



Gonçalo Franco Pita Louro Alves

Licenciado em Ciências de Engenharia
Electrotécnica e de Computadores

A Framework for Semantic Checking of Information Systems

Dissertação para obtenção do Grau de Mestre em
Engenharia Electrotécnica e de Computadores

Orientador: Ricardo Jardim-Gonçalves, Professor Auxiliar,
FCT-UNL

Co-orientador: João Filipe dos Santos Sarraipa, Investigador,
UNINOVA

Júri:

Presidente:	Doutor João Francisco Alves Martins
Arguente:	Doutor João Pedro Mendonça de Assunção da Silva
Vogais:	Doutor Ricardo Luís Rosa Jardim Gonçalves
	Mestre João Filipe dos Santos Sarraipa



FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Setembro de 2012

A Framework for Semantic Checking of Information Systems

Copyright © Gonçalo Franco Pita Louro Alves, FCT/UNL, UNL

A Faculdade de Ciências e Tecnologia e a Universidade Nova de Lisboa tem o direito, perpetuo e sem limites geográficos, de arquivar e publicar esta dissertação através de exemplares impressos reproduzidos em papel ou de forma digital, ou por qualquer outro meio conhecido ou que venha a ser inventado, e de a divulgar através de repositórios científicos e de admitir a sua cópia e distribuição com objectivos educacionais ou de investigação, não comerciais, desde que seja dado crédito ao autor e editor.

ACKNOWLEDGEMENTS

First of all, I would like to thank all the people who helped me during my academic course.

To my advisor, Professor Ricardo Gonçalves for giving me the opportunity to work with him and with his research group, and for giving me valuable advice during the execution of this work.

To all members of GRIS, and especially to João Sarraipa for being there every day, for his attention, guidance and support during the research and the preparation of this dissertation.

To my friends, Gonçalo Barros, Francisco Esteves, Nuno Vasconcelos, João Silva, João Filipe, Gonçalo Carvalho, João Melo, Pedro Almeida, Ricardo Lampreia and to everyone else whom I may have forgot to mention, for all your support and for providing many unforgettable moments during this academic experience.

To my parents, brother and all of my family for providing with everything you could and for trying to give me the best possible future. You mean a lot to me.

Finally, to my girlfriend Íris, for always being there for me and for supporting me, pushing me to go further and to never give up, but more importantly for always believing in me.

To all, you have my deepest and sincerest gratitude.

ABSTRACT

In this day and age, enterprises often find that their business benefits greatly if they collaborate with others in order to be more competitive and productive. However these collaborations often come with some costs since the worldwide diversity of communities has led to the development of various knowledge representation elements, namely ontologies that, in most cases, are not semantically equivalent. Consequently, even though some enterprises may operate in the same domain, they can have different representations of that same knowledge. However, even after solving this issue and establishing a semantic alignment with other systems, they do not remain unchanged. Subsequently, a regular check of its semantic alignment is needed.

To aid in the resolution of this semantic interoperability problem, the author proposes a framework that intends to provide generic solutions and a mean to validate the semantic consistency of ontologies in various scenarios, thus maintaining the interoperability state between the enrolled systems.

KEYWORDS

Semantic Interoperability, Ontology Validation, Consistency Checking;

Nos dias de hoje, as empresas muitas vezes verificam que o seu negócio beneficia bastante quando colaboram com outros, aumentando a sua competitividade e produtividade. Contudo estas colaborações tipicamente têm algum custo associado, pois a diversidade global de comunidades conduziu ao desenvolvimento de vários elementos de representação de conhecimento, nomeadamente ontologias, que não são semanticamente coincidentes. Consequentemente, e apesar de algumas empresas trabalharem sobre um mesmo domínio, estas podem ter diferentes representações de um mesmo conhecimento. Porém, mesmo após ultrapassar esta barreira e se estabelecer um alinhamento semântico com outros sistemas, estes não permanecem inalterados. Por conseguinte, é necessário verificar regularmente o alinhamento semântico dos sistemas.

Para ajudar a solucionar estes problema de interoperabilidade semântica, o autor propõe uma estrutura que tem a intenção de proporcionar soluções genéricas e meios para validar a consistência de ontologias a nível semântico numa variedade de cenários, de modo a manter o estado de interoperabilidade entre os sistemas envolvidos.

PALAVRAS-CHAVE

Interoperabilidade Semântica, Validação de Ontologias, Verificação de Consistência;

TABLE OF CONTENTS

1. Introduction	1
1.1. Background Observation	1
1.2. Motivation	2
1.3. Research Method	2
1.4. Research Questions and Problems	4
1.5. Hypothesis	4
1.6. Dissertation Outline	4
2. Ontology Based Solutions for Knowledge Representation	7
2.1. Ontology Operations & Learning	7
2.1.1. Ontology mapping/matching	7
2.1.2. Ontology alignment	8
2.1.3. Ontology merging	8
2.1.4. Ontology Learning	9
2.2. Ontology Management Tools	9
2.2.1. Protégé	10
2.2.2. Ontopia	10
2.2.3. TM4L	12
2.2.4. Ontology Management Tools Concluding Remarks	13
2.3. Ontology Visualization	13
2.3.1. Ontopia Vizigator	14
2.3.2. Jambalaya	14
2.3.3. OntoGraf	16
2.3.4. TM4L Viewer	16
2.3.5. DebateGraph	17
2.3.6. TheBrain	18
2.3.7. XMind	19
2.3.8. Ontology Visualization Tools Concluding Remarks	19
2.4. Ontology Reasoners	20
2.4.1. HermiT	20
2.4.2. Pellet	22
2.4.3. FaCT++	23
2.4.4. RacerPro	23
2.4.5. Ontology Reasoners Concluding Remarks	24
3. Semantic Checking Framework	27
3.1. Interoperability	27
3.2. MENTOR Methodology	28
3.2.1. Mediator Ontology	31
3.3. Consistency Checking	32
3.3.1. Interoperability Checking	33
3.3.2. Semantic Checking	34
3.3.3. Semantic Adaptability Using a Mapping Tuple	35
3.4. Semantic Checking Framework	37
3.5. Concluding Remarks	39
4. Application Scenarios	41
4.1. Mechanical Scenario	41

4.1.1.	Single Structural Semantic Checking.....	43
4.1.2.	Single Structural Semantic Checking Concluding Remarks.....	44
4.1.3.	Single Conceptual Checking at MENTOR Scenario	44
4.1.4.	Single Conceptual Semantic Checking Concluding Remarks.....	45
4.1.5.	Multiple Conceptual Semantic Checking.....	45
4.1.6.	Multiple Conceptual Semantic Checking Concluding Remarks	50
4.2.	ENSEMBLE Scenario.....	51
4.2.1.	Composite Ontologies Checking at ENSEMBLE Scenario	53
4.2.2.	Multiple Structural Semantic Checking at ENSEMBLE Scenario.....	54
4.2.3.	ENSEMBLE Scenario Concluding Remarks	56
5.	Proof-of-Concept Implementation	59
5.1.	Used Technologies.....	59
5.1.1.	Java	59
5.1.2.	MySQL.....	59
5.1.3.	Protégé / Protégé-OWL API.....	60
5.1.4.	Changes and Annotations API.....	60
5.2.	Architecture.....	60
5.2.1.	Synchronization Module.....	61
5.2.2.	ChAO Ontology.....	61
5.2.3.	Wiki DB.....	61
5.2.4.	FInES Wiki	62
5.2.5.	EISB Reference Ontology.....	65
5.3.	Synchronization execution flows	73
5.3.1.	EISB Ontology to FInES Wiki Synchronization Execution Flow	73
5.3.2.	FInES Wiki to EISB Ontology Synchronization Execution Flow	74
5.4.	Concluding Remarks	75
6.	Synchronization Tool Demonstration	77
6.1.	Ontology to Wiki Synchronization Demonstration	77
6.1.1.	New Scientific Area instance	78
6.1.2.	Remove Scientific Area class	81
6.2.	Wiki to Ontology Synchronization Demonstration	83
6.2.1.	New Publication	84
6.2.2.	Edit Scientific Area.....	87
6.3.	Synchronization Tool Demonstration Concluding Remarks	89
7.	Conclusions and Future Work	91
7.1.	Research Validation	91
7.2.	Future Work	92
8.	References	93
9.	Appendix	97
9.1.	Ontology to Wiki Synchronization – New Scientific Area instance code example	97
9.2.	Ontology to Wiki Synchronization – Scientific Area class removal code example	97
9.3.	Wiki to Ontology Synchronization – New Publication code example	98
9.4.	Wiki to Ontology Synchronization – Edit Scientific Area code example	99

LIST OF FIGURES

Figure 1.1 - Phases of the Classical Research Method [6].....	2
Figure 2.1 - Ontology mapping/matching.....	8
Figure 2.2 - Ontology alignment	8
Figure 2.3 - Ontology merging.....	9
Figure 2.4 - Snapshot of the Protégé GUI	10
Figure 2.5 - Ontopoly snapshot	11
Figure 2.6 – Omigator snapshots - (a) Omnigator Main Page with index of topic maps; (b) Browsing a topic map.....	11
Figure 2.7 - Snapshot of TM4L user interface	12
Figure 2.8 - Ontopia Vizigator snapshot	14
Figure 2.9 - Jambalaya snapshots (a) Sink Tree view; (b) Nested Graph view	15
Figure 2.10 - OntoGraf snapshot.....	16
Figure 2.11 - Snapshot of the TM4L Viewer	17
Figure 2.12 - Snapshot of the debateGraph visualization tool	18
Figure 2.13 - Snapshot of theBrain visualization tool	18
Figure 2.14 - Snapshot of the XMind visualization tool.....	19
Figure 2.15 - HermiT reasoner Protégé plugin output - inconsistent ontology.....	21
Figure 2.16 - HermiT reasoner example using the command line	21
Figure 2.17 - HermiT reasoner java application integration example	22
Figure 2.18 - Pellet reasoner Protégé plugin output - inconsistent ontology	23
Figure 2.19 - FaCT++ reasoner Protégé plugin output - inconsistent ontology	23
Figure 2.20 - RacerPro reasoner supported features [40]	24
Figure 3.1 - Enterprise Interoperability [46]	27
Figure 3.2 - MENTOR Methodology [48]	30
Figure 3.3 - MENTOR prototype [49].....	30
Figure 3.4 - Mediator Ontology Structure [51]	31
Figure 3.5 - Mapping design and execution flow in data exchange	32
Figure 3.6 - Conformance Testing Example [57].....	33
Figure 3.7 - Interoperability Testing Example [57].....	34
Figure 3.8 - (a) Single Semantic Checking; (b) Composite Semantic Checking; (c) Multiple Semantic Checking	34
Figure 3.9 - Knowledge Mapping Types [50].....	36
Figure 4.1 - MENTOR scenario overview	42
Figure 4.2 - Used Ontologies.....	43
Figure 4.3 - Pellet reasoner output	44
Figure 4.4 - Reasoning Example (Retailer Ontology)	45
Figure 4.5 - Reasoning Example (Retailer and Reference Ontologies)	47
Figure 4.6 - Reasoning Example (Manufacturer and Reference Ontologies)	47
Figure 4.7 - Multiple Conceptual Semantic Checking Example	50
Figure 4.8 - ENSEMBLE scenario overview	51
Figure 4.9 - EISB Reference Ontology	52
Figure 4.10 - (a) FInES wiki Main Page; (b) FInES wiki article example	53
Figure 4.11 - EISB Reference Ontology and FInES Wiki Structural Comparison	55
Figure 4.12 - Ontology/Wiki Synchronization (a) Using Web Services; (b) Using XML/RDF Files	55
Figure 5.1 - Synchronization tool architecture	60
Figure 5.2 - Example of changes recorded in the ChAO ontology.....	61
Figure 5.3 - Wiki DB example.....	62
Figure 5.4 - FInES Wiki Homepage: 1 - FInES Reserach Roadmap; 2 - FInES Task Forces; 3 – EISB	62

Figure 5.5 - FInES Wiki: EISB Scientific Areas and Glossary.....	63
Figure 5.6 - FInES Wiki: (a) EISB Glossary; (b) Scientific Area category page example	63
Figure 5.7 - FInES Wiki: Scientific Area page example	64
Figure 5.8 - FInES Wiki: Sub Scientific Area page example.....	64
Figure 5.9 - FInES Wiki: EI Ingredient page example.....	65
Figure 5.10 - FInES Wiki: Publications page example.....	65
Figure 5.11 - EISB Reference Ontology overview	72
Figure 5.12 - Ontology to Wiki Synchronization execution flow	74
Figure 5.13 - Wiki to Ontology synchronization execution flow.....	75
Figure 6.1 - Synchronization tool GUI	77
Figure 6.2 - Ontology to Wiki Synchronization - New Scientific Area instance detection.....	79
Figure 6.3 - Ontology to Wiki Synchronization - Scientific Area instance	79
Figure 6.4 - Ontology to Wiki Synchronization - New Scientific Area instance finished synchronization	80
Figure 6.5 - Ontology to Wiki new Scientific Area synchronization example	80
Figure 6.6 - Ontology to Wiki Synchronization - Deleted Class detection.....	81
Figure 6.7 – EISB Reference Ontology (a) Before class deletion; (b) After class deletion	81
Figure 6.8 - Ontology to Wiki Synchronization - Wiki page deletion (Java GUI).....	82
Figure 6.9 - Ontology to Wiki Synchronizaton. (a) Wiki page before deletion; (b) Wiki page after deletion.....	83
Figure 6.10 - Wiki to Ontology Synchronization example - New publication detection	84
Figure 6.11 - Wiki to Ontology synchronization example - Publication to be synchronized.....	85
Figure 6.12 – Finished wiki to ontology synchronization process: (a) - java GUI; (b) Created instance	86
Figure 6.13 - Wiki to Ontology new publication synchronization example	86
Figure 6.14 - Wiki to Ontology Synchronization example - Edited Scientific area detection.....	87
Figure 6.15 - Scientific area page - (a) Before editing; (b) After editing.....	88
Figure 6.16 - Edited Scientific Area Synchronization - (a) Instance before editing; (b) Instance after editing; (c) Finished process - Java GUI.....	88
Figure 6.17 - Wiki to Ontology Edited Scientific Area example.....	89

LIST OF TABLES

Table 2.1 - Comparison between Ontology management tools	13
Table 2.2 - Comparison between ontology visualization tools	19
Table 2.3 - Ontology Reasoners Comparison	24
Table 3.1 - Semantic Mismatches [51]	36
Table 3.2 - Semantic Checking Framework.....	38
Table 4.1 – Framework applicability scenarios	41
Table 4.2 - Retailer Ontology Terms and Definitions	42
Table 4.3 - Manufacturer Ontology Terms and Definitions	42
Table 4.4 - Reference Ontology Terms and Definitions	43
Table 4.5 – Retailer Reference Mappings	46
Table 4.6 - Manufacturer - Reference Mappings	46
Table 4.7 - Reference - Manufacturer Conceptual Mappings	48
Table 4.8 - SWRL rules defined in the retailer - reference example	48
Table 4.9 - SWRL rules defined in the manufacturer - reference example	49
Table 4.10 - Identification of conceptual losses in information	50
Table 6.1 - Ontology to Wiki synchronization cases analysis	78
Table 6.2 - Wiki to Ontology Synchronization cases analysis.....	83

API - Application Programming Interface

ChAO – Changes and Annotations Ontology

CTM – Compact Topic Maps

DB – Database

EI – Enterprise Interoperability

EISB - Enterprise Interoperability Science Base

ENSEMBLE - Envisioning, Supporting and Promoting Future Internet Enterprise Systems Research through Scientific Collaboration

FinES – Future Internet Enterprise Systems

GUI – Graphical User Interface

HTML – Hypertext Markup Language

JDBC – Java Database Connectivity Driver

JVM - Java Virtual Machine

KB – Knowledge Base

KRE – Knowledge Representation Element

LTM – Linear Topic Maps

MENTOR - Methodology for Enterprise Reference Ontology Development

MO - Mediator Ontology

OWL – Web Ontology Language

RDF – Resource Description Framework

SHRIMP – Simple Hierarchical Multi-Perspective

SQL - Structured Query Language

SWRL - Semantic Web Rule Language

TM4L - Topic Maps 4 E-Learning

XML – Extensible Markup Language

XTM – XML Topic Maps

1. INTRODUCTION

Nowadays, in an increasingly global business environment, several companies have found that to make themselves more competitive and productive they have to collaborate with other enterprises, to compete with the larger organizations [1]. However the globalization that led to the collaboration between companies, also led to the development of various Knowledge Representation Elements (KREs), such as ontologies, which are not semantically coincident [2]. As a result enterprises are engaging in some standstills regarding the lack of interoperability of systems and software applications to manage and increase their collaborative business.

Since various companies that operate in the same domain may have different representations of a same Knowledge Base (KB), when they describe it electronically it will most likely lead to different representation models [1]. Thus interoperability problems, particularly regarding the semantics of the concepts involved, may surface when these different systems try to exchange or share information with one another.

Even after having established seamless communication and semantic alignment between systems it was identified the necessity of having “something” that allows companies to track their semantic evolution to keep the consistency and validity of their KREs. Since this is a vast and complex subject, it was recognized that a structured solution that encompasses several different scenarios was a possible step forward in help solving some of the semantic interoperability problems. Therefore the idea of a framework was conceived. A framework is a structure for supporting or enclosing something else, especially a skeletal support used as the basis for something being constructed [3].

To this effect, an interoperability framework that provides a set of assumptions, concepts, values and practices (methods & tools) [4] and that contemplates several scenarios for the semantic checking is a possible solution to the semantic interoperability maintenance issue.

1.1. Background Observation

Since interoperability between enterprises is becoming increasingly important to assure competitiveness and productivity, there is a need to constantly verify if the involved systems remain interoperable, particularly on a semantic level. For this reason, there is a need to have validation elements to ensure this interoperable state.

Due to the use of ontologies in enterprises to represent knowledge and consequently its semantics, it is needed to analyse its integration with other KREs. Thus a path to follow is to analyse the various KREs with a high relevance to ontologies.

1.2. Motivation

Although some work has been done in the Enterprise Interoperability (EI) field, these focus more on the seamless interoperability between enterprises rather than verifying the consistency of the exchanged information. In fact a research roadmap (Enterprise Interoperability Research Roadmap) has been defined with the main objective of identifying the main areas of research within the EI domain [5]. As a consequence, one of the great motivations for this dissertation work is the fact that the semantic interoperability between businesses and enterprises is an authentic research challenge and it is a research area that is in constant contact with the industrial world.

Furthermore, enterprises would benefit greatly if it is assured that the information they exchange, besides being received, is also well perceived by others, since communication would be made with much less effort.

Therefore this works aim is to provide a possible solution in the field of semantic interoperability, with focus on the verification of the semantic consistency of information, by proposing a framework to serve as a backbone in solving these issues.

1.3. Research Method

The research method adopted in this work is centred on the classical method, which is composed by seven steps, conveniently ordered from a more theoretical to a more practical view of the system, in addition to an eighth step which is the passage from the theoretical work to the industrial world. This research method starts by defining the research theme and area and leads to the testing step and results analysis. Since this method is iterative, the researcher can go back to the first steps if the obtained results weren't the expected ones to try a new approach. Figure 1.1 represents the different steps of this method that are described afterwards.



Figure 1.1 - Phases of the Classical Research Method [6]

A brief description of the steps, according to [6], follows.

1. **Research Question / Problem:** This is the most important step in research. It is a period of

study that intends to define the area of interest of the research. The research question must be clearly defined, making the study feasible and capable of being validated or refuted. Furthermore, a research question can be complemented with several minor questions to refine the main idea of the research subject. This is presented on section 1.4 - Research Questions and Problems.

2. Background / Observation: This step contemplates the study of the work done before about the same research area. In other words, this is where the state of the art research takes place. This is accomplished by reviewing literature and scientific projects bringing up the ideas of what was already tested and accomplished. Furthermore it is important to have a big variety of documents for searching information on the area of interest, since some of the literature although very reliable, can be outdated and on the other hand, some documentation can be recent and have very innovative ideas but low reliability. Finally, it is also in this step that the researcher defines what differs from the previous work to the one being developed, as well as the methodologies taken when approaching the solution.

The background observation (state of the art study) is comprehended in sections 2 and 3 of this dissertation.

3. Formulate Hypothesis: As its name indicates, in this step the researcher formulates the hypotheses in order to make the research problem simpler to understand, stating the ambitions to accomplish at the end of the project. The hypothesis can be seen as an educated guess since it states the predicted relationship amongst variables.

