or air to the engine, controlling
engine rpm.

</rdfs:comment>

</owl:Class>

<owl:Class rdf:ID="Windscreen">

<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">

window in front of a car to protect occupants of a vehicle

</rdfs:comment>

<rdfs:subClassOf rdf:resource="#Item"/>

</owl:Class>

<owl:Class rdf:ID="Push_Rod">

<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
```

```
a mechanical device consisting of bars at either end of a vehicle to absorb shock and
prevent serious damage
</rdfs:comment>
<rdfs:subClassOf rdf:resource="#Item"/>
</owl:Class>
<owl:Class rdf:ID="Long_Distance_Light">
<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">a strong, bright
light</rdfs:comment>
<rdfs:subClassOf rdf:resource="#Item"/>
</owl:Class>
<owl:Class rdf:ID="Headlight">
<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
a powerful light with reflector; attached to the front of an automobile or locomotive
</rdfs:comment>
<rdfs:subClassOf rdf:resource="#Item"/>
</owl:Class>
<owl:ObjectProperty rdf:ID="has_door">
<rdfs:domain rdf:resource="#Car_Body"/>
<rdfs:range rdf:resource="#Door"/>
<rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="has_car_window">
<rdfs:domain rdf:resource="#Car_Body"/>
<rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
<rdfs:range rdf:resource="#Car_Window"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="has_low_beam_light">
<rdfs:range rdf:resource="#Low_Beam_Light"/>
<rdfs:domain rdf:resource="#Lightning_System"/>
<rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="has_stoplight">
<rdfs:domain rdf:resource="#Lightning_System"/>
<rdfs:range rdf:resource="#Stoplight"/>
<rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="has_oil_sump">
<rdfs:range rdf:resource="#Oil_Sump"/>
<rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
<rdfs:domain rdf:resource="#Engine"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="has_push_rod">
```

```
<rdfs:domain rdf:resource="#Car_Body"/>
<rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
<rdfs:range rdf:resource="#Push_Rod"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="has_foglamp">
<rdfs:domain rdf:resource="#Lightning_System"/>
<rdfs:range rdf:resource="#Foglamp"/>
<rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="has_cylinder_block">
<rdfs:domain rdf:resource="#Engine"/>
<rdfs:range rdf:resource="#Cylinder_Block"/>
<rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="has_headlight">
<rdfs:domain rdf:resource="#Lightning_System"/>
<rdfs:range rdf:resource="#Headlight"/>
<rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
</owl:ObjectProperty>
<owl:DatatypeProperty rdf:ID="price_currency">
<rdfs:domain rdf:resource="#Item"/>
<rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
<rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="push_rod_position">
<rdfs:range>
<owl:DataRange>
<owl:oneOf rdf:parseType="Resource">
<rdf:rest rdf:parseType="Resource">
<rdf:rest rdf:resource="http://www.w3.org/1999/02/22rdfsyntaxns#nil"/>
<rdf:first rdf:datatype="http://www.w3.org/2001/XMLSchema#string">rear</rdf:first>
</rdf:rest>
<rdf:first rdf:datatype="http://www.w3.org/2001/XMLSchema#string">front</rdf:first>
</owl:oneOf>
</owl:DataRange>
</rdfs:range>
<rdfs:domain rdf:resource="#Push_Rod"/>
<rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="diameter">
<rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#float"/>
```

```
<rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
<rdfs:domain rdf:resource="#Wheel"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="horsepower">
<rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
<rdfs:domain rdf:resource="#Engine"/>
<rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="transmitting_aerial_type">
<rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
<rdfs:domain rdf:resource="#Transmitting_Aerial"/>
<rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="wheel_rim_material">
<rdfs:domain rdf:resource="#Wheel"/>
<rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
<rdfs:range>
<owl:DataRange>
<owl:oneOf rdf:parseType="Resource">
<rdf:rest rdf:parseType="Resource">
<rdf:rest rdf:resource="http://www.w3.org/1999/02/22rdfsyntaxns#nil"/>
<rdf:first rdf:datatype="http://www.w3.org/2001/XMLSchema#string">aluminium</rdf:first>
</rdf:rest>
<rdf:first rdf:datatype="http://www.w3.org/2001/XMLSchema#string">steel</rdf:first>
</owl:oneOf>
</owl:DataRange>
</rdfs:range>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="manufacturer">
<rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
<rdfs:domain rdf:resource="#Wheel"/>
<rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="frequency">
<rdfs:domain rdf:resource="#Transmitting_Aerial"/>
<rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
<rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#float"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="torque">
<rdfs:domain rdf:resource="#Engine"/>
<rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
```