The hypotheses for this research work are presented in section 1.5 of this document.

4. Design Experiment: This step works as a preparation for the experimental step, where a prototype or system architecture is designed. In addition, it is significant to find a validation plan for the previous step, i.e. the hypothesis.

5. Test Hypothesis: This step comprehends the implementation of the designed prototype and the evaluation of the obtained results. A large amount of tests (especially in different scenarios) should be done in order to test effusively the outcomes given by the system. These outcomes are supposed to be collected for later analysis.

6. Interpret / Analyse Results: After the batteries of tests have been made to the system it is the time to evaluate and analyse the achieved results. It is at this point that the veracity and confidence in the hypothesis are put to the test. A number of outcomes are possible, the results can be satisfactory, proving the author right, or they can completely miss the initial idea. If the results point straight to the hypothesis, then it is reasonable to say that a good prevision was made and it is

possible to consider what comes after, making some recommendations for further research. But even if the results are not what was expected it should not be taken as a failure, but as an opportunity to improve the original approach and go back again to the first steps of this research method. The researcher can then try a different approach from the one taken before.

7. Publish Findings: The final results, if consistent, must end up in a valuable contribution to the scientific community as scientific papers. These papers can be then presented in conferences, where the author has the chance to show in person his ideas for the research, presenting the results and answer questions of other researchers to prove the efficiency of the results.

8. Transition to Industry: Upon validation from the scientific community, the conducted work should be analysed for a possible industrial application in order to capitalize from it and contribute to the entrepreneurial world. This can be accomplished by passing the developed work from a prototype stage to a fully functional industry application which can be presented to various enterprises and businesses.

1.4. Research Questions and Problems

- How can the semantic consistency of the data exchanged between enterprises information systems be checked?

1.5. Hypothesis

- With a proper framework that provides guidelines for semantic consistency checking complemented with possible resolutions for each case, the data exchange between enterprises is facilitated and its understanding maintained.

1.6. Dissertation Outline

The first section of this work is the *Introduction*, which addresses the purpose of this work as well as the main ideas that led to the creation of this dissertation. Furthermore, it presents the authors motivation behind this work in addition to the background observation that was conducted and the adopted research method. Finally, this section identifies the research questions and problems that this dissertation addresses and the hypothesis for attempting to solve them.

Section 2 is named *Ontology Based Solutions for Knowledge Representation* and addresses the background research that was conducted. It covers the main tools for building ontologies as well as techniques and operations that can be applied on ontologies.

Section 3 is named *Semantic Checking Framework* and covers a background research about interoperability and consistency checking in ontologies. Furthermore this section introduces the framework that is proposed in this work as a solution to the semantic checking of information systems issue.

The next section (4), *Application Scenarios*, describes two situations where the proposed framework was applied. Firstly a mechanical scenario is presented, that features the interaction between a bolt retailer and a manufacturer. The second scenario refers to the Envisioning, Supporting and Promoting Future Internet Enterprise Systems Research through Scientific Collaboration (ENSEMBLE) project. The described scenarios were also used to demonstrate the validity of the ideas presented in this work.

Section 5 is called *Proof-of-Concept Implementation* and as its name indicates, features the architecture of the developed prototype, the technologies used to develop it and why they were chosen. Furthermore it is presented the execution flow of the prototype to serves as a complement to the architecture in the sense that it shows in detail the flow of the system. Furthermore, this chapter presents and describes in detail the involved elements in the system, namely the EISB (Enterprise Interoperability Science Base) Reference Ontology and the FInES (Future Internet Enterprise Systems) wiki.

The following section is the *Synchronization Tool Demonstration* chapter which shows the results of the implemented prototype by featuring some execution examples of the developed prototype.

Finally this document comes to a close with the *Conclusions and Future Work* chapter where, as indicated by its name, the concluding remarks and future work topics are presented. Furthermore, this section also intends to prove that the Hypothesis is valid, or not, regarding the Research Questions and Problems identified in the beginning of this work.

2. ONTOLOGY BASED SOLUTIONS FOR KNOWLEDGE REPRESENTATION

This chapter comprehends the state of the art study regarding ontology operations, reasoners and management and visualization tools. This study focuses mainly on ontologies since they are capable of encoding the knowledge of a certain domain in machine-processable form to make it available to other information systems [7]. Therefore ontologies have been widely adopted as mechanisms to represent knowledge on a given domain.

This chapter is structured as follows; firstly, some ontologies operations are presented and described, as well as the concept of ontology learning. Following is the study of selected ontology management and visualization tools. Finally, the review of certain ontology reasoners is presented.

2.1. Ontology Operations & Learning

Ontology operations usually refer to the methods used to integrate two or more ontologies, while ontology learning refers to the fact of extracting ontological elements in order to build new ontologies. A summary of the ontology operations that are going to be discussed in detail in the following subsections are:

- Ontology mapping/matching;
- Ontology alignment;
- Ontology merging;

After the execution of any of these operations the user should check the resulting ontology for inconsistencies or loss of information [8].

To conclude this subsection, the concept of ontology learning is described and presented in detail.

2.1.1. Ontology mapping/matching

As referred by the de Bruijn et al in [9], ontology mapping is a (declarative) specification of the semantic overlap between two ontologies.

This operation consists in mapping or matching each entity (class, relation, attribute, etc.) of an ontology to the corresponding entity in another ontology, as illustrated in Figure 2.1. The corresponding entities must have the same meaning, which means that usually the correspondences are expressed in a one-to-one fashion. This process won't modify the involved ontologies, and as a result the mapping operation will only produce a set of correspondences. [8]

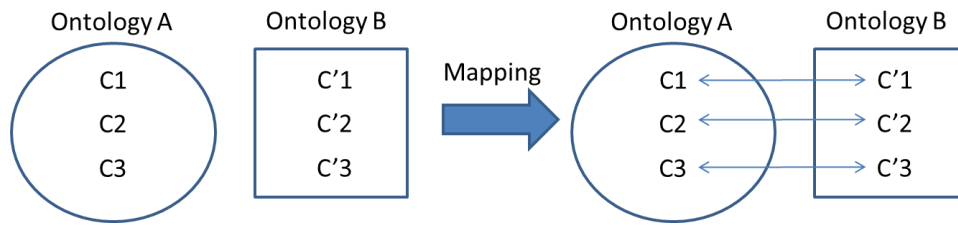


Figure 2.1 - Ontology mapping/matching

2.1.2. Ontology alignment

Much like the mapping process, in the alignment operation the original ontologies persist with links established between them [10], which is why this operation is often considered a synonym of ontology mapping. However, the original ontologies might suffer alterations because this process implies a mutual agreement between the ontologies in order to make them aligned and coherent with one another, eliminating unnecessary information [8]. This is why this method is usually applied when the involved ontologies cover domains that are complementary to each other. This way the original ontologies are more likely to remain unaltered diminishing the likelihood of occurring inconsistencies of information. As illustrated in Figure 2.2, the two original ontologies (A and B) were aligned so that the resulting ontology of the operation, in this case, consists of the greyish area of ontology A.

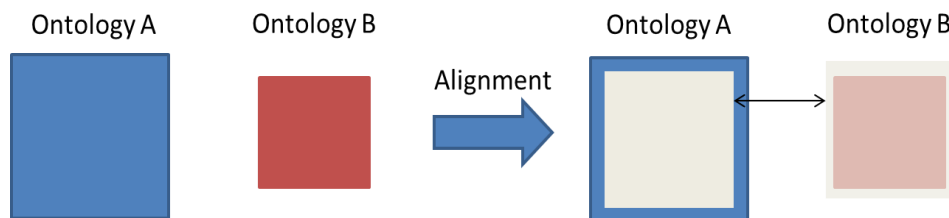


Figure 2.2 - Ontology alignment

2.1.3. Ontology merging

The process of ontology merging consists in integrating or merging two or more existing ontologies to form a new ontology. In this operation, the source ontologies are usually discarded and only the new ontology remains active. Although in some cases the source ontologies could also remain active after the merging process. In the merging operation, often the original ontologies cover similar or overlapping domains [10].

According to de Bruijn et al in [9] there are two approaches to the ontology merging operation. In the first approach, the input of the merging process is a collection of ontologies and the outcome is one new, merged, ontology which captures the original ontologies. In the second approach the original ontologies are not replaced, but rather a 'view', called bridge ontology, is created which imports the original ontologies and specifies the correspondences using bridge axioms.

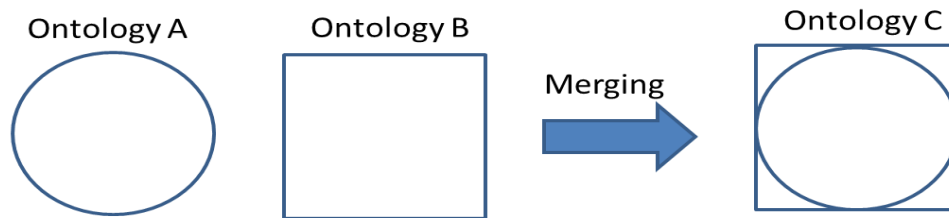


Figure 2.3 - Ontology merging

Figure 2.3 shows a small example where ontologies A and B are merged together to form a new ontology (C) that consists of the source ontologies.

It is worthy of note that the result of the merging process (or any other that promotes changes to the ontologies) should be tested in order to identify inconsistencies or loss of information [8].

2.1.4. Ontology Learning

Ontology Learning refers to extracting ontological elements (conceptual knowledge) from input and building an ontology from them [11]. Furthermore, within the research community, ontology learning is mainly associated to the process of discovering ontological knowledge from various forms of data [13]. According to Cimiano et al in [12] there are three kinds of data to which ontology learning can be applied, which are, structured data (e.g. databases), semi-structured data (e.g. HyperText Markup Language - HTML or Extensible Markup Language - XML) and unstructured data (e.g. text) documents. However, it can also be used as support to the refinement and expansion of existing ontologies that could have been built following a traditional basis by means of incorporating new knowledge in an automatic way [13].

To achieve the goal of discovering ontological knowledge from various forms of data, diverse ontology learning techniques have been developed. These serve the purpose of supporting an ontology engineer in the task of creating and maintaining an ontology [12]. Most of these techniques are drawn from well-established disciplines such as machine learning, natural language processing, statistics, knowledge acquisition, information retrieval, artificial intelligence, reasoning and Database (DB) management [11][14]. However these techniques are not exclusive to one another, i.e., they can be combined to form a more powerful method to achieve the goals of ontology learning. For example, linguistic-based methods are commonly applied with statistical approaches to calculate the relevance of concept to the given domain, these methods include techniques based on linguistic patterns, pattern-based extraction, methods that measures the semantic relativeness between terms within a domain.

2.2. Ontology Management Tools

Ontology management tools are pieces of software that enable the user to create, edit or perform

other operations on ontologies. As referred by Youn, S et al in [15], ontology tools can be applicable for all stages of the ontology life cycle (creation, population, validation, deployment, maintenance and evolution). These tools support a variety of ontology languages such as the Web Ontology Language (OWL), Resource Description Framework (RDF) or XML which are used to implement the ontologies. In this subsection three ontology management tools are presented, Protégé, Ontopia and Topic Maps 4 E-Learning (TM4L), although there are many more.

2.2.1. Protégé

Protégé is a free, open-source platform, with a suite of tools to construct domain models and knowledge-based applications with ontologies [16]. This tool allows the user to perform numerous ontology operations, such as creating, populating, validation or visualization. It also enables the creation of domain ontologies, definition of classes, class hierarchies, variable-value restrictions, and the relationships between classes and the properties of these relationships [16]. Apart from these features, Protégé also allows the user to export or import ontologies provided they are in OWL/XML or RDF/XML formats. Regarding the Graphical User Interface (GUI), Protégé consists of a tab navigation system, much like a web browser, allowing for a much smoother learning curve. Navigating through the tabs the user can easily see the entities, classes, instances and relations that compose the ontology, as illustrated in Figure 2.4.

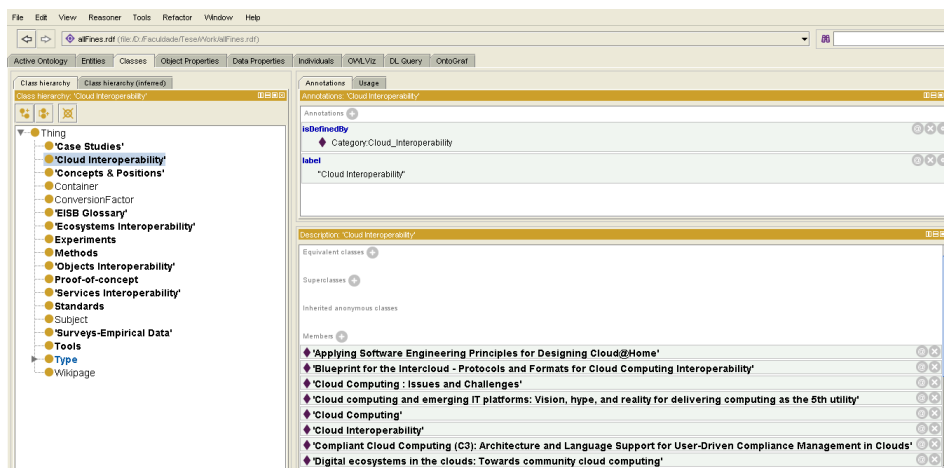


Figure 2.4 - Snapshot of the Protégé GUI

2.2.2. Ontopia

Ontopia is an open source suite of tools for building applications based on topic maps [17]. As a side note, topic maps are an ISO standard for describing knowledge structures and associating them with information resources. As such they constitute an enabling technology for knowledge management [18]. This ontology management tool has essentially three main components. The first component is the ontology editor named Ontopoly that allows the user to incrementally design topic map ontologies

using a user-friendly web interface, as shown in Figure 2.5. The Ontopoly editor also provides the user the possibility to populate the ontologies and to store them in files or databases [19].



Figure 2.5 - Ontopoly snapshot

The second main component of Ontopia is the ontologies browser called Omnigator and has a variety of features. It is web-based and can be used to display any topic map [20], as illustrated in Figure 2.6, whether the topic map was created with the Ontopia editor (Ontopoly) or imported from another ontology editor (e.g. Protégé). Additionally, the Omnigator also features an exportation plugin, that allows saving the ontology into various file formats such as RDF, XML Topic Maps (XTM 1.0, 2.0 or 2.1) or Linear Topic Map (LTM), a topic map query interface, topic maps validation, statistics and merging. One great advantage of this tool is that it allows the user to follow links associated to classes or instances. For example, navigating to a class through omnigator one could follow the link associated with that class and be redirected to the designated web page.

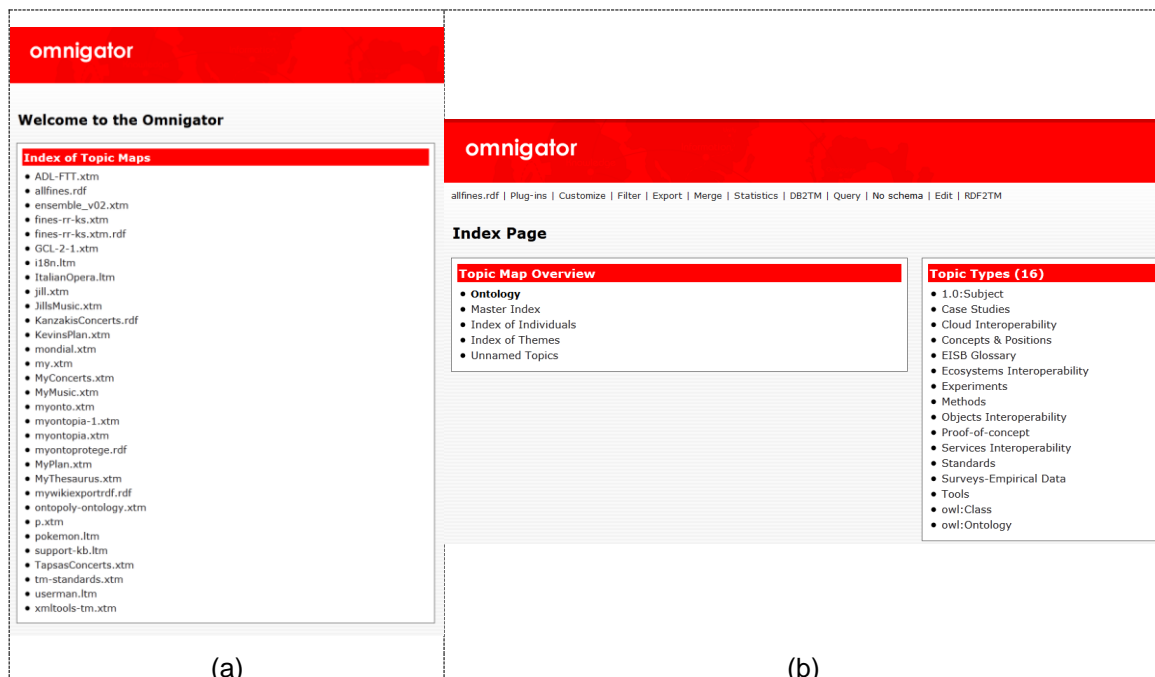


Figure 2.6 – Omnigator snapshots - (a) Omnigator Main Page with index of topic maps; (b) Browsing a topic map

Finally, the third main component of the Ontopia tool suite is the graphical visualization feature named Vizigator (visual navigator). Since section 2.3.1 is dedicated to this component, there won't be a detailed description of it here. However, as a very brief and short introduction, the Vizigator is used to

show graphical visualizations of topic maps and is subdivided in two components, the VizDesktop and the Vizlet.

2.2.3. TM4L

The TM4L tool is somewhat similar to Ontopia, in a sense that it also uses the topic maps technology to manage ontologies. However, Ontopia is web-based and TM4L is more of a “standalone” or “offline” product. This tool provides support in conceptual structure design and maintenance through its functionality for editing, browsing, and combining such structures, coupled with support for relating concepts, linking concepts to resources, merging ontologies, external searching for resources, defining perspectives, etc.[21]. TM4L has a user-friendly interface, which guides the users to create and update topic as well as their relations and resources [21]. This tool is divided into two constituents, the editor and the viewer.

The TM4L editor is what allows the user to create, edit and manage ontologies using topic maps. About formats, TM4L saves the topic maps in the XTM format by default, however TM4L comes equipped with a XTM to RDF converter granting compatibility with RDF applications, such as Protégé, for example. Since this is a topic maps based tool, the main objects it manipulates are topics (representing domain ontology concepts), relationships between them, resources, and contexts (represented by themes) [21]. Regarding the user interface, TM4L uses a tab navigation system, as seen in Figure 2.7 similar to the one used in Protégé, from which the user can access the topic map, the topics, relationships, themes and the graphical visualization of the topic map.

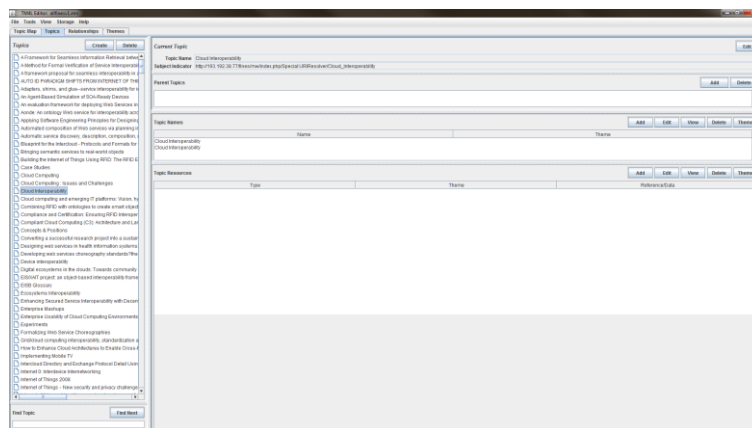


Figure 2.7 - Snapshot of TM4L user interface

Regarding the TM4L viewer, it will be described in greater detail in section 2.3.4. However as a very brief description, the TM4L viewer displays the topic map in graph like format where the topics and instances (in different colours) are nodes of the graph and the different relations are lines (also in different colours) connecting them.

2.2.4. Ontology Management Tools Concluding Remarks

In conclusion of this section, Table 2.1 is presented in which a comparison of the main features of the described ontology management tools is conducted. Namely, the characteristics being compared are the supported file formats for import and export and if the management tool provides means for a graphic visualization of ontologies.

Table 2.1 - Comparison between Ontology management tools

Ontology Management Tool	Import Format	Export Format	Graphic Visualization
Protégé	RDF, OWL	RDF/XML, OWL/XML in all versions. In versions 3.4.x, CLIPS, N-TRIPLE, N3, TURTLE. In versions 4.x, KRSS2, OBO 1.2, Latex.	Yes. In versions 3.4.x through plugins like Jambalaya. On versions 4.x through plugins like OntoGraf
Ontopia	RDF, XTM, CTM, TM/XML	XTM 1.0, XTM 2.0, XTM 2.1, RDF/XML, CXTM, LTM and TM/XML	Yes, through the Vizigator tool
TM4L	XTM, RDF (though to work RDF must be converted to XTM)	XTM, RDF (through the XTM to RDF converter tool)	Yes, through the TM4L Viewer

As seen in this table, they all seem to be very complete, since they all provide support for various file types and graphical visualization methods. However, Protégé is more adequate for beginning ontology development since it has a more user-friendly interface and has a smoother learning curve. Nonetheless, the choice between which tools to use should come down to the needs of each user. If topic map technology is used, then Ontopia and TM4L are best suited, with Ontopia being more complete, specifically regarding the supported file formats. On the other hand, if OWL or RDF files are used to store the ontology then Protégé is the best choice.

2.3. Ontology Visualization

Ontology visualization refers to the graphical visualization of ontologies. These representations can be accomplished by means of directed or nested graphs, topic maps or other techniques. However this isn't an easy operation to accomplish, because ontologies are more than just a hierarchy of concepts [22]. They are the sum of various relations and attributes between classes and entities, and in turn, these can have a wide number of instances, so it can be difficult to represent ontologies effectively. It is worthy of note that the examples used to take the snapshots for the figures were taken from the FInES wiki [23], upon extraction of its contents to an RDF file. The examples will highlight the cloud interoperability wiki category (class) and all of its pages (instances).

In the following subsections some examples of ontology visualization tools are described in detail.

2.3.1. Ontopia Vizigator

The Vizigator (visual navigator) is an ontology visualization tool from the Ontopia tool suite that displays ontologies in form of topic maps, as illustrated in Figure 2.8

It shows graphical visualizations of the structure of a topic map for seeing larger patterns in complex data, or simply as a visually attractive and user-friendly alternative way of displaying the topic map [24].

It was also said in the Ontopia dedicated section that the Vizigator tool has two main components, the VizDesktop and the Vizlet. The first component provides a graphical interface where the user can load a topic map or ontology to display, in a variety of formats including RDF, XTM, Compact Topic Maps (CTM) and LTM, and configure the visualization through a set of operations like filtering and scoping. These options enable the user to configure which associations, classes or instances to show, or what colours and shapes represent the various components of the ontology. In short the user can fine tune the display to ensure the best results. The second component refers to a Java applet for displaying visualizations on the web which is called the Vizlet [24].

Setting up the visualization requires no programming, the user only has to create a configuration in VizDesktop and deploy the applet together with the necessary web service interface on the server side [24].

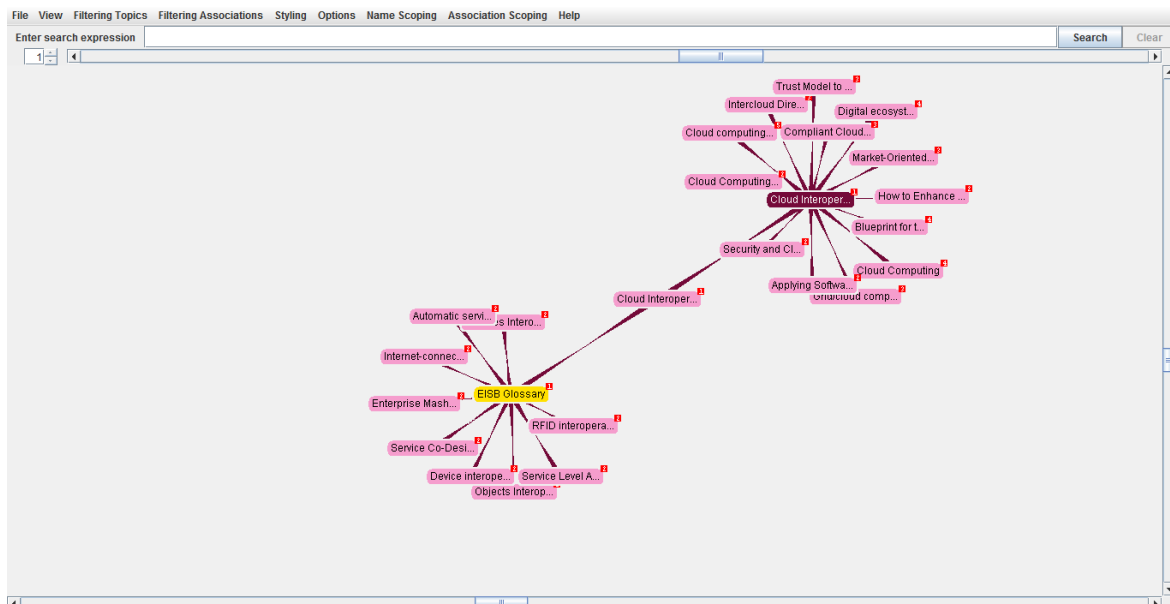


Figure 2.8 - Ontopia Vizigator snapshot

2.3.2. Jambalaya

Jambalaya is a plugin created for Protégé that uses Shrimp (Simple Hierarchical Multi-Perspective) to visualize the user created ontologies.

The Shrimp visualization technique uses a nested graph view to present information that is hierarchically structured. It introduces the concept of nested interchangeable views to allow a user to explore multiple perspectives of information at different levels of abstraction [25].

In Jambalaya, there are many types of views available. The user has choices that range from the nested graph to the sink tree views. Furthermore the user is able to choose the layout of those views, such as radial or grid layouts. The classes and instances are represented as nodes in the graph. However they are represented differently according to view type chosen, as shown in Figure 2.9. In the nested view, the classes (or instances) are represented within the class they belong to, that is they are nested inside their superclass node. As for the sink tree view, the classes and instances are still represented as nodes, though the relations are represented by directed arcs connecting them. Apart from this visualization features, Jambalaya also allows the user to filter contents of the visualization, to search for a specific class, instance or relation or zoom in or out for a more detailed or more generic view. These features result in an environment where the user can interact directly with the information space enhancing their understanding of the information structures, thus promoting further exploration [25].

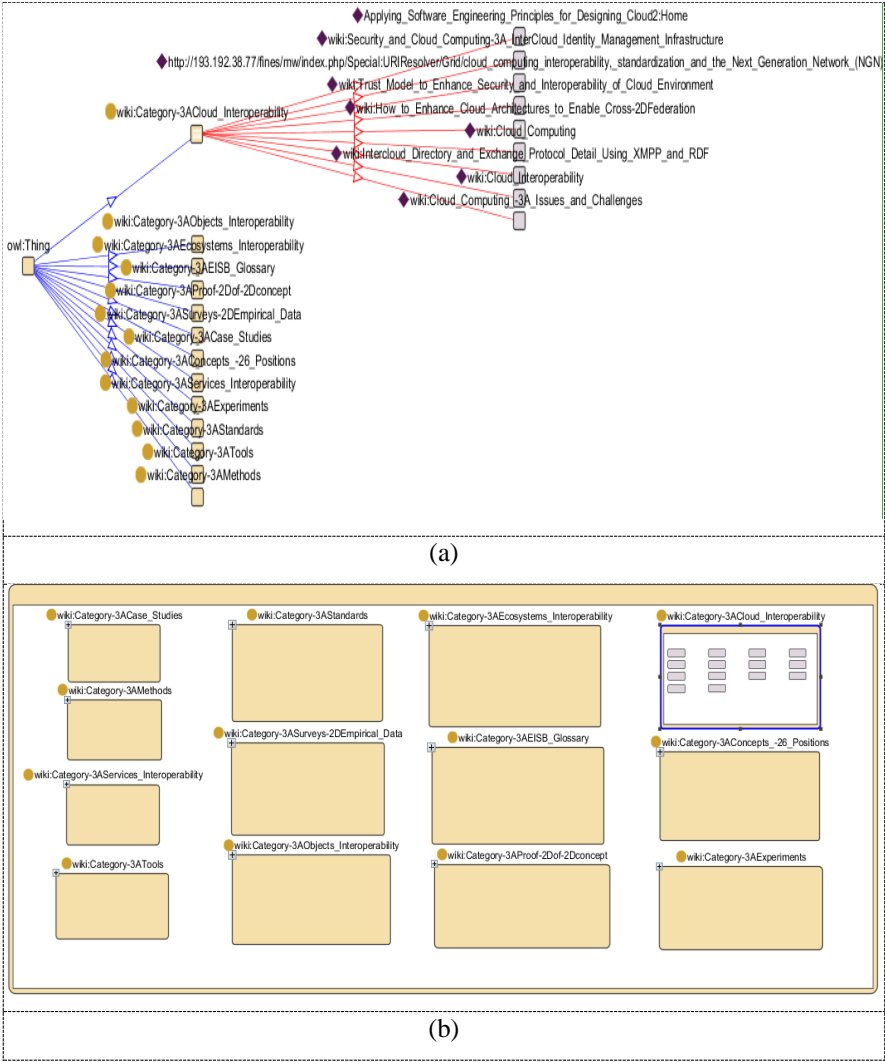


Figure 2.9 - Jambalaya snapshots (a) Sink Tree view; (b) Nested Graph view

2.3.3. OntoGraf

The OntoGraf is an ontology visualization tool available as a plugin for Protégé versions 4.x. It gives support for interactively navigating the relationships of OWL ontologies and it also supports various layouts for automatically organizing the structure of the ontology [26]. Much like the other visualization tools described, OntoGraf displays all information regarding a class (subclasses, instances, etc.) and it also represents the various relationships which are represented by directed arcs and differentiates them through different colours.

It is a very similar tool to Jambalaya since it provides similar views, however it doesn't feature the nested graph view (figure 2.9 (b)). On the other hand it is able to better present complex information than Jambalaya as one can see by comparing Figure 2.9 (a) and Figure 2.10 that represent exactly the same scenario gathered from the FInES wiki [23]. Jambalaya depicts a confusing scenario, where the labels of the classes and instances are all overlapping. On the contrary OntoGraf is able to keep things very neat, clearly representing all the classes and instances with the labels being completely readable and all the relationships also clearly visible.

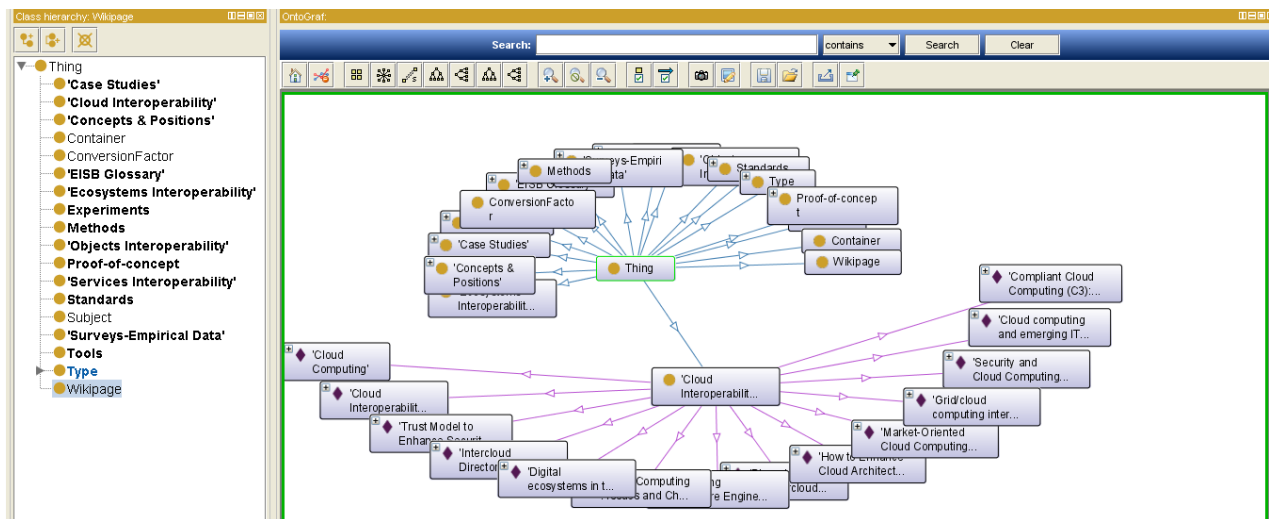


Figure 2.10 - OntoGraf snapshot

2.3.4. TM4L Viewer

The TM4L Viewer displays the topic maps using a graph, where the topics and instances are represented as nodes of the graph (with different colours) and the relations are represented as lines connecting the nodes also with different colours (depending on the type of relation). It is worthy of note that the relations and nodes are labelled so that the user can easily see what they are and their relation. Moreover this tool also has a hierarchical tree view where the user can easily observe the instances and relations of a topic in a more structured manner. Apart from these features the TM4L Viewer also provides a topic maps index where the user can choose between listing topic types, relationships, subject topics, relationship types, resource types, member types and themes (contexts).

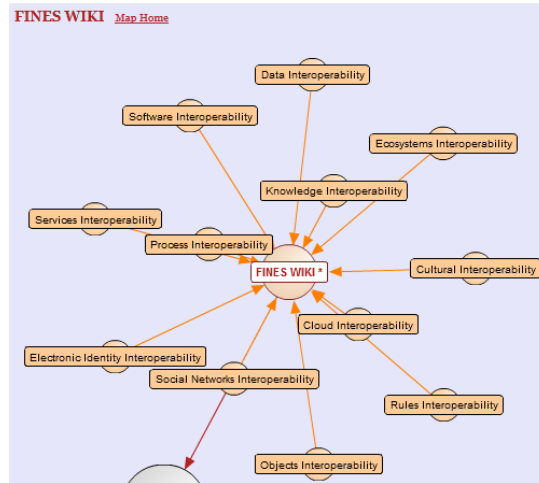


Figure 2.12 - Snapshot of the debateGraph visualization tool

2.3.6. TheBrain

This tool is based on the mind map technology and can be used as a mean for ontology visualization. It uses a graphical layout of topics connected by lines that radiate out from a central topic [28]. However it is a very dynamic tool since any topic can be the central one as the user shifts contexts or changes the focused topic. Up to this point, the Brain tool seems very similar to the other ontology visualization tools already presented. However this tool has some features that the others do not. One of these features is the possibility of attaching files or URL's to each topic allowing the user to be redirected to those sources thus providing complementary information about the topic. Another important feature is the possibility of uploading and sharing the created mind map to a website using simple HTML code, thus allowing other users to navigate online through the map. Figure 2.13 represents the same example gathered from the FInES wiki that was used in the previous ontology visualization tools. As can be seen, this tool centres the focused topic and arranges the other topics neatly in the side so that they can easily be selected if the user so desires.

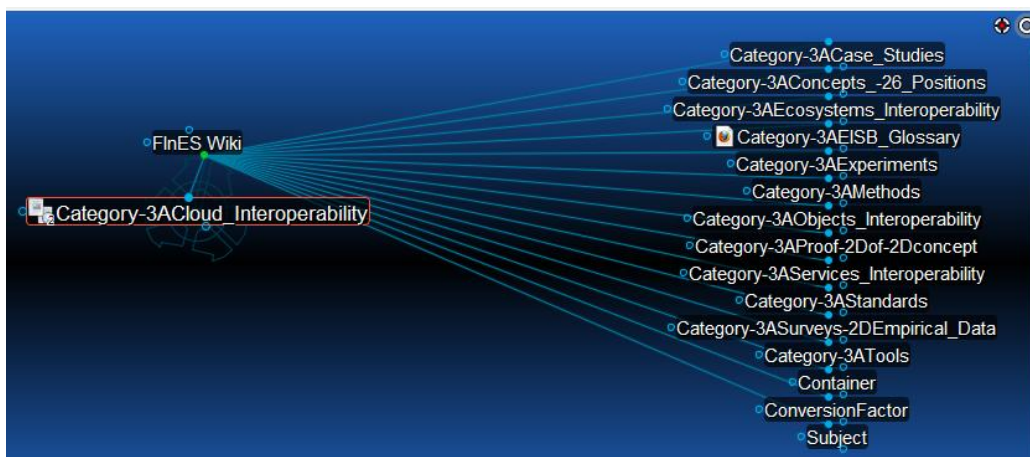


Figure 2.13 - Snapshot of theBrain visualization tool

2.3.7. XMind

XMind is an open source tool that contributes to building a cutting-edge brainstorming/mind-mapping facility, focused on both usability and extendibility [29]. The structure in XMind contains a root in the center, with main branches radiating from it, similarly to “theBrain” tool. Its features contemplate several mind map templates, the ability to import and export mind maps in a variety of file formats and it can also be shared on the web or embedded in a webpage [30]. This tool can be of great use in terms of ontology visualization because the information can be arranged as to maintain good readability and more importantly it can clearly represent the class hierarchy, as well as the properties that relate the several classes. However a major downside to this tool is that it doesn’t work with ontology files such as, OWL or RDF, thus the classes and properties have to be built manually, which for complex ontologies, can be very error-prone and extenuating.

Figure 2.14 shows an example gathered from the FInES wiki, and as can be observed, it contains a root topic, and its branches represent classes, while the blue dotted lines represent the relations between them. This example can attest to the capability of this tool to represent the relations and class hierarchy of an ontology, although this is mainly a mind mapping tool.

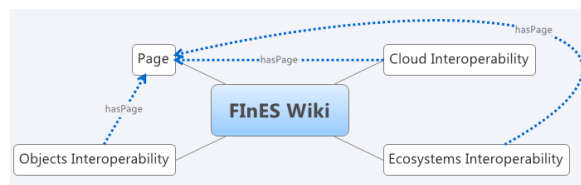


Figure 2.14 - Snapshot of the XMind visualization tool

2.3.8. Ontology Visualization Tools Concluding Remarks

To conclude this section Table 2.2 is presented, where the studied ontology visualization tools are compared regarding their supported file formats, possibility of embedding the visualization online, support for multiple users and elements disposition and readability.

Table 2.2 - Comparison between ontology visualization tools

Ontology Visualization Tool	Supported File Formats	Online Embedding	Multiple Users Support	Readability
Ontopia Vizigator	XTM, CTM, LTM, RDF and TM/XML	Yes. Through Java applet + web service interface	Yes	Medium
Jambalaya	OWL 1.0, RDF	No	N.A.	Bad
OntoGraf	OWL, RDF	No	N.A.	Medium
TM4L Viewer	XTM and LTM	No	N.A.	Medium
DebateGraph	N.A.	Yes	Yes	Good
theBrain	XML, DOCX, MMAP, XMMAP, OPML, MM, OWL and TXT	Yes	Yes	Good
XMind	XMIND, MMAP, XMP and MM	Yes	Yes	Good

At first glance all of the presented visualization tools seem similar since all of them represent the concepts similarly to a topic map, with the focus on one topic and linking related topics through lines or

arcs. However when their specifications are more thoroughly analysed, differences between them begin to emerge, as shown in the table. Beside these differences, one cannot clearly state that a tool is better than the other. Still, depending on the technology used to develop the ontologies or their end use, some tools can be more suited than others. For example, if topic map type files are used then perhaps it is best to use Ontopia's Vizigator or the TM4L Viewer. On the other hand if the ontologies are developed using the OWL or RDF file formats then the Jambalaya and OntoGraf tools are perhaps more suited for a better visualization. Furthermore if the end use for the visualization is an online application then DebateGraph or theBrain or even XMind are more suited as they offer a more simple solution for online integration. The multiple users feature relates to the capability of the tool to support users editing or viewing the ontology at the same time. Unfortunately this feature could not be tested for the Jambalaya, OntoGraf and TM4L viewer tools, hence the "Not Applicable" (N.A.) value. Lastly there's the readability attribute, which is evaluated according to three levels, "bad", "medium", and "good". The lowest value is "bad" and means that the elements aren't clearly shown or the labels aren't read easily, signifying that the concepts are piled on top of each other creating a lot of confusion and not allowing a good overview of the structure of the ontology. The "medium" value means that the concepts are still presented somewhat confusingly, however it is possible to have a better overview of the ontologies structure. The highest value for this attribute is "good" and it means that the elements are neatly shown, all the labels are easily readable and the structure of the ontology is well represented. It is also worthy of note that the readability attribute refers to large or complex ontologies, since for simple or small ontologies, all of the tools perform satisfactorily.