```
<rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="capacity_of_cylinders">
<rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
<rdfs:domain rdf:resource="#Engine"/>
<rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#float"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="tension">
<rdfs:domain>
<owl:Class>
<owl:unionOf rdf:parseType="Collection">
<owl:Class rdf:about="#Long_Distance_Light"/>
<owl:Class rdf:about="#Low_Beam_Light"/>
<owl:Class rdf:about="#Parking_Light"/>
<owl:Class rdf:about="#Headlight"/>
<owl:Class rdf:about="#Stoplight"/>
<owl:Class rdf:about="#Electric_Battery"/>
</owl:unionOf>
</owl:Class>
</rdfs:domain>
<rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
<rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="tire_profile_depth">
<rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#float"/>
<rdfs:domain rdf:resource="#Wheel"/>
<rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
</owl:DatatypeProperty>
<owl:FunctionalProperty rdf:ID="number_of_cylinders">
<rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
<rdfs:domain rdf:resource="#Cylinder_Block"/>
<rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="tire_type">
<rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
<rdfs:domain rdf:resource="#Wheel"/>
<rdfs:range>
<owl:DataRange>
<owl:oneOf rdf:parseType="Resource">
<rdf:first rdf:datatype="http://www.w3.org/2001/XMLSchema#string">summer</rdf:first>
<rdf:rest rdf:parseType="Resource">
```

```
<rdf:first rdf:datatype="http://www.w3.org/2001/XMLSchema#string">winter</rdf:first>
<rdf:rest rdf:resource="http://www.w3.org/1999/02/22rdfsyntaxns#nil"/>
</rdf:rest>
</owl:oneOf>
</owl:DataRange>
</rdfs:range>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="fuel_ingestion">
<rdfs:domain rdf:resource="#Engine"/>
<rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
<rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#float"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="fuel">
<rdfs:domain rdf:resource="#Engine"/>
<rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
<rdfs:range>
<owl:DataRange>
<owl:oneOf rdf:parseType="Resource">
<rdf:rest rdf:parseType="Resource">
<rdf:first
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">gasoline_unleaded</rdf:first>
<rdf:rest rdf:parseType="Resource">
<rdf:rest rdf:resource="http://www.w3.org/1999/02/22rdfsyntaxns#nil"/>
<rdf:first
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">super_unleaded</rdf:first>
</rdf:rest>
</rdf:rest>
<rdf:first rdf:datatype="http://www.w3.org/2001/XMLSchema#string">diesel</rdf:first>
</owl:oneOf>
</owl:DataRange>
</rdfs:range>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="has_sparking_plug">
<rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
<rdfs:domain rdf:resource="#Engine"/>
<rdfs:range rdf:resource="#Sparking_Plug"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="has_long_distance_light">
<rdfs:domain rdf:resource="#Lightning_System"/>
<rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
<rdfs:range rdf:resource="#Long_Distance_Light"/>
</owl:FunctionalProperty>
```

```
<owl:FunctionalProperty rdf:ID="has_engine_block">
<rdfs:range rdf:resource="#Engine_Block"/>
<rdfs:domain rdf:resource="#Engine"/>
<rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="has_vbelt">
<rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
<rdfs:domain rdf:resource="#Engine"/>
<rdfs:range rdf:resource="#VBelt"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="arrangement_of_cylinders">
<rdfs:domain rdf:resource="#Cylinder_Block"/>
<rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
<rdfs:range>
<owl:DataRange>
<owl:oneOf rdf:parseType="Resource">
<rdf:rest rdf:parseType="Resource">
<rdf:first rdf:datatype="http://www.w3.org/2001/XMLSchema#string">v</rdf:first>
<rdf:rest rdf:resource="http://www.w3.org/1999/02/22rdfsyntaxns#nil"/>
</rdf:rest>
<rdf:first rdf:datatype="http://www.w3.org/2001/XMLSchema#string">inline</rdf:first>
</owl:oneOf>
</owl:DataRange>
</rdfs:range>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="width">
<rdfs:domain rdf:resource="#Wheel"/>
<rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#float"/>
<rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="transmission_type">
<rdfs:domain rdf:resource="#Engine"/>
<rdfs:range>
<owl:DataRange>
<owl:oneOf rdf:parseType="Resource">
<rdf:first rdf:datatype="http://www.w3.org/2001/XMLSchema#string">manual</rdf:first>
<rdf:rest rdf:parseType="Resource">
<rdf:rest rdf:resource="http://www.w3.org/1999/02/22rdfsyntaxns#nil"/>
<rdf:first rdf:datatype="http://www.w3.org/2001/XMLSchema#string">automatic</rdf:first>
</rdf:rest>
</owl:oneOf>
```