2.4. Ontology Reasoners

Reasoners are key components for working with OWL ontologies. In fact, querying an ontology should be done using reasoners. The reason for this is that knowledge in an ontology might not be explicit and a reasoner is required to deduce implicit knowledge so that the correct query results are obtained [31]. These tools work based on description logic, where logical consequences are inferred, using an inference engine, based on a predefined set of rules and are often based on a hypertableau algorithm [32]. Reasoners are often used paired with ontology editing tools, like the ones previously presented, with the objective of computing the class hierarchy and alert users to inconsistencies within the ontology [33].

In this subsection four of the most known description logic reasoners will be presented, HermiT [34], Pellet [35], FaCT++ [36] and RacerPro [37].

2.4.1. HermiT

HermiT is an open source ontology reasoner that given an OWL file, can determine whether or not an ontology is consistent, identify subsumption relationships between classes, among other functions [34]. This reasoner has essentially three modes of operation. It can be used as Protégé plugin, from

the command line or in java applications [39].

- **HermiT as Protégé plugin**

In this mode of operation, HermiT can be accessed directly from the Protégé GUI from a drop down menu on the menu bar. When the reasoner is run the consistency of the ontology is assessed. If the ontology is inconsistent, a pop up message appears to alert the user to that fact, as shown in Figure 2.15. On the other hand, if the ontology is consistent, the results can be seen by choosing to view the inferred components from the Protégé GUI as illustrated in Figure 4.4.

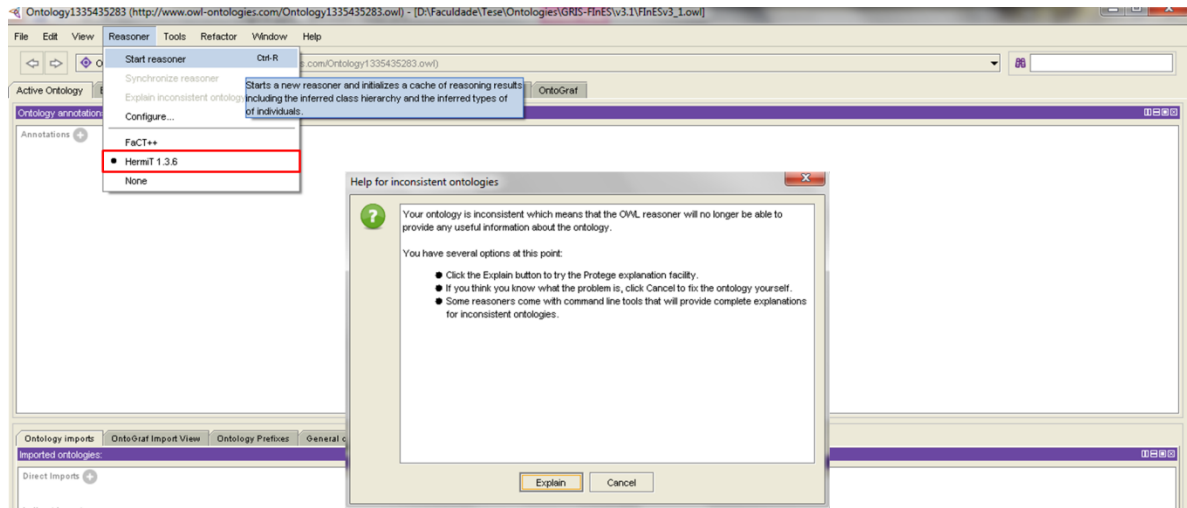


Figure 2.15 - HermiT reasoner Protégé plugin output - inconsistent ontology

- **Using HermiT from the command line**

When HermiT is used from the command line, different common reasoning tasks can be configured for the reasoner to perform. In the example featured in Figure 2.16 HermiT was used to classify an ontology, outputting the class hierarchy. The command to invoke HermiT from a shell is “java -jar HermiT.jar” followed by the arguments that serve to tell which operation the reasoner is to perform.

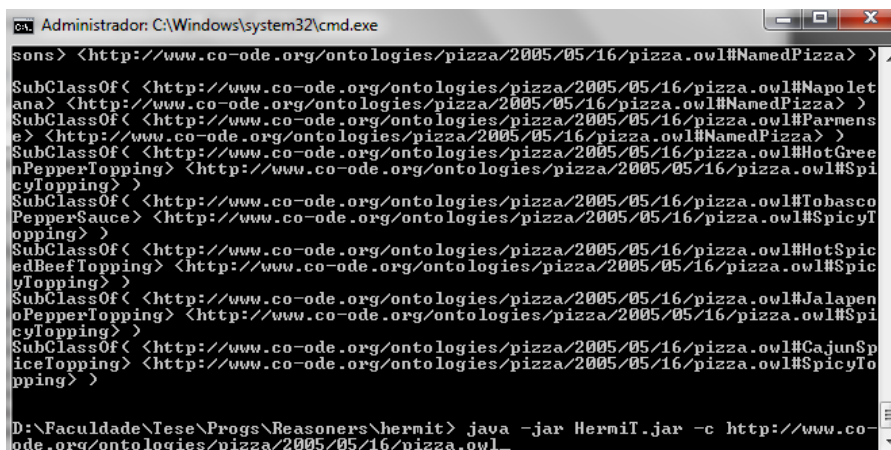
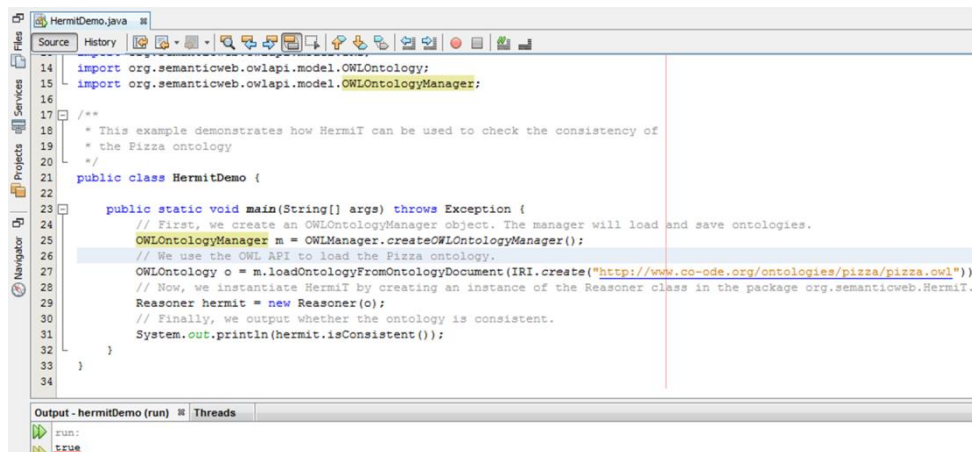


Figure 2.16 - HermiT reasoner example using the command line

- **Using HermiT in java applications**

This reasoner can be used in java applications through the OWL Reasoner interface that is available in the OWL Application Programming Interface (API). It can be used to integrate HermiT with user developed applications or tools. In the example shown in Figure 2.17, a simple demo application was created where the consistency of an ontology is tested. If the ontology is consistent the program returns the Boolean value “true”, else if it isn’t consistent the program returns the Boolean value “false”.



```
14 import org.semanticweb.owlapi.model.OWLOntology;
15 import org.semanticweb.owlapi.model.OWLOntologyManager;
16
17 /**
18  * This example demonstrates how HermiT can be used to check the consistency of
19  * the Pizza ontology
20  */
21 public class HermitDemo {
22
23     public static void main(String[] args) throws Exception {
24         // First, we create an OWLOntologyManager object. The manager will load and save ontologies.
25         OWLOntologyManager m = OWLManager.createOWLOntologyManager();
26         // We use the OWL API to load the Pizza ontology.
27         OWLOntology o = m.loadOntologyFromOntologyDocument(IRI.create("http://www.co-ode.org/ontologies/pizza/pizza.owl"));
28         // Now, we instantiate HermiT by creating an instance of the Reasoner class in the package org.semanticweb.HermiT.
29         Reasoner hermit = new Reasoner(o);
30         // Finally, we output whether the ontology is consistent.
31         System.out.println(hermit.isConsistent());
32     }
33 }
34
```

Output - hermitDemo (run) | Threads

```
run:
true
```

Figure 2.17 - HermiT reasoner java application integration example

2.4.2. Pellet

Pellet is an OWL description logic reasoner that features standard reasoning services, such as, consistency checking, concept satisfiability, classification and realization [40]. As it happens with the HermiT reasoner, Pellet also has multiple interfaces from which users can access its reasoning capabilities, for instance, a command line interface, an API and as a Protégé plugin. The command line interface is more suited for simple reasoning tasks, while the API is better for standalone applications and the Protégé plugin is useful when the ontology is being developed using that editor.

An example of consistency checking of an ontology using this reasoner is shown in Figure 4.3.

The procedure to using this reasoner as a Protégé plugin is the same as the one described for HermiT. An example of consistency checking with this interface is shown in Figure 2.18, where the consistency of an ontology is tested with the result being that it is inconsistent.

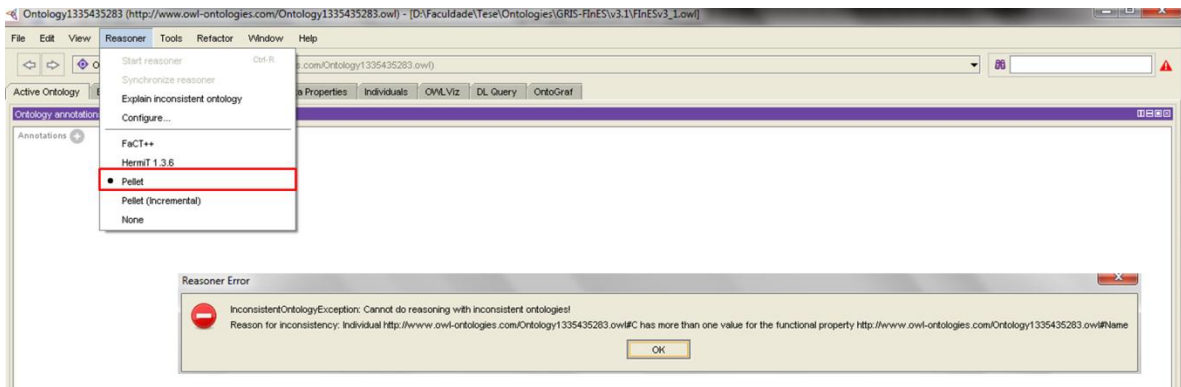


Figure 2.18 - Pellet reasoner Protégé plugin output - inconsistent ontology

2.4.3. FaCT++

FaCT++ is also an open source OWL description logic reasoner that uses FaCT algorithms, but with a different internal architecture [36]. This reasoner can be used as standalone reasoner, as back-end reasoner for an OWL API based application [38] or as a plugin for the Protégé ontology editor. FaCT++ is implemented using C++ in order to create a more efficient tool, and to maximise portability [36]. As happens with the previously presented reasoners, FaCT++ is also capable of verifying the consistency of OWL ontologies and classifying the ontology to compute the class hierarchy.

The example featured in Figure 2.19, illustrates the output of the execution of the FaCT++ reasoner, as a Protégé plugin, on an inconsistent ontology.

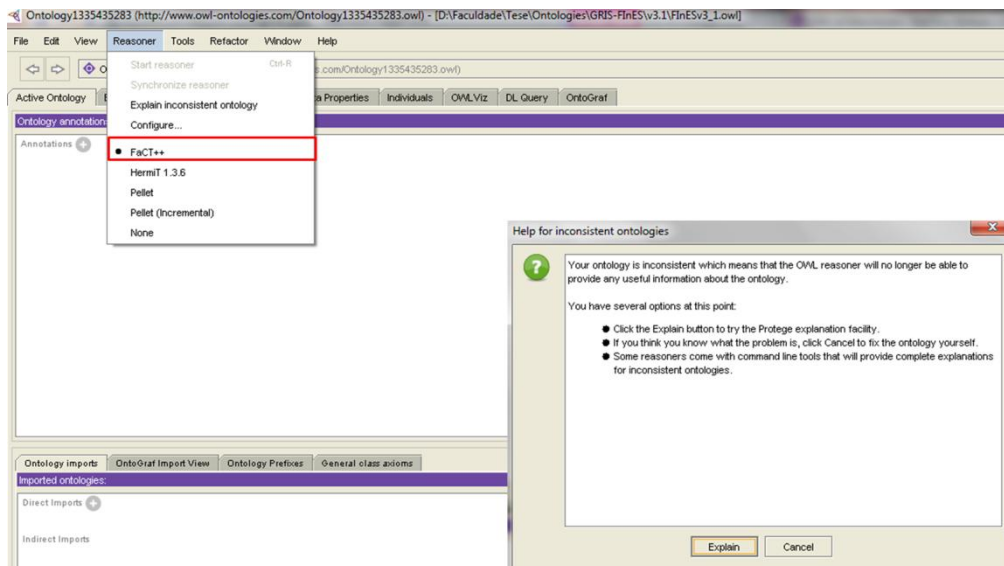


Figure 2.19 - FaCT++ reasoner Protégé plugin output - inconsistent ontology

2.4.4. RacerPro

The Renamed ABox and Concept Expression Reasoner (RacerPro) is a description logic reasoner for OWL or RDF ontologies [37]. It can be used as a plugin for Protégé, via an http/XML DIG protocol or it

can be used on a standalone application via a Java or LISP API. Its main functionalities include [41]:

- Check the consistency of an OWL ontology and a set of data descriptions.
- Find implicit subclass relationships induced by the declaration in the ontology.
- Find synonyms for resources (either classes or instance names).
- Incremental query answering for information retrieval tasks (retrieve the next n results of a query). In addition, RacerPro supports the adaptive use of computational resource: Answers which require few computational resources are delivered first, and user applications can decide whether computing all answers is worth the effort.

To have a better understanding of its features, Figure 2.20 is presented, which illustrates the technologies that this reasoner integrates and supports.

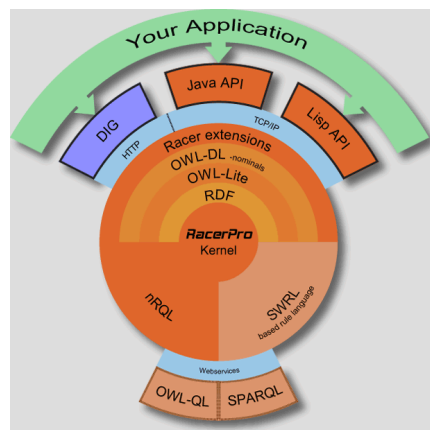


Figure 2.20 - RacerPro reasoner supported features [41]

2.4.5. Ontology Reasoners Concluding Remarks

As a conclusion to this subsection Table 2.3 is presented where some features of the presented reasoners are put side by side for a better general view. It isn't the objective of this work to make an exhaustive comparison of these reasoners, but it is suffice to say that these tools are quite similar to each other varying only in their architectures, implementations and speed of execution of the reasoning tasks.

Table 2.3 - Ontology Reasoners Comparison

Ontology Reasoner	User Interface	Ontology Consistency Checking	Ontology Classification	Standalone applications integration
HermiT	Command line, Protégé plugin, API	Yes	Yes	Yes
Pellet	Command line, Protégé plugin, API	Yes	Yes	Yes
FaCT++	Command line, Protégé plugin	Yes	Yes	N.A.
RacerPro	Protégé plugin, API	Yes	N.A.	Yes

As seen in the table, regarding the user interface, all of the presented reasoners can be used as a plugin for the Protégé ontology editor. This a great benefit because the consistency of the ontology can be checked as its being developed. The command line feature is also useful because it allows a direct consistency checking of the ontology without having the need of additional programs, however

RacerPro doesn't implement this feature. An API is particularly useful when integrating reasoning features to user developed applications. Out of the studied reasoners, only FaCT++ doesn't implement this feature. Regarding the consistency checking of ontologies, all of the reasoners are capable of doing so, since it's their main objective. Referring to the classification of an ontologies taxonomy, only RacerPro doesn't have this capacity. Finally, the integration of reasoning features with standalone applications isn't accomplished by FaCT++ since it doesn't provide an API.

In spite of the chosen reasoner it can be concluded that these tools are indeed very important upon developing ontologies. They can ensure that the conducted work remains solid and error free during its evolution regarding its consistency.

3. SEMANTIC CHECKING FRAMEWORK

In this chapter, the semantic checking framework proposed by the author is presented along with an extensive description of its purpose and guidelines. In addition, to provide a context as to why and how this framework was developed, a background study on the problematic of systems interoperability and consistency checking is also presented. This study is important because it introduces key concepts to the problematic addressed in this work such as, consistency checking and semantic checking.

3.1. Interoperability

According to the IEEE standards glossary [42] interoperability is the ability of a system or a product to work with other systems or products without special effort on the part of the customer. Still, the popular perception is that interoperability is synonymous with connectivity. However, interoperability is much more than just connectivity. It is also a function of operational concepts and scenarios, policies, processes and procedures [43]. Nonetheless, there are other definitions of interoperability such as the one in [44], which regards interoperability as the ability of a set of communicating entities to exchange specified state data and operate on that state data according to specified, agreed-upon, operational semantics. Interoperability can also be seen in an EI point of view being defined as the ability of interaction between enterprises. The enterprise interoperability is achieved if the interaction can, at least, take place at the three levels: data, application and business process [45]. Despite these different definitions, the one adopted in this work is the one defined in [44] as it is deemed by the author as the most suitable to the topic of this dissertation.

Nowadays, as information systems in enterprises and organizations keep evolving and become more complex, the need for interoperable operation, automated data interchange and coordinated behaviour of large scale infrastructures becomes highly critical [46]. Regarding enterprise systems as layered systems, to achieve meaningful interoperability between enterprises, interoperability must be achieved on all layers [47], as seen in Figure 3.1.

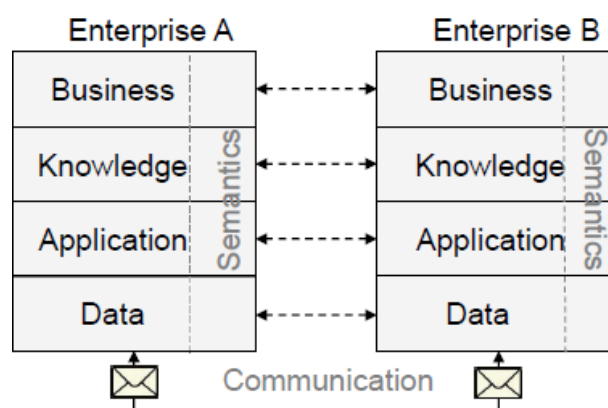


Figure 3.1 - Enterprise Interoperability [47]

Yet, interoperability isn't only a technical issue. The rise of other challenges have led to the categorization of interoperability into several fields, such as, data, organizational, semantic, syntactic, etc. Data interoperability denotes the agreed format in which data is exchanged between collaborating enterprises. Organizational interoperability deals with the ability of enterprises to collaborate and exchange information despite having different internal structures and processes. Semantic interoperability offers cooperating enterprises the ability to bridge semantic conflicts arising from differences in implicit meanings, perspectives and assumptions by creating a compatible environment based on agreed concepts between the entities [48]. Syntactic interoperability allows multiple software components to cooperate regardless of their different implementation languages, interfaces or execution platforms [48].

There are several ways to achieve interoperability, either by implementing standards [42] or, in the case of ontologies, by performing operations to integrate them or by resorting to a methodology, such as MENTOR (Methodology for Enterprise Reference Ontology Development), to build a reference ontology to serve as a bridge between the source ontologies. However, it is needed to take into account that the execution of any operation can result, in some cases, in loss of information. Therefore, after conducting operations to integrate or to make two or more systems interoperable, it is needed to check the consistency of the output, independently of which type of interoperability considered.

3.2. MENTOR Methodology

MENTOR is a methodology that helps an organization to build and adapt a domain reference ontology [49]. MENTOR provides a methodology that allows ontology building from scratch, ontology reengineering, cooperative ontology building and ontology merging methods.

This methodology is comprised of two phases, each with three steps, as seen in Figure 3.2. The first phase (Lexicon Settlement Phase) represents the domain knowledge acquisition and is divided in the following steps:

- Terminology Gathering – In this step all the relevant terms or concepts in a specific domain are gathered, with the all the participants giving their inputs [49]. The terms gathered in this step should reference the contributors so that they can provide their definitions during the next step;
- Glossary Building – In this step, each contribute provides their annotations of the previously established terms. Then the terms enter a cycle where they are reviewed in order to reach a reference definition. This cycle has two possible outputs. If there isn't an agreement then the participants produce a semantic mismatches record for future mappings. On the other hand if everyone agrees on the definitions then the glossary is produced and the process is advanced to the next step;
- Thesaurus Building – This step is constituted by a cycle where the knowledge engineers define a taxonomic structure from the glossary terms [49]. Then the other terms are classified

into semantic proper paths in the existing taxonomic structure down to the thesaurus leafs [49]. Equally to the previous step, the process only advances if there is an agreement between the participants. If an agreement isn't reached then the cycles starts all over again. On the contrary, if there is an agreement then the thesaurus is produced and process advances to the next phase. The defined thesaurus will enhance the ontology harmonization process in the next phase [49].

The second phase (Reference Ontology Building Phase) is where the reference ontology is built and the semantic mappings between the organizational ontologies and the reference one are established [49]. This phase is composed by the following steps:

- Ontologies Gathering – This step comprehends the collection of ontologies or other types of knowledge representation techniques within the specified domain;
- Ontologies Harmonization – This step is supported by two cycles. First there is a discussion about the structure of the reference ontology where the previously defined thesaurus is taken into account. Once again, if an agreement is reached by all, then the cycle is repeated. If a consensus is reached then the taxonomy of the reference ontology is defined. From there the step advances to the second cycle where the contents of the gathered ontologies are harmonized using the semantic mismatches previously recorded. However new mismatches may be found and these need to be recorded as well. When the participants reach an agreement the reference ontology is finalized and the process can advance to the final step;
- Ontologies Mapping – This step is executed whenever there are semantic mismatches to record [49]. These semantic mismatches are used to produce mapping tables that describe the ontological relationships between the reference ontology and each one of the source ontologies [49].

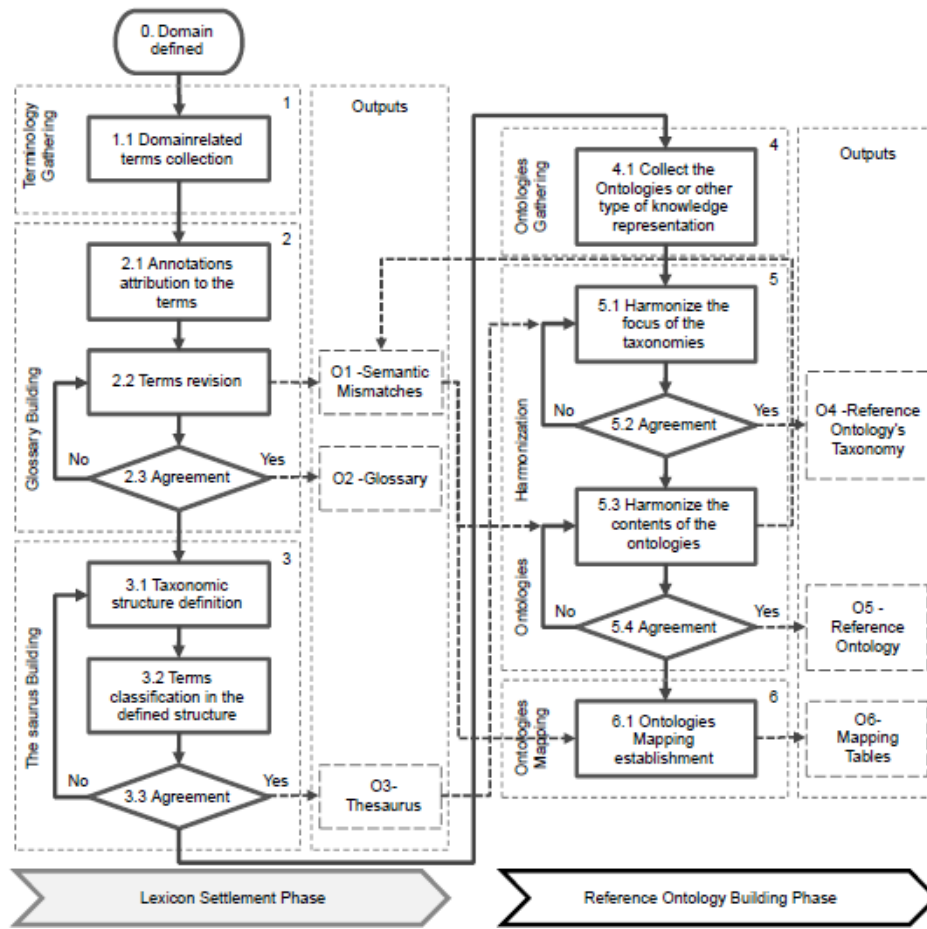


Figure 3.2 - MENTOR Methodology [49]

Some work has already been conducted by Gaspar in [50] in order to enrich MENTOR with qualitative information collective methods and developed a functioning prototype, illustrated in Figure 3.3 that implements some of the described steps.

MENTOR

Username Password

New User?

MENTOR login page

Bolt_Suppliers Terms Revisions Panel

Finished Revision?

Term Manager	
Term	Description
<input type="radio"/> Bolt	Headed fasteners having external threads that meet an exacting, uniform bolt thread specification (such as M, MJ, UN, UNR, and UNI) such that they can accept a no tapered nut
<input type="radio"/> Tolerance	Permissible limits of variation that a measure can fall within, determined by the inspection phase after manufacture of the component
<input type="radio"/> Pitch	The axial distance between a point on a thread flank and the equivalent point on the immediately adjacent and corresponding flank
<input type="radio"/> s	Dimension across flats in a hexagonal head

Terms revision (Glossary Building step)

Figure 3.3 - MENTOR prototype [50]

3.2.1. Mediator Ontology

As previously referred, one of the steps in this methodology comprises the establishment of mappings to record the possible existing semantic mismatches. Since this is not an easy task, MENTOR uses a Mediator Ontology (MO) as a reference for mediating the mapping establishment and its subsequent ‘mapping records’ reasoning [50]. This allows communities to build systems with reasoning capabilities able to understand each other’s representation format, without having to change their data and communication functions [49]. Apart from the feature of enabling seamless communication between different systems, the MO is also able to represent ontology semantic operations such as, the semantic mismatches found in the Glossary building step, the semantic transformations identified in the harmonization process, the ontologies mapping and other ontology operations (e.g. versioning) [49]. To be able to represent these ontology operations, the MO is uses a five-tuple mapping expression proposed by Agostinho et al. in [51]. According to the tuple philosophy, all the information about the mappings should be stored in a dedicated KB so that it becomes computer processable and so that readjustments are easier to manage. In this case the KB is the MO which is defined in the OWL format with the structure represented in Figure 3.4.

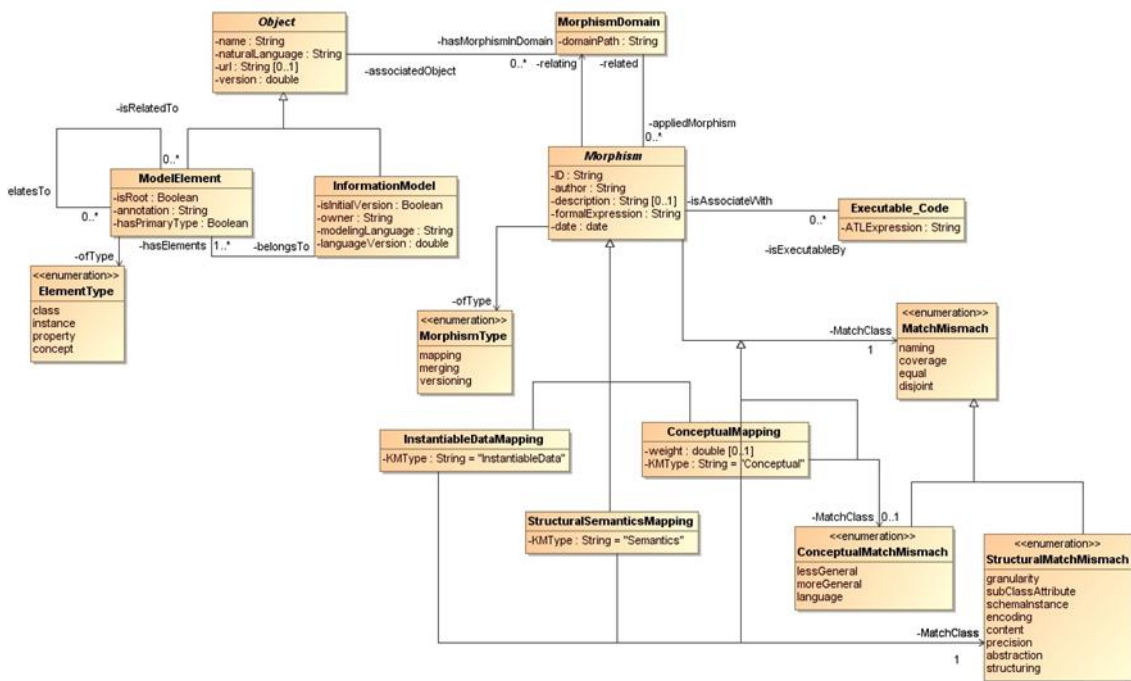


Figure 3.4 - Mediator Ontology Structure [52]

The structure of the MO, presented in the previous figure is described as follows: the MO has two main classes: “Object” and “Morphism”. The “Object” represents any “InformationModel” (IM) which is the model/ontology itself and “ModelElements” (also belonging to the IM) that can either be classes, properties or instances. The “Morphism” associates a pair of “Objects” (related and relating), and classifies their relationship with a “MorphismType”, “KnowledgeMappingType” (if the morphism is a mapping), and “Match/Mismatch” class. The “Morphism” is also prepared to store transformation oriented “ExecutableCode” that will be written in the ATLAS Transformation Language and can be

used by several organizations to automatically transform and exchange data with their business partners [51].

With the mappings stored in the mediator, all information regarding them can be accessed by local systems of business partners that wish to communicate. The translation from one message format to another is the responsibility of the mediator, therefore assuring seamless communication between different systems. Figure 3.5 illustrates the general vision of the flow of the system. At the beginning, all the required mappings, using the tuples, are established and stored in the MO. Then, when one of the business partners wants to communicate with another, it simply sends its message to mediator who is then in charge of transforming its format and forwarding it to the destination.

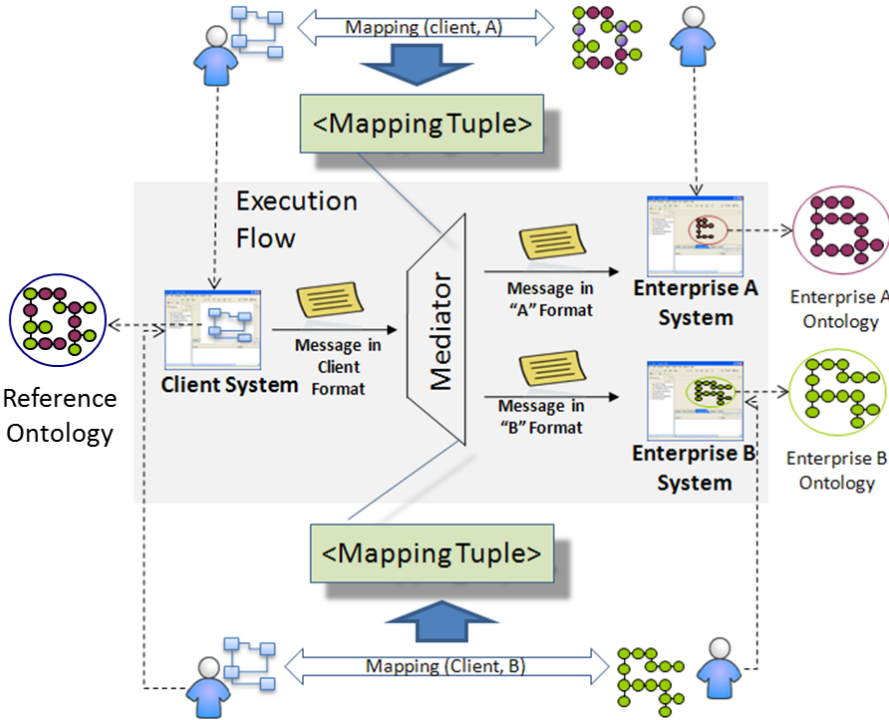


Figure 3.5 - Mapping design and execution flow in data exchange

3.3. Consistency Checking

Consistency is defined in the Oxford dictionary [54] as the quality of achieving a level of performance which does not vary greatly in quality. This can be interpreted as something that has an accordance with previously stated facts or characteristics. That being said, the consistency of an ontology can be defined as incorporating new information in accordance to the one that was previously represented in the ontology. Therefore, consistency checking is one of the most important phases in ontology maintenance. As ontologies evolve, i.e., modifications in the application domain, incorporating additional functionality according to changes in the users' needs, organizing information in a better way, etc. [55] it is important to have a mechanism that can validate that the information within the ontology remains consistent. Much work has been done in this field, such as, frameworks that provide

strategies for detecting and repairing inconsistencies [56] and how to deal with the evolution of ontologies in order to maintain their consistency [55]. Other work that has been conducted in this area features tools to help prevent or detect and fix inconsistencies. Such tools are mostly descriptive logic reasoning tools that infer logical consequences, through an inference engine, based on a set of rules or facts. Examples of consistency checking tools are ConsVISor [57], FaCT++ [36] or HerMiT [34].

Consistency checking can be divided into two categories that are referred here as interoperability checking and semantic checking. The latter being the main focus of this dissertation.

3.3.1. Interoperability Checking

As information systems in companies and enterprises evolve and grow larger and more complex, a previous interoperable state with other systems, within the same or between different companies, can become compromised. Therefore there is a necessity to continuously verify if the systems are still functioning properly with one another, i.e., if they remain interoperable. This is often done by using tests designed specifically to achieve this goal. From a general perspective, two types of testing are relevant in the entrepreneurial context, conformance and interoperability testing [58]. Conformance testing involves the verification of whether an implementation is in conformity with the underlying specifications. This kind of testing is the first step toward interoperability with other conformant systems as prescribed by the specification [58]. An example of conformance testing is shown in Figure 3.6.

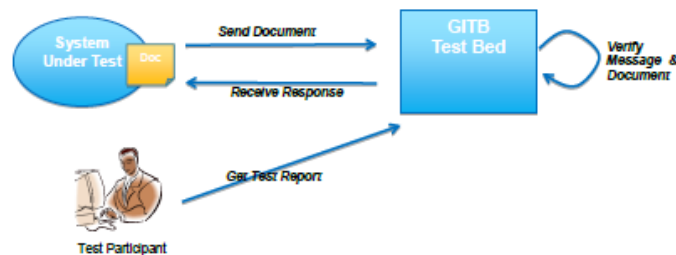


Figure 3.6 - Conformance Testing Example [58]

On the other hand, interoperability testing consists in verifying if the involved systems are actually able to intercommunicate based on some exchange scenarios, as seen in Figure 3.7. However, this form of testing is generally more difficult to automate than the previous one and requires more human involvement and coordination [58]. Furthermore, human involvement is highly costly and leaves room for human error due to the repetitive nature of the tests and the high number of interfaces involved in the testing of complex systems [59].

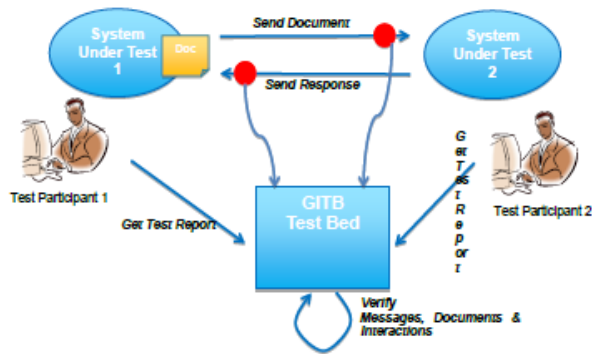


Figure 3.7 - Interoperability Testing Example [58]

Also according to [58], software implementations can be certified and correct information exchange between systems if both types of testing are used, meaning that conformance testing isn't a substitute for interoperability and vice-versa. Furthermore, the quality of the interoperability specifications impacts the difficulty in the application of the tests.

3.3.2. Semantic Checking

Semantic checking refers to the validation of ontological concepts regarding their semantics. This is a very important step if one is to have interoperability between several ontologies. According to Li et al. in [46], there are three types of semantic checking, single, composite and multiple. In the first case the semantic checking is done within a single ontology and it is only deemed consistent if it satisfies a set of concepts and axioms and if all used entities is defined. The second type refers to the semantic checking of ontologies (or subsets of ontologies) within ontologies. Also, in this case an ontology is deemed consistent if the ontology itself and all its included ontologies are consistent. Finally the third type is the main focus of this work and depicts a scenario where several separate ontologies interact with each other. In this case of multiple semantic checking the goal is to validate if all knowledge represented in a given ontology can be represented in another (within the same domain), by means of a reference ontology, for example. Conceptual representations of each of these types can be seen in Figure 3.8.

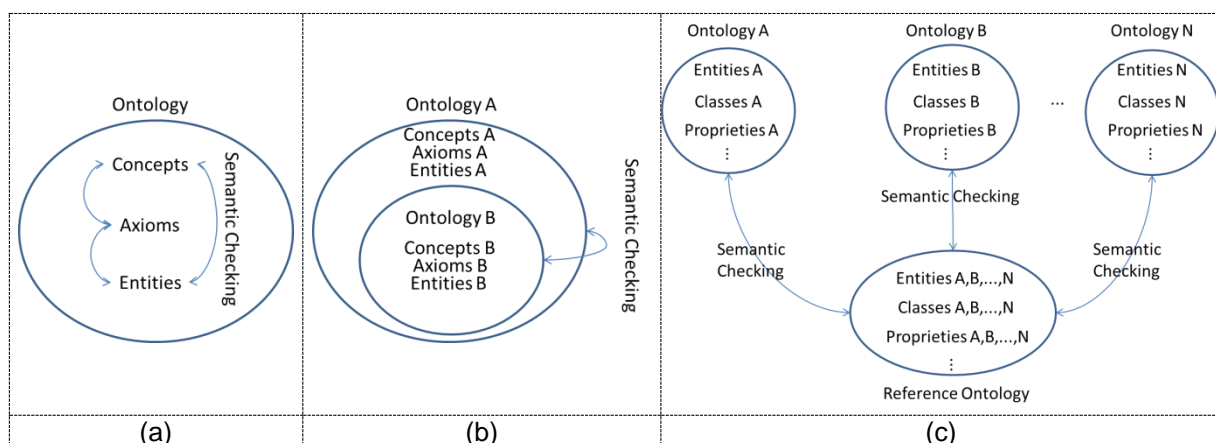


Figure 3.8 - (a) Single Semantic Checking; (b) Composite Semantic Checking; (c) Multiple Semantic Checking

In this work, semantic checking will be accomplished by using a reasoning process aided by rules defined in Semantic Web Rule Language (SWRL).

Haase et al. in [55] further propose three types of consistency regarding the semantics of a single ontology. It refers to structural, logical and user-defined consistency. Structural consistency considers constraints that are defined for the ontology model with respect to the constructs that are allowed to form the elements of the ontology [55], which means that an ontology is only deemed structurally consistent if no elements of the ontology violate its defined structure. For example, consider an ontology that represents a simple bank domain, where there are employees, clients and accounts and that there is a constraint that doesn't allow an employee to be both employee and client. If the bank manager tries to open an account for himself, thus becoming both client and employee, then the ontology would become structurally inconsistent. Logical consistency focuses on whether the ontology does not contain any contradicting information, i.e. it is semantically correct [55]. For an ontology to be logically consistent it must satisfy each of its axioms. Considering the previous example, if an axiom stating that there is a client named John and assuming that an employee named John already exists, then the addition of this axiom would lead to a logically inconsistent ontology because it was previously defined that employees cannot be both clients and employees. Finally, user-defined consistency takes into account specific user requirements that are external to the ontology itself. Even if an ontology is structurally and logically consistent it may still violate user requirements [55]. Two types of user-defined consistency were identified, generic and domain dependent. The former refers to consistency conditions applicable across domains. The latter refers to consistency conditions that take into account the semantics of a particular formalism of the domain [55].