```
</owl:DataRange>
</rdfs:range>
<rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="price_value">
<rdfs:domain rdf:resource="#Item"/>
<rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
<rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#float"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="has_parking_light">
<rdfs:domain rdf:resource="#Lightning_System"/>
<rdfs:range rdf:resource="#Parking_Light"/>
<rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="wavelength">
<rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
<rdfs:domain rdf:resource="#Transmitting_Aerial"/>
<rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="holder_nr">
<rdfs:domain>
<owl:Class>
<owl:unionOf rdf:parseType="Collection">
<owl:Class rdf:about="#Long_Distance_Light"/>
<owl:Class rdf:about="#Low_Beam_Light"/>
<owl:Class rdf:about="#Parking_Light"/>
<owl:Class rdf:about="#Headlight"/>
<owl:Class rdf:about="#Stoplight"/>
</owl:unionOf>
</owl:Class>
</rdfs:domain>
<rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
<rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
</owl:FunctionalProperty>
<Cylinder_Block rdf:ID="carAssembling2OWL_Instance_20">
<price_value rdf:datatype="http://www.w3.org/2001/XMLSchema#float">1200.0</price_value>
<price_currency
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">EUR</price_currency>
</Cylinder_Block>
<Engine_Block rdf:ID="carAssembling2OWL_Instance_21">
<price_value rdf:datatype="http://www.w3.org/2001/XMLSchema#float">1170.0</price_value>
```

```
<price_currency
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">EUR</price_currency>

</Engine_Block>

<Sparking_Plug rdf:ID="carAssembling2OWL_Instance_38">

<price_currency
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">EUR</price_currency>

<price_value rdf:datatype="http://www.w3.org/2001/XMLSchema#float">35.0</price_value>

</Sparking_Plug>

<Windscreen rdf:ID="carAssembling2OWL_Instance_36">

<price_value rdf:datatype="http://www.w3.org/2001/XMLSchema#float">200.0</price_value>

<price_currency
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">EUR</price_currency>

</Windscreen>

<Headlight rdf:ID="carAssembling2OWL_Instance_22">

<price_currency
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">EUR</price_currency>

<price_value rdf:datatype="http://www.w3.org/2001/XMLSchema#float">45.0</price_value>

</Headlight>

<Rear_Lamp rdf:ID="carAssembling2OWL_Instance_37">

<price_value rdf:datatype="http://www.w3.org/2001/XMLSchema#float">45.0</price_value>

<price_currency
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">EUR</price_currency>

</Rear_Lamp>

<Suspension_System rdf:ID="carAssembling2OWL_Instance_28">

<price_value rdf:datatype="http://www.w3.org/2001/XMLSchema#float">400.0</price_value>

<price_currency
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">EUR</price_currency>

</Suspension_System>

<Stoplight rdf:ID="carAssembling2OWL_Instance_11">

<price_value rdf:datatype="http://www.w3.org/2001/XMLSchema#float">9.9</price_value>

<price_currency
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">EUR</price_currency>

</Stoplight>

<Belt rdf:ID="carAssembling2OWL_Instance_33">

<price_currency
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">EUR</price_currency>

<price_value rdf:datatype="http://www.w3.org/2001/XMLSchema#float">35.0</price_value>

</Belt>

<Engine rdf:ID="carAssembling2OWL_Instance_16">

<price_currency
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">EUR</price_currency>

<horsepower rdf:datatype="http://www.w3.org/2001/XMLSchema#int">100</horsepower>

<price_value rdf:datatype="http://www.w3.org/2001/XMLSchema#float">1400.0</price_value>

<fuel rdf:datatype="http://www.w3.org/2001/XMLSchema#string">diesel</fuel>
```

```
</Engine>
<Push_Rod rdf:ID="carAssembling2OWL_Instance_23">
<price_value rdf:datatype="http://www.w3.org/2001/XMLSchema#float">230.0</price_value>
<price_currency
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">EUR</price_currency>
</Push_Rod>
<Transmitting_Aerial rdf:ID="carAssembling2OWL_Instance_31">
<price_value rdf:datatype="http://www.w3.org/2001/XMLSchema#float">45.0</price_value>
<price_currency
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">EUR</price_currency>
</Transmitting_Aerial>
<Long_Distance_Light rdf:ID="carAssembling2OWL_Instance_10">
<price_value rdf:datatype="http://www.w3.org/2001/XMLSchema#float">9.9</price_value>
<price_currency
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">EUR</price_currency>
</Long_Distance_Light>
<Engine rdf:ID="carAssembling2OWL_Instance_15">
<price_currency
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">EUR</price_currency>
<horsepower rdf:datatype="http://www.w3.org/2001/XMLSchema#int">73</horsepower>
<capacity_of_cylinders
rdf:datatype="http://www.w3.org/2001/XMLSchema#float">1.9</capacity_of_cylinders>
<price_value rdf:datatype="http://www.w3.org/2001/XMLSchema#float">1250.0</price_value>
<fuel_ingestion rdf:datatype="http://www.w3.org/2001/XMLSchema#float">7.8</fuel_ingestion>
</Engine>
<Wheel rdf:ID="carAssembling2OWL_Instance_13">
<price_currency
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">EUR</price_currency>
<width rdf:datatype="http://www.w3.org/2001/XMLSchema#float">51.0</width>
<tire_profile_depth
rdf:datatype="http://www.w3.org/2001/XMLSchema#float">5.9</tire_profile_depth>
<price_value rdf:datatype="http://www.w3.org/2001/XMLSchema#float">109.9</price_value>
<manufacturer
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Michelin</manufacturer>
</Wheel>
<Airbag rdf:ID="carAssembling2OWL_Instance_27">
<price_value rdf:datatype="http://www.w3.org/2001/XMLSchema#float">1200.0</price_value>
<price_currency
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">EUR</price_currency>
</Airbag>
<Wheel rdf:ID="carAssembling2OWL_Instance_12">
<price_value rdf:datatype="http://www.w3.org/2001/XMLSchema#float">79.9</price_value>
<manufacturer rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Kleber</manufacturer>
<price_currency
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">EUR</price_currency>
```

```
</Wheel>
<Gear_Lever rdf:ID="carAssembling2OWL_Instance_35">
<price_currency
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">EUR</price_currency>
<price_value rdf:datatype="http://www.w3.org/2001/XMLSchema#float">90.0</price_value>
</Gear_Lever>
<Engine rdf:ID="carAssembling2OWL_Instance_14">
<horsepower rdf:datatype="http://www.w3.org/2001/XMLSchema#int">225</horsepower>
<price_currency
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">EUR</price_currency>
<price_value rdf:datatype="http://www.w3.org/2001/XMLSchema#float">1300.0</price_value>
</Engine>
<Car_Horn rdf:ID="carAssembling2OWL_Instance_34">
<price_currency
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">EUR</price_currency>
<price_value rdf:datatype="http://www.w3.org/2001/XMLSchema#float">30.0</price_value>
</Car_Horn>
<VBelt rdf:ID="carAssembling2OWL_Instance_19">
<price_value rdf:datatype="http://www.w3.org/2001/XMLSchema#float">69.9</price_value>
<price_currency
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">EUR</price_currency>
</VBelt>
<Parking_Brake rdf:ID="carAssembling2OWL_Instance_25">
<price_value rdf:datatype="http://www.w3.org/2001/XMLSchema#float">209.9</price_value>
<price_currency
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">EUR</price_currency>
</Parking_Brake>
<Car_Window rdf:ID="carAssembling2OWL_Instance_26">
<price_currency
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">EUR</price_currency>
<price_value rdf:datatype="http://www.w3.org/2001/XMLSchema#float">270.0</price_value>
</Car_Window>
<Clutch rdf:ID="carAssembling2OWL_Instance_29">
<price_value rdf:datatype="http://www.w3.org/2001/XMLSchema#float">40.0</price_value>
<price_currency
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">EUR</price_currency>
</Clutch>
<Speed_Indicator rdf:ID="carAssembling2OWL_Instance_24">
<price_value rdf:datatype="http://www.w3.org/2001/XMLSchema#float">220.0</price_value>
<price_currency
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">EUR</price_currency>
</Speed_Indicator>
<Rearview_Mirror rdf:ID="carAssembling2OWL_Instance_32">
<price_value rdf:datatype="http://www.w3.org/2001/XMLSchema#float">50.0</price_value>
```