3.3.3. Semantic Adaptability Using a Mapping Tuple

Either being used in the form of traditional databases, architectural models, or domain ontologies, models can be described on multiple formats, languages, expressiveness levels, and for different purposes. A model can be characterized according to four dimensions: *Metamodel* - the modelling primitives of the language for modelling (e.g. ER, OWL, XSD) are represented by a set of labels defined in the metamodel; *Structure* - corresponding to the topology associated to the model schema; *Terminology* - the labels of the model elements that don't refer to modelling primitives; *Semantics* - given a "Universe of Discourse", the interpretations that can be associated with the model [51]. In this case the information models are ontologies where mappings are established to relate each element of the source ontology to a corresponding element in the target one. However, a formalism able to represent these mappings is needed because it could facilitate the integration and use of various knowledge sources to the semantics adaptability of the information systems [53]. To ensure semantic interoperability and minimize inconsistencies, Agostinho et al. in [51] proposed a tuple based mapping scheme. They used a 5-tuple mapping expression to formalize morphisms between model elements enriched with semantic information that enables fast human readability. This mapping tuple expression contains 5 fields, ID, MElems, KMType, MatchClass and Exp. The ID is the unique identifier of the mapping tuple. The MElems field indicates the pair of mapped elements. KMType is the knowledge

mapping type which can be Structural Semantic, Instantiable Data or Conceptual as stated in previous section and illustrated in Figure 3.9. The MatchClass field stands for the semantic mismatch classification which depends on the knowledge mapping type. Finally the Exp field is the mapping expression that translates and further specifies the previous tuple components.

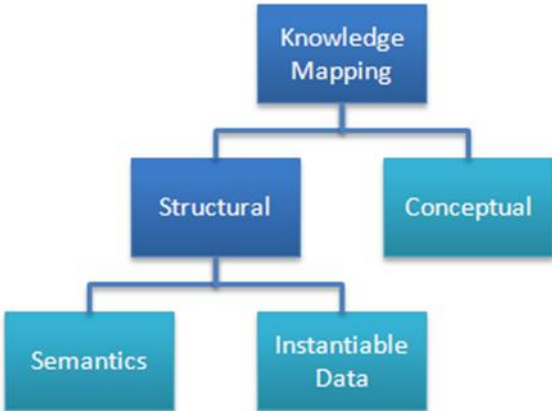


Figure 3.9 - Knowledge Mapping Types [51]

Although the mappings are made to minimize inconsistencies, imperfect mappings can lead to such inconsistencies called semantic mismatches. These mismatches have been identified in [51] as lossy, when losses of information are recorded and as lossless when no information loss is recorded. A summary of the identified semantic mismatches can be seen in Table 3.1

Table 3.1 - Semantic Mismatches [52]

Mismatch		Description	Examples
Lossless	Naming	Different labels for same concept of structure	
	Granularity	Same information decomposed in or composed by (sub)attributes	
	Structuring	Different design structures for the same information	
	SubClass-Attribute	An attribute, with a predefined value set represented by a subclass hierarchy (or vice-versa)	
	Schema-Instance	An attribute value in one model can be a part of the other's model schema (or vice-versa)	
	Encoding	Different formats of data or units of measure	

Mismatch		Description	Examples
Lossy	Content	Different content denoted by the same concept	
	Coverage	Absence of information	
	Precision	Accuracy of information	
	Abstraction	Level of specialisation	

These mismatches are often observed when mapping operations between ontologies are executed. Therefore, this can be associated with the MENTOR methodology approach to the semantic alignment of the involved ontologies. Thus the MO, which uses these tuple based mappings to represent ontology semantic operations and records any mismatches that occur during the operations.

3.4. Semantic Checking Framework

There are three approaches to the issue of semantic checking, the one suggested by Li et al. in [46], the one proposed by Haase et al. in [55], and the one by Agostinho et al. in [51].

Starting with the approach described in [46], it features a more general method to the semantic checking issue, since the ontologies are considered as a whole. This means that only the architectural aspects of the ontology based system are considered, i.e., if the system is composed of a single ontology, or if there are multiple separate ontologies interacting each other. This has led the author to adopt this method to serve as basis for the scenarios identified in the framework.

Referring to the approaches to the semantic checking issue by Haase et al. and Agostinho et al. these seem quite similar at first sight. However in [55] the approach is more of a structural point of view, encompassing the semantics and data instances of the ontologies. On the other hand, the method described in [51] is more specific, since besides considering the structural aspects of ontologies, namely its semantics and data instances, it also considers the conceptual aspect of ontologies. This conceptual aspect is about the meanings of the used terms, i.e., if the concepts are well characterized. Due to the specificity in this approach, the author chose to use the knowledge mapping types seen in Figure 3.9, applied to the scenarios presented by Li et al. in [46], illustrated in Figure 3.8, to build the framework.

To help maintain semantic interoperability in the enrolled systems, the author proposes a semantic checking framework (Table 3.2), which shows the main characteristics that an ontology based information system should comply to maintain semantic consistency.

Table 3.2 - Semantic Checking Framework

	Single Ontology	Composite Ontologies	Multiple Ontologies
Structural	1. Automatic reasoning	3. Automatic reasoning; Automatic synchronization	5. Ad hoc synchronization; Automatic reasoning
Conceptual	2. Human action plus automatic reasoning	4. Human action plus automatic reasoning; Automatic synchronization	6. Human action plus automatic reasoning; Ad hoc synchronization

More specifically, this framework intends to evaluate, in each case, if the information models are consistent according to their structural and its conceptual definition. Technically, each of these cases can be verified by resorting to description logic reasoners by using inference engines. These reasoners derive logical consequences from a set of pre-defined rules which aim to represent the semantic mappings between the elements of the information models. However in some cases, further mechanisms are needed to verify the semantic consistency of the system.

This framework is composed of 6 items. Framework items 1 and 2 refer to scenarios where only a single ontology is involved. For item 1 (single ontology – structural consistency checking), a simple reasoning process suffices to verify the structural consistency of the ontology. This process was named automatic because it is only needed to execute a typical reasoner on the ontology and it automatically infers that the ontology is structurally consistent. This can be done because descriptive logic reasoning tools infer specific logical consequences, through an inference engine, based on a set of rules or facts. Regarding item 2 of the framework, besides an automatic reasoning process similar to the previous situation, human action is also needed. This is because the user needs to create elements of the concepts to test if after running the reasoner such concepts are well positioned in the ontology, thus verifying their conceptual definitions.

Items 3 and 4 of the framework denote cases where composite ontologies are involved. On item 3, in addition to an automatic reasoning process, an automatic synchronization mechanism is also required. Since composite ontologies are composed of two or more ontologies merged together, a synchronization mechanism is needed to validate its structural consistency. This is because any structural change that occurs in one of the ontologies needs to be reflected in all the other KREs. On the other hand, item 4 additionally requires human interaction to the automatic reasoning and synchronization processes. This is because the user needs to create elements of the concepts represented in the ontology to verify its conceptual definitions, achieving the same objective mentioned for item 2. Moreover in this case, the concepts need to be well represented in the merged ontology to avoid repetitions and that is why the synchronization and reasoning are both required.

Finally, items 5 and 6 of the framework are applicable in scenarios where multiple but separate

ontologies are involved. In item 5, besides having an automatic reasoning process, it also requires an *ad hoc* synchronization process in order to align the knowledge represented in the various KREs. This means that any changes that occur in a given element of the system must be reflected in the others in order to maintain consistency. Since these types of systems can be very complex, knowing the synchronization method facilitates the semantic checking process. This is because the users need to know what the system is prepared for, i.e., its capabilities in order to execute the modifications on one side to be properly reflected in the other. If the user doesn't have a grasp of the system is prepared for then it could lead to misalignment of the represented knowledge which could lead back to a non-interoperable state. In entry 6 it is needed human intervention, for the same reasons that figure in the other conceptual checking cases. The user needs to create elements that intend to represent certain concepts, and these elements must be well represented in the other ontologies that compose the system. To accomplish this, a reasoner is executed as in the other conceptual checking items. Here the synchronization process is also used for aligning the knowledge represented in the various KREs.

3.5. Concluding Remarks

In this chapter the proposed semantic checking framework was presented. Its goal is to provide effective means to check if the data exchanged between enterprises information systems is facilitated and its understanding maintained. To that effect, generic guidelines are proposed for each case so that they can be applied to any system to assure semantic consistency of the exchanged data.

In conclusion of this chapter, this framework can be a valuable advantage in terms of verifying and maintaining the semantic consistency if the involved systems.

4. APPLICATION SCENARIOS

In this section two scenarios are presented that intend to demonstrate the applicability of the proposed framework. Firstly, a mechanical scenario is introduced, where a relation between a bolt supplier and manufacturer is illustrated. The second scenario refers to the ENSEMBLE project and intends to further demonstrate the applicability of some of the framework guidelines.

Table 4.1 indulges the cases that are being considered in these scenarios. This table has the same structure of the framework. However, its cells contain the scenarios that were identified as being better suited to a specific framework item. For items 1, 2 and 6 of the framework, the Mechanical Scenario presented in section 4.1 was used to validate and demonstrate them. On the other hand, for items 3, 4 and 5 of the framework the ENSEMBLE project scenario, presented in section 4.2, was used to validate and demonstrate these items. Furthermore, in chapter 5 a synchronization tool prototype is described and in chapter 6, framework item 5 is thoroughly demonstrated through use case examples of that same tool.

Table 4.1 – Framework applicability scenarios

	Single Ontology	Composite Ontologies	Multiple Ontologies
Structural Semantic	1.Mechanical Scenario	3.ENSEMBLE Scenario	5.ENSEMBLE Scenario
Conceptual	2.Mechanical Scenario	4.ENSEMBLE Scenario	6.Mechanical Scenario

4.1. Mechanical Scenario

This scenario depicts a relation between a bolt retailer and manufacturer. Each enterprise has its own ontology with its own representation of the domain. To be able to collaborate with one another it was decided to follow the MENTOR methodology in order to build a reference ontology to serve as a mediator to their interactions. Thus, this scenario main goal is to check the consistency of the ontologies, after applying MENTOR, regarding their semantics.

Protégé 4.1 was chosen as the ontology management tool, instead of Ontopia or TM4L, through this scenario due to its user friendly interface and the built-in reasoner plugins to conduct the semantic checking. Regarding the reasoning process, the Hermit reasoner was chosen to verify the consistency of the ontologies in scenarios 4.1.3 and 4.1.5. While the Pellet reasoner was chosen to perform the semantic checking in scenario 4.1.1. Some rules were also defined, in the SWRL language, to aid in the reasoning process. Figure 4.1 illustrates an overview of this scenario.

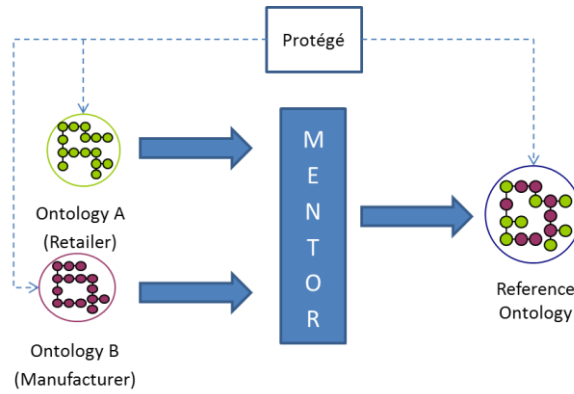


Figure 4.1 - MENTOR scenario overview

For this purpose it was used the MENTOR methodology, which comprehends a series of steps, one of them being the glossary building phase, where the domain terms and definitions are gathered. In this case, the definitions adopted by each of the implemented ontologies (retailer, manufacturer and reference) are presented in Table 4.2, Table 4.3 and Table 4.4, respectively, and are based on the ones by Sarraipa et al. in [49].

Table 4.2 - Retailer Ontology Terms and Definitions

Ontology	Term	Definition	Category
Retailer	Bolt	Headed fasteners having external threads that meet an exacting, uniform bolt thread specification (such as M, MJ, UN, UNR and UNJ) such that they can accept a no tapered nut.	Class
	Nominal Diameter	The diameter of an imaginary cylindrical surface tangent to the crests of an external and (or) to the roots of an internal thread.	Class
	Maximum Diameter	The maximum value acceptable for the diameter obtained from a predefined allowed upper deviation of the nominal diameter.	Class
	Minimum Diameter	The minimum value acceptable for the diameter obtained from a predefined allowed lower deviation of the nominal diameter	Class

Note that during the harmonization phase the maximum and minimum diameter concepts were obtained based on equations [i] and [ii] that use the upper and lower tolerance proprieties.

$$\text{upper tolerance} + \text{nominal diameter} = \text{maximum diameter} \quad [i]$$

$$\text{lower tolerance} + \text{nominal diameter} = \text{minimum diameter} \quad [ii]$$

Table 4.3 - Manufacturer Ontology Terms and Definitions

Ontology	Term	Definition	Category
Manufacturer	Bolt	Term used for a threaded fastener, with a head, designed to be used in conjunction with a nut.	Class
	Nominal Diameter	Diameter of an imaginary cylinder parallel with the crests of the thread; in other words it is the distance from crest to crest for an external thread, or root to root for an internal thread.	Class
	Tolerance	Allowable deviation from a nominal or specified dimension, determining maximum and minimum material condition.	Class

After gathering the terms and definitions from both entities, the reference ones were established as seen in Table 4.4.

Table 4.4 - Reference Ontology Terms and Definitions

Ontology	Term	Definition	Category
Reference	Bolt	Headed fasteners having external threads that meet an exacting, uniform bolt thread specification (e.g. M, MJ, UN, UNR, UNJ) such that they can accept a no tapered nut.	Class
	Nominal Diameter	In a hexagonal bolt's head, is the dimension of the nominal diameter tangent to the flats (also expressed as the dimension across flats which correspond to the size of wrench to use). The diameter of an imaginary cylindrical surface tangent to the crests of an external and (or) to the roots of an internal thread.	Class
	Upper Tolerance	Maximum value of allowable deviation from a nominal or specified dimension.	Class
	Lower Tolerance	Minimum value of allowable deviation from a nominal or specified dimension.	Class

Note that the reference ontology distinguishes between “Upper and Lower Tolerances” while the manufacturer ontology does not. Also it doesn’t define the “Maximum and Minimum Diameters” as in the retailer ontology because these can easily be obtained from the “Nominal Diameter and Upper and Lower Tolerances” as specified in the previous equations.

Upon obtaining the reference ontology the next step is to try and accomplish the previously established goal for this scenario. This means that is needed to validate if the reference ontology indeed represents the knowledge gathered from the enterprises and if this representation is able to do so without any loss of information. The ontologies used to verify this scenario are represented in Figure 4.2.

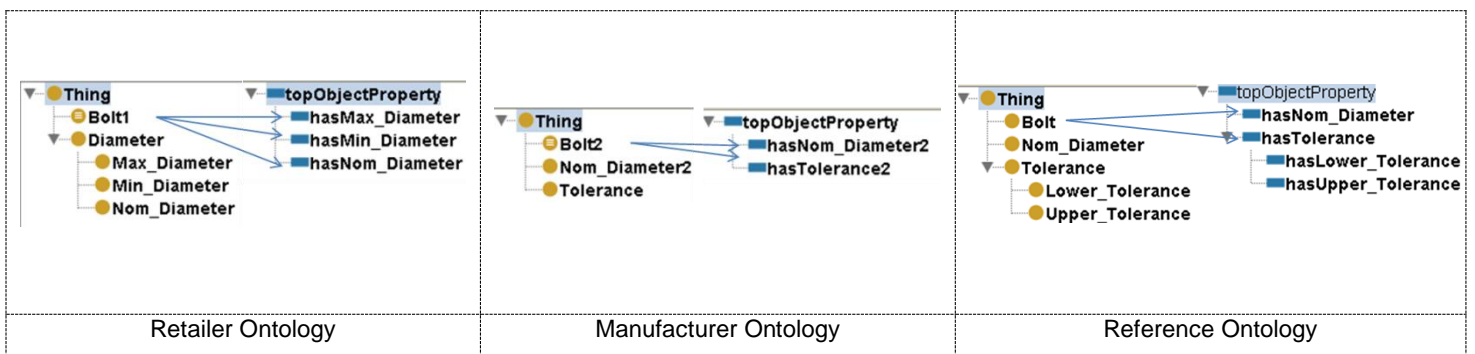


Figure 4.2 - Used Ontologies

As referred in Table 4.1, this scenario is used to validate some of the framework items, namely items 1, 2 and 6, and to that effect specific examples are presented for each case.

4.1.1. Single Structural Semantic Checking

This scenario intends to demonstrate the applicability of the proposed framework regarding its item 1. As previously indicated, this case only requires an automatic reasoning process in order to verify the

structural consistency of a single ontology. To this effect, the retailer ontology, shown on the left part of Figure 4.2, was used to validate this case. The ontology was then submitted to the reasoning process, using the Pellet reasoner [35], and the structural consistency of the ontology was confirmed, as shown in Figure 4.3. As stated previously, Pellet was the chosen reasoner, instead of the others presented in section 2.4, to perform this task due to the simplicity of its use as a command line interface and of its output.

```
Pellet is an OWL ontology reasoner.  
For more information, see http://clarkparsia.com/pellet  
C:\Users\Gonçalo\Desktop\pellet-2.3.0>pellet consistency D:\Faculdade\Tese\Work\testOnto\single.owl  
Consistent: Yes
```

Figure 4.3 - Pellet reasoner output

4.1.2. Single Structural Semantic Checking Concluding Remarks

As indicated by the framework in item 1, the structural consistency of a single ontology was verified by resorting to an automatic inference mechanism. In this case the chosen ontology was submitted to the Pellet reasoner and its output was an assertion to whether the ontology was consistent or not, which, in this case, its consistency was effectively verified.

4.1.3. Single Conceptual Checking at MENTOR Scenario

This situation refers to item 2 of the proposed framework. In this case, the chosen ontology was also the one from the retailer enterprise. However, as stated earlier, the chosen reasoning tool was HermiT [34] as a plugin in Protégé due to its effectiveness and simplicity. The basis for this example is the creation of instances in the 'Thing' class, to ensure that the instances aren't initially associated with any class. Then a reasoning process is started to verify if the instances are placed in their corresponding classes, in order to validate its conceptual definition.

As seen in Figure 4.4 (left), instances ('b1', 'maxD', 'minD', 'n') were defined as being in the 'Thing' class. It is also shown the structural properties that comprise instance 'b1' and the expressions that define the bolt concept. These properties indicate that a bolt instance must be comprised of a minimum diameter, a nominal diameter and a maximum diameter. The class expressions define a criterion that an instance must meet in order to belong in that class. It is based on these expressions and properties that the reasoning process is able to infer the correct consequences. The creation of the instances had to be done manually as it was suggested by the framework. Afterwards the reasoning process was executed and the output is shown in the right part of Figure 4.4. As it can be seen, the Bolt class is highlighted and it shows the 'b1' instance as an inferred member of that class, thus validating the bolt concept for this ontology. Although the output only highlights the instance that refers to the bolt class ('b1'), the other instances ('maxD', 'minD' and 'n') were also inferred to their proper classes thus validating the conceptual consistency of the ontology.

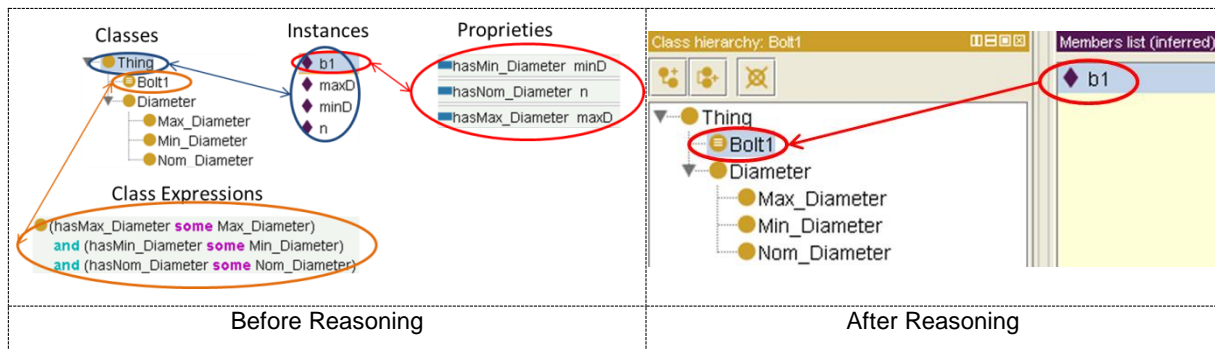


Figure 4.4 - Reasoning Example (Retailer Ontology)

4.1.4. Single Conceptual Semantic Checking Concluding Remarks

Based on inference mechanisms, more specifically using the HerMiT reasoner and some human intervention, it is possible to assess the conceptual consistency of this ontology, as indicated in item 2 of the proposed framework. As seen in the example HerMiT was able to successfully infer the created instances to their corresponding classes. Therefore it is possible to conclude that this ontology is conceptually consistent.