```
<price_currency
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">EUR</price_currency>
</Rearview_Mirror>
<Oil_Sump rdf:ID="carAssembling2OWL_Instance_18">
<price_value rdf:datatype="http://www.w3.org/2001/XMLSchema#float">120.0</price_value>
<price_currency
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">EUR</price_currency>
</Oil_Sump>
<Foglamp rdf:ID="carAssembling2OWL_Instance_0">
<price_value rdf:datatype="http://www.w3.org/2001/XMLSchema#float">45.0</price_value>
<price_currency
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">EUR</price_currency>
</Foglamp>
<Accelerator rdf:ID="carAssembling2OWL_Instance_30">
<price_currency
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">EUR</price_currency>
<price_value rdf:datatype="http://www.w3.org/2001/XMLSchema#float">40.0</price_value>
</Accelerator>
<Low_Beam_Light rdf:ID="carAssembling2OWL_Instance_9">
<price_value rdf:datatype="http://www.w3.org/2001/XMLSchema#float">9.9</price_value>
<price_currency
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">EUR</price_currency>
</Low_Beam_Light>
<Parking_Light rdf:ID="carAssembling2OWL_Instance_17">
<price_currency
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">EUR</price_currency>
<price_value rdf:datatype="http://www.w3.org/2001/XMLSchema#float">12.5</price_value>
</Parking_Light>
<Electric_Battery rdf:ID="carAssembling2OWL_Instance_39">
<price_currency
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">EUR</price_currency>
<price_value rdf:datatype="http://www.w3.org/2001/XMLSchema#float">110.0</price_value>
</Electric_Battery>
</rdf:RDF>
<!  Created with Protege (with OWL Plugin 1.2, Build 162) http://protege.stanford.edu>
```

Appendix F

Structure of OWL Documents

This Appendix presents the structure of OWL documents, taking into consideration the structures used in this thesis.

Namespaces

Since ontologies are distinct resources, they must have identifiers making it possible to uniquely identify the concepts. XML namespaces provide a method to avoid element name conflicts, using Uniform Resource Identifiers (URI) references.

In RDF, an URI identifies a namespace which belongs to a schema. The declarations are included in the `rdf:RDF` tag. The built-in vocabulary for OWL is defined in the OWL namespace `http:/www.w3.org200207owl#`. An example is:

```
<rdf:RDF>
xmlns:protege="http://protege.stanford.edu/plugins/owl/protege#"
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#"
xmlns="http://www.owl-ontologies.com/unnamed.owl#"
</rdf:RDF>
```

Besides the namespaces for RDF, RDF(S) and OWL, this example declares a namespace for Protégé and another one `http://www.owl-ontologies.com/unnamed.owl#` that is used for all user-defined entities. This namespace can be edited in Protégé during the ontology definition process.

Ontology Header

The document header contains generic information about the ontology. An example is:

```
<owl:Ontology rdf:about=" ">
<owl:imports rdf:resource="http://protege.stanford.edu/plugins/owl/protege"/>
<rdfs:comment>Example of Ontology Header</rdfs:comment>
</owl:Ontology>
```

The first tag `<owl:Ontology rdf:about=" ">` states that this block describes the current ontology.

`<owl:imports>` refers another OWL ontology. An URI specifies from where the ontology is imported. The meanings of the definitions of the referenced ontology are considered to be part of the importing ontology. Since the imported ontology in this example contains a class definition for `<owl:Class rdf:ID="PAL-Constraint"/>`, it can be used in the importing ontology, e.g., by instantiating this class via `<protege:PAL-Constraint rdf:ID="CarAssembling_00141"/>`. This concrete import is required due to the OWL plug-in used within Protégé.

`<owl:imports>` statements are transitive, which means that if ontology A imports ontology B, and ontology B imports ontology C, then ontology A imports both, ontology B and ontology C.

Importing an ontology into itself is regarded as a null action. If ontology A imports B and B imports A, they are considered to be equivalent.

Classes

By defining classes, resources with similar characteristics are combined. Every class is associated with a set of individuals, called the class extension. The individuals in the class extension are also called instances of the class.

The simplest way of describing a class is through a class name, like this short example shows:

```
<owl:Class rdf:ID="Automobile_Part">
```

This will assert the triple `"ex:Automobile_Part rdf:type owl:Class"`, where `ex` is the namespace of the relevant ontology.

The effective use of ontologies depends on the ability to reason about individuals. Hence, a mechanism is necessary to describe the classes the individuals belong to and the properties they inherit. Although it is possible to assert specific

properties about individuals, most of the power of ontologies is a result of class-based reasoning. Therefore, OWL provides class axioms that state additional characteristics of a class.

To express taxonomy, the basic construct is `rdfs:subClassOf`. It states that the class extension of one class description is a subset of the class extension of another class description. An example:

```
<owl:Class rdf:ID="Light">
<rdfs:subClassOf rdf:resource="#Automobile_Part"/>
<protege:abstract>true<protege:abstract>
<rdfs:comment rdf:datatype="http:/www.w3.org2001XMLSchema#string">any device serving as a source of car illumination
</rdfs:comment>
</owl:Class>
```

This class axiom declares a subclass relation between the two OWL classes. Subclass relations provide necessary conditions for belonging to a class. In this case, an individual belonging to Light is also to be an Automobile_Part.

Additionaly to user-defined taxonomic relations, `owl:Class` is implicitly a subclass of the predefined class `owl:Thing`. This means that the class extension of `owl:Thing` is the set of all individuals.

The `owl:unionOf` property links one class to a list of class descriptions. The statement describes an anonymous class whose class extension contains those individuals that occur in at least one of the class extensions of the class descriptions in the list. An example:

```
<owl:FunctionalProperty rdf:ID="holder_nr">
<rdfs:domain>
<owl:Class>
<owl:unionOf rdf:parseType="Collection">
<owl:Class rdf:about="#Stoplight"/>
<owl:Class rdf:about="#Low_Beam_Light"/>
<owl:Class rdf:about="#Long_Distance_Light"/>
<owl:Class rdf:about="#Parking_Light"/>
<owl:Class rdf:about="#Head_Light"/></owl:unionOf>
</owl:Class>
</rdfs:domain>
<rdfs:range rdf:resource="http:/www.w3.org2001XMLSchema#int"/>
<rdf:type rdf:resource="http:/www.w3.org200207owl#DatatypeProperty"/>
</owl:FunctionalProperty>
```

The `owl:FunctionalProperty` with `rdf:ID="holder_nr"` belongs to each class that is listed in the set `owl:unionOf`.

Individuals

Individuals describe members of classes. They are defined with individual axioms, which are also called facts. Facts are statements indicating class membership and property values of individuals. An example:

```
<Wheel rdf:ID="carAssembling2_00213">
<price_value rdf:datatype="http:/www.w3.org2001XMLSchema#float">89.9</price_value>
<price_currency rdf:datatype="http:/www.w3.org2001XMLSchema#string">EUR
</price_currency>
<manufacturer rdf:datatype="http:/www.w3.org2001XMLSchema#string">Michelin
</manufacturer>
</Wheel>
```

The individual named `carAssembling2_00213` is an instance of the class Wheel. This name was generated automatically by Protégé. The facts are `price_value`, `price_currency` and `manufacturer`. Each one is defined with an individual value.

Properties

Properties are binary relations that allow the assertion of general facts about the members of classes and specific facts about individuals. OWL distinguishes between two main categories: datatype properties and object properties.

Datatype properties describe relations between instances of classes and RDF literals and XML Schema datatypes. In the following example, "horsepower" describes the relation between the class "Motor" and the XML datatype int. The axiom `rdfs:range` asserts that the values of this property must belong to data values in the specified data range, while `rdfs:domain` determines the class. Additionally, the property can be commented using the appropriate RDFS tag.