4.1.5. Multiple Conceptual Semantic Checking

This example features the case of conceptual validation of multiple ontologies, item 6 of the framework, namely between the retailer and reference ontologies and between the manufacturer and reference ontologies. To portray the relations between the retailer, manufacturer and reference, tuple-based mappings were defined between their concepts. Table 4.5 and Table 4.6 show the mappings between the retailer and reference, and between the manufacturer and reference, respectively.

As an example as to how this mappings are built, consider the bolt definitions adopted by the manufacturer and reference entities. Firstly an ID is attributed to serve as a unique identifier to that mapping. Then the two terms are compared, where 'a' is the manufacturer definition of the bolt concept and 'b' the one defined by the reference. These two terms are then classified according to their knowledge mapping type shown in Figure 3.9. In this case they have been identified as belonging to the "Conceptual" knowledge type. Then the two definitions of the bolt concept are compared and classified according to the semantic mismatches presented in Table 3.1. In this case, by resorting to the bolt definitions presented in Table 4.3 and Table 4.4 it is easily verified that the reference definition is more complete and as such, the MatchClass was defined as less general, because it's the manufacturers term in relation to the reference term. Finally, the expression is defined according to the MatchClass, using set theory symbols. In this case the manufacturers' term is contained in the reference one.

Table 4.5 – Retailer Reference Mappings

ID	Retailer1_1		Retailer2_2	Retailer2.3_3	Retailer2.1_4	Retailer2.2_5
Melem s = (a,b)	a	Retailer. Bolt1	Retailer.Diamet er	Retailer.Nom_D iameter	Retailer.Max_Dia meter	Retailer.Min_Dia meter
	b	Referenc e.Bolt	Reference.Nom_ Diameter	Reference.Nom_ Diameter	Reference.Upper _Tolerance	Reference.Lower _Tolerance
KMTyp e	Conceptual		Conceptual	Conceptual	Conceptual	Conceptual
Match Class	Equal		More General	Less General	More General	More General
Exp	$b = a$		$b \supseteq a$	$b \subseteq a$	$b \supseteq a$	$b \supseteq a$

Table 4.6 - Manufacturer - Reference Mappings

ID	Manufacturer1 _1		Manufacturer2_2	Manufacturer 1.3_3	Manufacturer3_2	Manufacturer2.3 _3
MElem s = (a,b)	a	Manufactur er.Bolt2	Manufacturer.No m_Diameter2	Manufacturer. Tolerance	Manufacturer.Tol erance	Manufacturer.Tol erance
	b	Reference. Bolt	Reference.Nom_ Diameter	Reference.Tol erance	Reference.Uppe r_Tolerance	Reference.Lowe r_Tolerance
KMTy pe	Conceptual		Conceptual	Conceptual	Conceptual	Conceptual
Match Class	Less General		Less General	Equal	More General	More General
Exp	$b \subseteq a$		$b \subseteq a$	$b = a$	$b \supseteq a$	$b \supseteq a$

With such mappings defined, it is very important to verify if the reference ontology indeed represents the knowledge gathered from the enterprises, and if any information model compliant with the reference ontology knowledge, is able to exchange data between the participant enterprises, without any loss of information independently of the direction that the data is transmitted to.

After obtaining the mappings, a reasoning approach to check if the concepts are well represented in the ontologies and aligned to all the participants' knowledge. In this case the process starts by pairing one of the enterprise ontologies with the reference one in the same KB. Then instances were created in the "Thing" class. These instances were created there to ensure that the reasoning process puts them in their corresponding classes. The example shown in Figure 4.5 refers to the retailer and reference ontologies.



Figure 4.5 - Reasoning Example (Retailer and Reference Ontologies)

As observed in Figure 4.5, two different types of 'Bolt' instances (i.e. "b" and "b1") were created and upon running the HerMiT reasoner it was observed that both instances were indeed placed in the 'Bolt' class of the retailer and reference ontologies (i.e. "Bolt" and "Bolt1"). Therefore it can be concluded that the ontologies remained consistent and a bolt represented in the retailer ontology is semantically equivalent to a bolt represented in the reference ontology.

The next example is shown in Figure 4.6 denotes the manufacturer and reference ontologies. The principle of this example is the same as in the one before, meaning that two different types of 'Bolt' instances ("b" and "b2") were created within the 'Thing' class and then the reasoning process was executed to verify if the instances were placed in their proper classes.

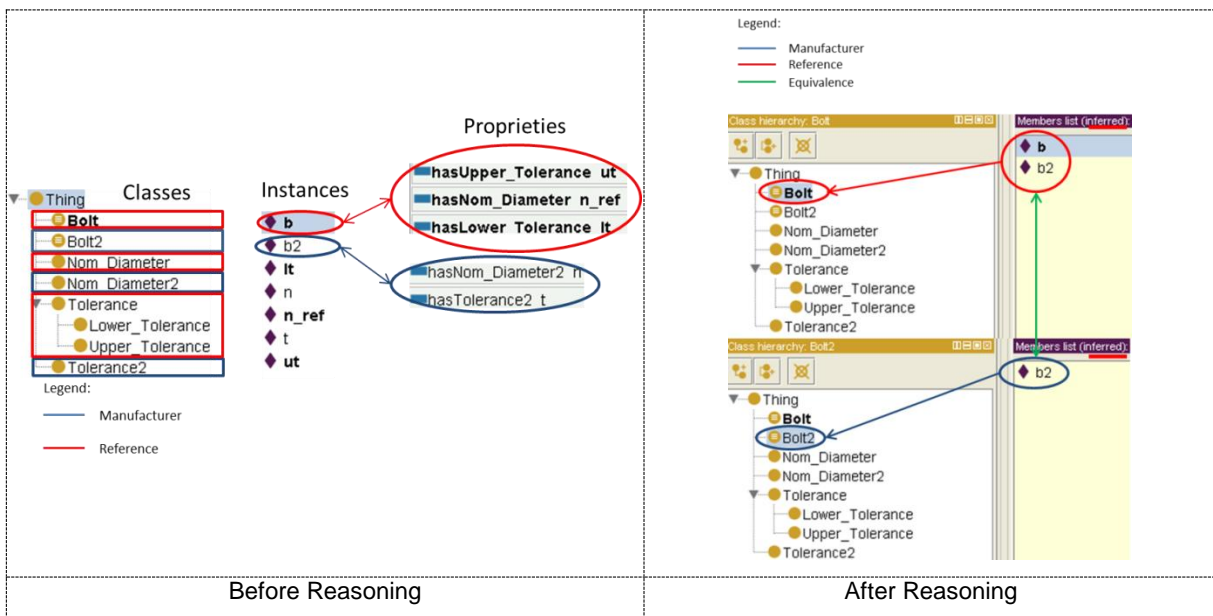


Figure 4.6 - Reasoning Example (Manufacturer and Reference Ontologies)

Contrarily to the previous example, in this case, it is possible to observe some loss of information because although both instances (“b” and “b2”) are represented within the reference ontology, the same cannot be said regarding the manufacturers’ ontology since only “b2” is represented. This is because of the “Tolerance” definitions represented by each of the ontologies. While the reference ontology distinguishes between “Upper and Lower Tolerances”, the manufacturers only define a single tolerance, assuming an equal value for “Upper” and “Lower”. This means that if different values for the “Upper and Lower Tolerances” are defined in the reference ontology then a conflict is created. Since the manufacturer ontology does not have such distinction and therefore doesn’t know which value is the correct one, leading to possible inconsistencies in the ontology. This loss of information is easily reflected in the mappings defined in the direction from the reference to the manufacturer, that are the same as the ones in Table 4.6 with the addition of the ones shown in Table 4.7.

Table 4.7 - Reference - Manufacturer Conceptual Mappings

ID	Reference3.1_1		Manufacturer3.2_2
MElems = (a,b)	a	Reference.Tolerance.Lower_Tolerance	Reference.Tolerance.Upper_Tolerance
	b	Manufacturer.Tolerance	Manufacturer.Tolerance
KMType	Conceptual		Conceptual
MatchClass	Abstraction		Abstraction
Exp	b = a		b = a

It is also worthy of remark that to aid in the reasoning process some rules were defined in SWRL. These rules serve the purpose of aiding the inference engine by providing it with additional facts and logical consequences that are based on the mappings defined earlier. Table 4.8 and Table 4.9 illustrate the rules defined in the first example and second examples, respectively and their purpose.

Table 4.8 - SWRL rules defined in the retailer - reference example

Rule	Purpose
Min_Diameter(?minD), Lower_Tolerance(?lt), Nom_Diameter(?n), Thing(?b), hasMin_Diameter(?b, ?minD), hasNom_Diameter(?b, ?n) -> hasLower_Tolerance(?b, ?lt)	If a bolt instance is defined as having a minimum diameter and a nominal diameter then it can be concluded that it also has a lower tolerance.
Max_Diameter(?maxD), Nom_Diameter(?n), Upper_Tolerance(?ut), Thing(?b), hasMax_Diameter(?b, ?maxD), hasNom_Diameter(?b, ?n) -> hasUpper_Tolerance(?b, ?ut)	If a bolt instance is defined as having a maximum diameter and a nominal diameter then it can be concluded that it also has an upper tolerance.
Max_Diameter(?maxD), Nom_Diameter(?n), Upper_Tolerance(?ut), Thing(?b), hasNom_Diameter(?b, ?n), hasUpper_Tolerance(?b, ?ut) -> hasMax_Diameter(?b, ?maxD)	If a bolt instance is defined as having a nominal diameter and an upper tolerance then it can be concluded that it also has a maximum diameter.
Min_Diameter(?minD), Lower_Tolerance(?lt), Nom_Diameter(?n), Thing(?b), hasLower_Tolerance(?b, ?lt), hasNom_Diameter(?b, ?n) -> hasMin_Diameter(?b, ?minD)	If a bolt instance is defined as having a nominal diameter and a lower tolerance then it can be concluded that it also has a minimum diameter.

The rules in the Table 4.8 explore the diameter and tolerance proprieties of the ontologies and proved to be invaluable to validate the semantic consistency of the ontologies. It is quite simple to conceive that bolts can have slight deviations regarding their diameters, so by defining a nominal diameter and

upper and lower tolerances it is easy to conclude that the bolt has maximum and minimum diameters. The contrary is also true, if a nominal diameter for a bolt is defined as a certain value and the end product records a slight deviation either by excess or default then it is easy to conclude that the bolt has upper and lower tolerances.

Table 4.9 - SWRL rules defined in the manufacturer - reference example

Rule	Purpose
Tolerance2(?t2), Lower_Tolerance(?lt), Thing(?b), hasTolerance2(?b, ?t2) -> hasLower_Tolerance(?b, ?lt)	If a bolt instance is defined has having a tolerance then it can be concluded that it also has a lower tolerance.
Tolerance2(?t2), Upper_Tolerance(?ut), Thing(?b), hasTolerance2(?b, ?t2) -> hasUpper_Tolerance(?b, ?ut)	If a bolt instance is defined has having a tolerance then it can be concluded that it also has an upper tolerance.

These rules in Table 4.9 exploit the tolerance definitions of the manufacturer and reference ontologies. In this case it is assumed that if a bolt is defined has having a tolerance it can be concluded that it has both the same upper and lower tolerances. However, unlike the previous example, the contrary is not true, since the bolt can have different upper and lower tolerances it is not possible to conclude that it has a single tolerance. As a consequence this can lead to inconsistencies as it was explained beforehand.

4.1.5.1. Multiple Conceptual Semantic Checking Demonstration Example

To better illustrate this semantic checking case, a practical example where a client orders a bolt product with particular specifications is described. As seen in Figure 4.7, the client specified a bolt with a nominal diameter of ‘10’ and upper and lower tolerances of ‘0.2’ and ‘0.1’, respectively. A message containing these specifications is then sent from the client system to the mediator in the reference ontology format. The mediator then translates the message from the reference format, to both the retailer and manufacturers before relaying it to them. Converting from the reference to the retailer format is fairly straightforward. Based on the previously presented mappings in Table 4.5, the mediator only has to sum the nominal diameter and the upper tolerance to obtain the maximum diameter, subtract the lower tolerance to the nominal diameter to obtain the minimum diameter and the nominal diameter is the same for both. However the case isn’t so simple when translating from the reference to the manufacturer format. While the nominal diameter remains the same for both formats, the manufacturer, doesn’t distinguish between upper and lower tolerances. Thus the mediator has to assume one of its values, either upper or lower tolerance (it’s up to the system developer to choose which one), as the tolerance in the manufacturer format. If the values for upper and tolerances happen to be equal, then there is no problem whatsoever, since it won’t have any adverse effect on the final product. On the other hand, if the values are different, as depicted in the example, then there will be loss information thus leading to inconsistencies, since the same bolt product is not equally represented in all formats.

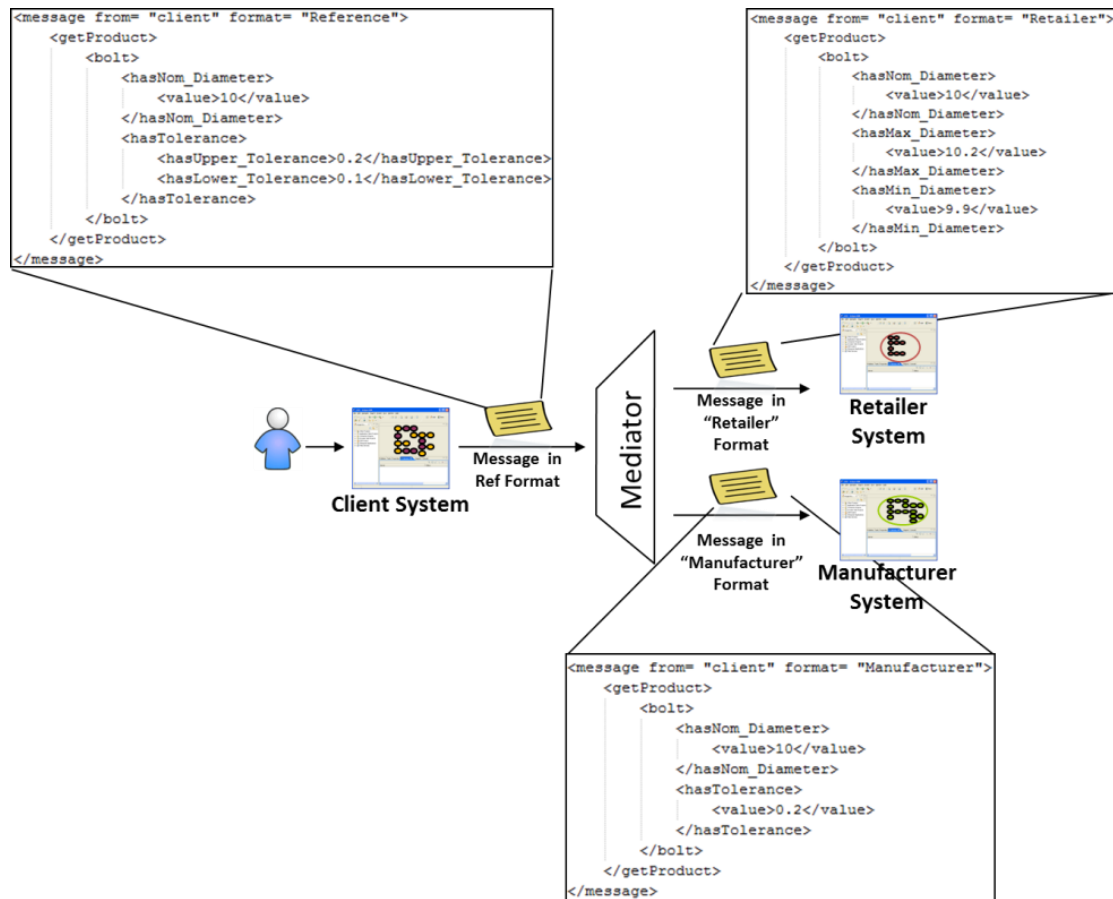


Figure 4.7 - Multiple Conceptual Semantic Checking Example

4.1.6. Multiple Conceptual Semantic Checking Concluding Remarks

To sum up this scenario, Table 4.10 illustrates the information exchange between ontologies and whether this exchange resulted in a loss of information.

Table 4.10 - Identification of conceptual losses in information

Ontologies Information Exchange (From – To)	Information Loss
Retailer – Reference	No
Reference – Retailer	No
Manufacturer - Reference	No
Reference - Manufacturer	Yes

As seen in Table 4.10 there was loss of information only in one case, from the reference to the manufacturer ontology. This means that the conceptual checking has failed in this case, since not all the knowledge represented in the reference ontology can be reproduced in the manufacturer ontology. As previously explained, this has to do with the tolerance definitions adopted by both entities. This loss was recorded from the reference from the manufacturer, what was to be expected when the mappings in this direction had a match class of Abstraction, which is a lossy semantic mismatch. On the other hand, no loss of information was recorded in the opposite direction, i.e. from the manufacturer to the reference. This is due to the fact that the tolerance concept of the manufacturer ontology is more

general than the upper and lower tolerance concepts of the reference ontology, thus the information can be 'split' evenly between the reference concepts. For example if the tolerance is defined as being 0,5 centimetres then the reference assumes the same value for both the upper and lower tolerances.

Regarding the retailer and reference ontologies, no information losses were recorded in both directions since the concepts defined in each one are quite similar to one another.

The previous conclusions can be reinforced further by analysing the practical example featuring an interaction between a client and a bolt retailer and manufacturer. In the example it can be observed that in fact there is loss of information between the reference (client) and the manufacturers' messages, specifically in the tolerance values interpreted by each one. Contrarily, no information loss was recorded from the client to the retailer. It is important to have semantic checking in this case, because it needs to be ensured that the product delivered to the client is in fact what was ordered in the first place. Therefore the data exchanged between the various entities must remain consistent to comply to all of the clients specifications.

4.2. ENSEMBLE Scenario

The work described in this section refers to the ENSEMBLE project. Its goal is to gather and provide knowledge in the EI and neighbouring domains, such as papers and publications, authors, domain experts, etc.

The application scenario that supports this work is depicted in Figure 4.8. Its aim is to provide a visual understanding of the architecture of the system, that is, how the system is structured by representing the most important components, how they are connected and what technologies were used to develop them. Furthermore it also depicts that will be developed in the future, such as the harmonization of the ontologies and its synchronization with the FInES wiki.

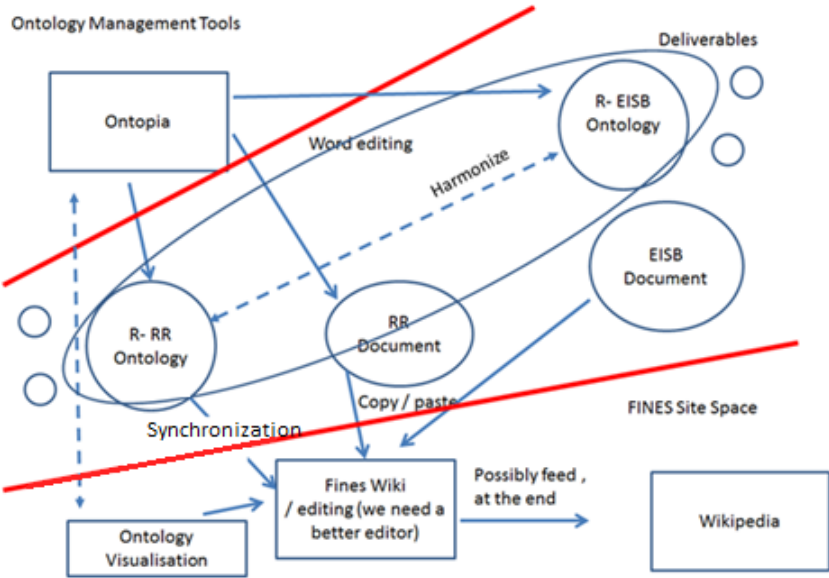


Figure 4.8 - ENSEMBLE scenario overview

As seen in Figure 4.8, the ontology management tool with which this work was developed is Ontopia, and apart from developing the ontologies, Ontopia is also used to visualize them, using the Vizigator tool. Regarding the R-RR (Reference – Research Roadmap) ontology and EISB ontology, they were obtained by combining several smaller ontologies using a reference ontology building methodology such as MENTOR. One of the goals of this application scenario consists in harmonizing these two ontologies into a single reference ontology for the whole EI community (see section 4.2.1). The FInES wiki functions as a source of knowledge and as a means of integrating all the knowledge gathered in the aforementioned ontologies, so these components need to be tightly synchronized (refer to section 4.2.2) as to avoid inconsistencies in the information.