```
<owl:DatatypeProperty rdf:ID="horsepower">
<rdfs:comment rdf:datatype="http:/www.w3.org2001XMLSchema#string">SAE@rpm
</rdfs:comment>
<rdfs:range rdf:resource="http:/www.w3.org2001XMLSchema#int"/>
<rdfs:domain rdf:resource="#Motor"/>
<rdf:type rdf:resource="http:/www.w3.org200207owl#FunctionalProperty"/>
</owl:DatatypeProperty>
```

The second category are object properties. They are relations between instances of two classes, i.e., they relate individuals to individuals. An example is:

```
<owl:ObjectProperty rdf:ID="has_oil_sump">
<rdfs:range rdf:resource="#Oil_Sump"/>
<rdf:type rdf:resource="http:/www.w3.org200207owl#FunctionalProperty">
<rdfs:domain rdf:resource="#Motor"/>
<owl:ObjectProperty>
```

The properties `rdfs:range` and `rdfs:domain` are built-in properties. In the case of an object property, the former syntactically relates a property to a class description, in this case Oil_Sump. The axiom asserts that the values of this property must belong to the class extension of the class description. The `rdfs:domain` syntactically relates a property to a class description. Expressed in natural language, we would say "a motor has an oil sump". An `rdfs:domain` axiom asserts that the subjects of such property statements must belong to the class extension of the indicated class description.

Another syntactic variation, semantically equivalent to the example above, is the following one. It uses the tag `owl:FunctionalProperty` instead of including this information in the rdf:type tag. The built-in class `owl:FunctionalProperty` is a special subclass of the RDF class `rdf:Property`.

```
<owl:ObjectProperty rdf:ID="has_oil_sump">
<rdfs:range rdf:resource="#Oil_Sump">
<rdfs:domain rdf:resource="#Motor">
</owl:ObjectProperty>
<owl:FunctionalProperty rdf:about="has_oil_sump">
```

A functional property is a property that can have only one value y for each instance x, this means that there can not be two distinct values y_1 and y_2 such that the pairs (x, y_1) and (x, y_2) are both instances of this property. Carried forward to this example, which states that the `has_oil_sump` property is functional, it means that "one oil sump can only belong to one motor". Both object properties and datatype properties can be declared as functional.

Datatypes

OWL allows three types of data range specifications. First of all, OWL uses the RDF datatyping scheme, which in turn is derived from the XML datatyping scheme. Secondly, it allows the use the RDFS class `rdfs:Literal`, which is the class of literal values such as strings and integers. Finally, the enumerated datatype can be used.

The enumerated datatype uses the `owl:oneOf` construct, which defines a range of data values. It is not only used in connection with properties, like in the following example, but also describe enumerated classes.

```
<owl:FunctionalProperty rdf:ID="fuel">
<rdfs:range>
<owl:DataRange>
<owl:oneOf rdf:parseType="Resource">
<rdf:first rdf:datatype="http:/www.w3.org2001XMLSchema#string">diesel</rdf:first>
<rdf:rest rdf:parseType="Resource">
<rdf:first rdf:datatype="http:/www.w3.org2001XMLSchema#string">gasoline_unleaded
</rdf:first>
<rdf:rest rdf:parseType="Resource">
<rdf:first rdf:datatype="http:/www.w3.org2001XMLSchema#string">super_unleaded
</rdf:first>
<rdf:rest rdf:resource="http:/www.w3.org19990222-rdf-syntax-ns#nil"/>
</rdf:rest>
</rdf:rest>
</owl:oneOf>
</owl:DataRange>
</rdfs:range>
<rdfs:domain rdf:resource="#Engine"/>
<rdf:type rdf:resource="http:/www.w3.org200207owl#DatatypeProperty"/>
</owl:FunctionalProperty>
```

In the case of an enumerated datatype, the subject of `owl:oneOf` is a blank node of class `owl:DataRange` and the object is a list of literals. RDF requires this collection to be a list of RDF node elements. In other words, the list of data values is a nested construction and has to be filled with the basic list constructs `rdf:first`, `rdf:rest` and `rdf:nil`, as the example clarifies.