In sum:

- **Ontopia** – Ontology management tool selected to develop the R-RR and R-EISB ontologies and to visualize them;
- **RR/EISB Documents** – Project deliverables;
- **R-EISB Ontology** – The need to have an advanced EISB service that is able to provide specific knowledge with several interrelationships led to the development of a KB ruled by a reference ontology. Therefore the EISB Reference Ontology, shown in part in Figure 4.9, main goal is to represent all the knowledge related to the EISB domain. Having this kind of knowledge would facilitate the search of specific information, for instance papers or methods of a determined EISB area or a specific set of tutorials related to a specific EISB topic, or even a set of expert researchers [61]. Another aim of this ontology is to serve as a facilitator for knowledge reasoning, enabling different views of the information either gathered from the wiki or directly from an administrator. [61]. Furthermore the EISB reference ontology can prove to be a valuable asset for the science base itself gathering meta-information relevant to both EI and the neighbouring domains [61].

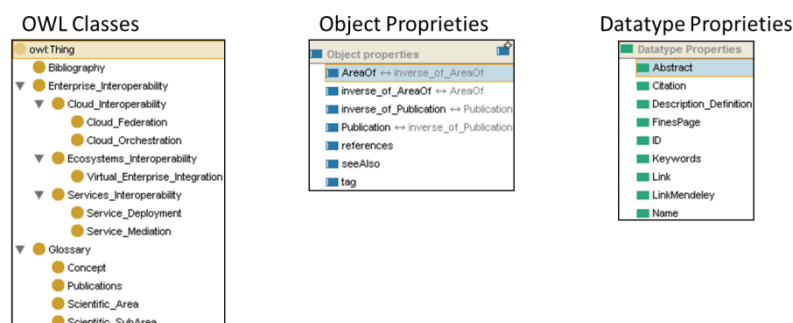
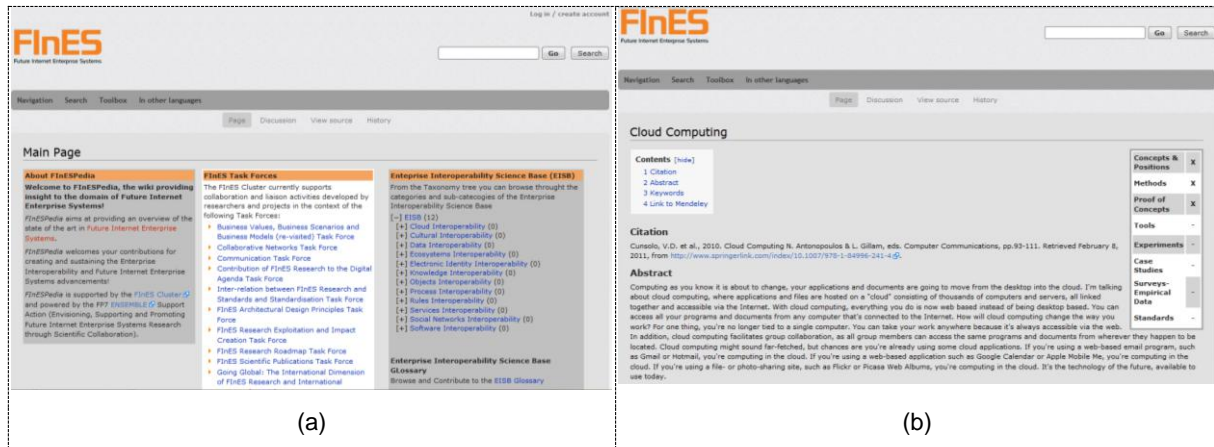


Figure 4.9 - EISB Reference Ontology

- **R-RR Ontology** – Ontology containing the knowledge gathered by the research roadmap team;
- **Ontology Visualization** – Using Ontopia’s Vizigator tool (see Figure 2.8);
- **Fines Wiki** – Source of knowledge more focused on the collaborative gathering of information

from domain experts. The wiki, depicted in Figure 4.10, is accessible through the FInES cluster portal [23], serves as tool to maintain all the EI state of the art research. In order to avoid replication of efforts it will be synchronized automatically with the reference ontology [61].



Despite this scenario being presented in its entirety in Figure 4.8, the work conducted in relation to this, focuses only in two of its aspects: 1) the harmonization (merging) process between the R-RR and R-EISB ontologies; and the synchronization process between the harmonized ontologies and the FInES wiki.

The first aspect, which is also described in subsection 4.2.1, regards to composite ontologies characteristics of the proposed framework for semantic checking, i.e., items 3 and 4 of it. This is because the result of merging ontologies is a KB constituted by composite ontologies.

The second aspect, which is also described in subsection 4.2.2, regards to the multiple structural semantic checking, i.e., item 5 of the proposed framework, since the harmonized ontologies and the wiki can be seen as separate KREs. Furthermore, in relation to this, chapter 5 presents a synchronization process prototype that is then semantically demonstrated using real examples in chapter 6.

4.2.1. Composite Ontologies Checking at ENSEMBLE Scenario

This scenario consists in harmonizing two ontologies namely, the EISB Reference Ontology and the EI Roadmap Ontology, in order to form a composite ontology. Therefore this scenario can be applied to both framework items 3 and 4.

Harmonizing the EISB Reference Ontology with the EI Roadmap Ontology

As seen in Figure 4.8, there is a step in which the harmonization of the EI roadmap ontology with the EISB reference ontology occurs. The goal of this harmonization is to have a single reference ontology to serve the ENSEMBLE project.

The harmonization can be achieved using any of the operations described in section 2.1, and the

impact of using each one is analysed. If the mapping operation is used, the source ontologies (EI roadmap and EISB reference ontologies) wouldn't suffer alterations. However, as the ontologies evolve (contents are updated, added, removed, etc.), new mappings between them would have to be made and consequently this would require constant supervision to ensure that there are no inconsistencies. Using this approach would also make the synchronization with the EISB (FInES) wiki extremely difficult because a three way synchronization would be required, i.e. between the EISB wiki and each of the ontologies and between the ontologies themselves. The alignment operation could alter the source ontologies in order to make them aligned and coherent with each other. However, since these ontologies aim to be complementary of each other this process would be essentially equal to the mapping operation, meaning that the previously described difficulties would remain. Finally, the merging operation could be used to simply integrate the ontologies with each other, where the output would be a single reference ontology. This process could be achieved using a methodology like MENTOR or by simply integrating the contents of one of the ontologies into the other. This method achieves the initial goal to have a single reference ontology. Furthermore, this process would make the synchronization process less difficult due to the existence of only one ontology to synchronize with the EISB wiki. However after the merging is complete the result should be thoroughly tested in order to avoid inconsistencies and losses of information. These tests should focus mainly in the structure and concepts of the resulting ontology. Therefore this scenario is a suitable candidate to validate framework items 3 and 4.

However, in the point of view of this dissertation, this scenario was merely identified as belonging to items 3 and 4 of the framework. Consequently, work in terms of validating or demonstrating this scenario isn't conducted in this dissertation and is considered as a possibility for future work.

4.2.2. Multiple Structural Semantic Checking at ENSEMBLE Scenario

Up to this point, this work has focused mainly in the validation of the consistency between multiple ontologies. However this scenario describes the validating of the semantic structure between the harmonized ontology of the previous step, and the EISB wiki, therefore relating this scenario to item 5 of the framework. Since these two entities, on the surface, seem to be quite different it is important for them to have a similar structure, as seen in Figure 4.11 and therefore the importance of the structural semantic checking step. Moreover, these two entities need to be tightly synchronized in order for the information to remain consistent.

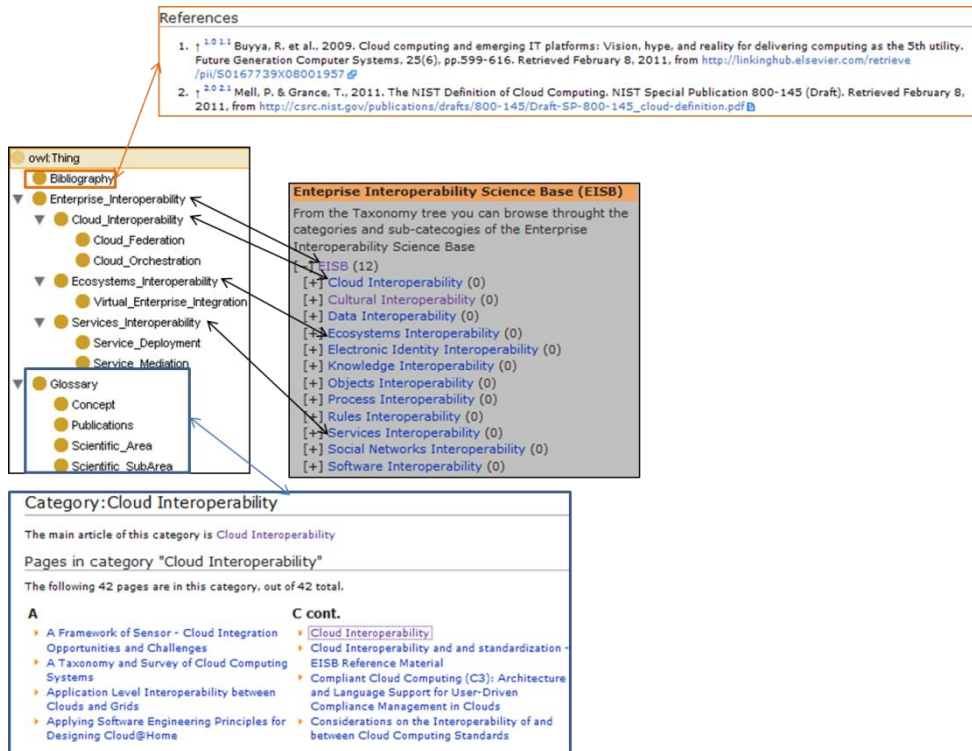


Figure 4.11 - EISB Reference Ontology and FInES Wiki Structural Comparison

Synchronization with the EISB Wiki

Since the EISB reference ontology and the EISB wiki are constantly evolving, any changes that occur on one side need to be reflected on the other. Therefore a method for synchronizing the EISB wiki and EISB reference ontology must be developed. In this dissertation it was defined and implemented a synchronization process based on the two possible solutions presented in Figure 4.12 that are discussed afterwards. However, as suggested by the guidelines of framework item 5, the effort here would be to understand the functionalities of the synchronization process, but not implement it.

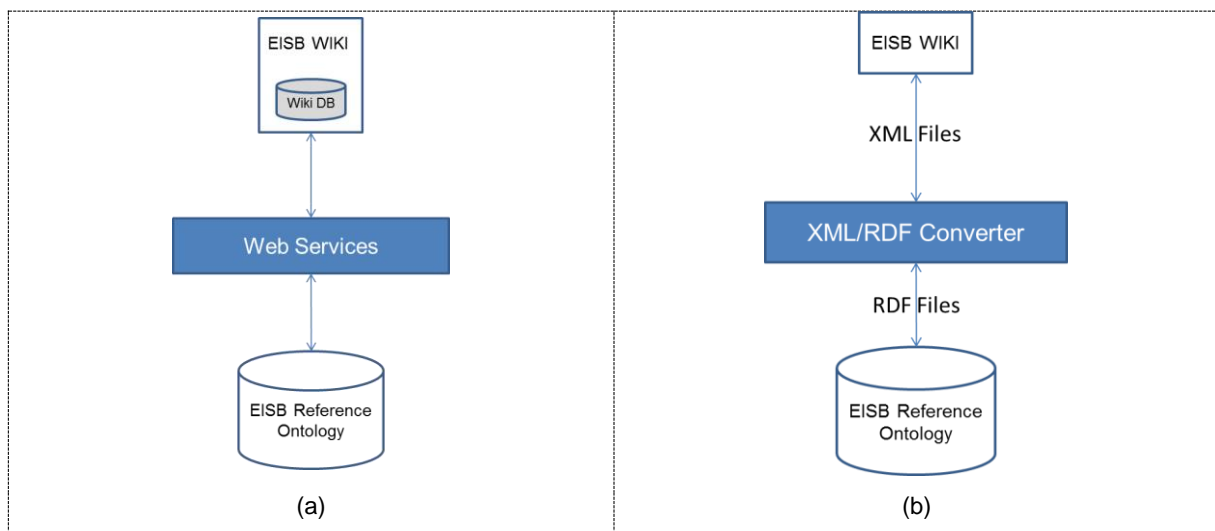


Figure 4.12 - Ontology/Wiki Synchronization (a) Using Web Services; (b) Using XML/RDF Files

As seen in Figure 4.12, (a) features web services as a possible solution to the synchronization issue, while (b) relies on XML/RDF files to solve the problem.

In solution (a) the idea is to have a web services layer that is able to connect both to the EISB wiki DB and the EISB reference ontology. Upon connection the web service would then retrieve the desired content through queries, either to the ontology or to the wiki DB. Then the retrieved content would simply be transported from the source to the destination, thus maintaining the contents harmonized in both ends.

In solution (b) the idea is to have files transfer from end to end. For instance, a system administrator would export the desired content from the wiki onto an XML file and convert it to the RDF format, with the help from a XML/RDF converter application. Then that converted file would be imported to the ontology, therefore updating the ontology with new information. The opposite operation is also possible, that is, exporting the desired content of the ontology to an RDF file and converting it to the XML format so that it can be imported to the wiki. Operations such as this are often referred as bulk load operations. A bulk load operation, in this case, would be an exportation of all the contents from one end (wiki or ontology) and import them into the other. However this solution is, at most, semi-automatic because it needs human intervention at the importing and exporting stages of the process. Furthermore the mappings required to convert from XML to RDF and vice-versa can be very complex.

Maintenance Strategy

For an efficient maintenance strategy to this project, one could look at both solutions presented in the previous section and state that they somewhat complement each other. Since solution (a) is highly dynamic, due to the features that web services provide, it is more suitable for scenarios when the changes, either on the ontology or the wiki are small. On the other hand, solution (b) is a better fit for bulk load operations. Concluding, one could apply both cases for a more efficient and complete solution to the synchronization issue. Solution (a) would then be applied in cases of small incremental changes and solution (b) in scenarios that would require large portions (or all) of data to be synchronized to either end.

4.2.3. ENSEMBLE Scenario Concluding Remarks

In this subsection, a scenario was presented that suits three items of the proposed framework. The harmonization process used to achieve a reference ontology suits items 3 and 4 of the framework, while the synchronization with the FInES wiki encompasses item 5.

The study of the presented scenario served an important purpose, since difficulties associated with the addressed items of the framework were identified and possible solutions were presented. Regarding the semantic checking of composite ontologies possible methods to accomplish harmonization were addressed along with their associated difficulties. Regarding the semantic checking of the structure of multiple KREs, it was identified the need of having a synchronization process, therefore its inclusion

as a possible scenario for item 5 of the framework, and two possible solutions were presented and discussed. Furthermore to facilitate the synchronization of the reference ontology with the FInES wiki, it is extremely important to verify, as the system evolves, if their structure remains consistent to ensure that the information represented on one side can be equally and accurately represented in the other.

5. PROOF-OF-CONCEPT IMPLEMENTATION

The objective here is to implement a proof-of-concept to validate the proposed framework, namely item 5, and to that effect, the previously presented scenario of section 4.2.2 was chosen. The solution presented here is related to the ad-hoc synchronization step of the fifth item of the proposed framework. It was chosen to implement a synchronization process to show that it is possible to effectively maintain consistent data between two different KREs.

This chapter is structured as follows; firstly the chosen technologies to implement the synchronization process are presented, followed by the architecture and description of its components, which has the objective of providing a general understanding of how the synchronization process is structured and how it is implemented. Finally, two sequence diagrams will be presented and analysed that show the flow of execution of the developed synchronization prototype.

5.1. Used Technologies

Before starting the development of the synchronization tool, a study of the required technologies was made. The result of this study is presented in the next subsections of this document which shows the chosen technologies for this project and their descriptions.

5.1.1. Java

The Java programming language is a general-purpose concurrent class-based object-oriented programming language, specifically designed to have as few implementation dependencies as possible [62]. This is a highly flexible language since it can run in any platform. This is possible because Java software is compiled into specific bytecode that is run on the Java Virtual Machine (JVM) instead of being compiled into platform-specific machine code.

The main reason the synchronization module was chosen to be developed in the java programming language was due to the fact that Protégé provides the previously presented API that allows the developer to manage an ontology programmatically. Java was also chosen due to its runtime performance and the fact that it is an open source software.

5.1.2. MySQL

MySQL is a widely popular open source DB software [63]. It is a DB management system that uses the SQL language (Structured Query Language) to perform operations on relational databases. This technology can also be embedded into others, allows the developer to build DB applications in their language of choice [64]. It can be embedded in the Java language via the JDBC (Java Database Connectivity) driver.

5.1.3. Protégé / Protégé-OWL API

This technology was already extensively presented in section 2.2.1 of this document and therefore it won't be re discussed here. However it is important to say that Protégé was chosen as the ontology management tool, instead of the other tools studied in section 2.2, due to the fact that Protégé provides a free API to manage ontologies programmatically.

The Protégé-OWL API is an open source Java library for the OWL language and RDF(s). It provides methods and classes that allow the developer to create or edit OWL data models, such as ontologies. It is possible to query and manipulate data within the model, for example, creating or deleting classes, properties and instances [65]. This API can be used to develop components that are executed in the Protégé user interface or it can be used to develop stand-alone applications, such as the prototype that was developed during the course of this dissertation.

5.1.4. Changes and Annotations API

This API enables tracking changes, annotating ontology components or changes and access to that information programmatically. The change tracking information annotation of ontology entities and changes is stored as instances of the changes and annotation ontology (ChAO), called the ChAO KB.[66]

5.2. Architecture

The architecture designed for the synchronization tool is an enhancement of the one previously presented in Figure 4.12 however the principle remains the same. The web services layer was dropped because the developed tool connects directly to the wiki DB via the JDBC driver and connects directly to the ontology using the Protégé-OWL API via its URL. A general overview of the synchronization tools architecture and the interaction between the different elements is shown in Figure 5.1.

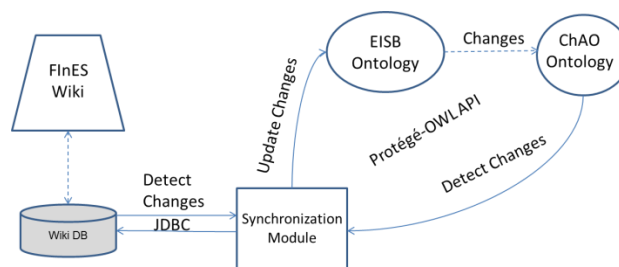


Figure 5.1 - Synchronization tool architecture

This architecture is composed of 5 main components:

- The FInES Wiki where the knowledge of the EI community is gathered;
- The FInES Wiki DB that contains all the contents of the FInES wiki and means of detecting

any changes that may occur;

- The EISB Ontology that also contains the knowledge of the EI community;
- The Changes and Annotations Ontology (ChAO) that contains the records of all the changes that took place in the EISB ontology;
- The Synchronization Module serves as a user interface to the whole synchronization process.

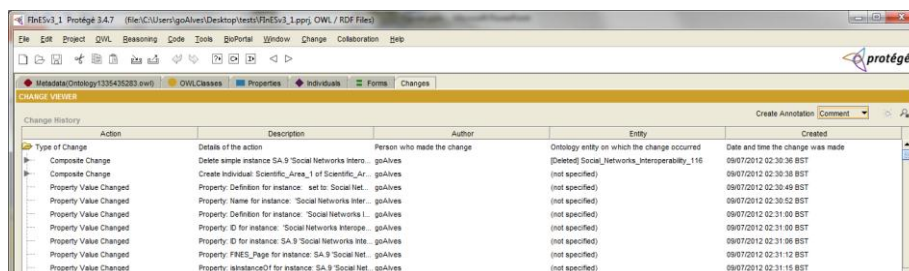
With this simple architecture users can easily synchronize wiki contents with the EISB ontology and vice versa. The java application uses the wiki DB to detect any changes that have occurred in the wiki since it was last run and then updates the ontology accordingly. On the reverse path, the java application uses the ChAO ontology to detect any changes that have occurred in the EISB ontology and then updates the wiki accordingly by placing the new contents directly into its DB.

5.2.1. Synchronization Module

The developed module is composed of 4 java classes. A class (“GUI.java”) that implements the user interface and performs the required initializations. Another developed class implements methods that support the interaction between the synchronization tool and the wiki DB (“Database.java”). Finally there are two more classes that serve the purpose of managing the actual synchronization between the ontology and the wiki, and between the wiki and the ontology, respectively (“Wiki2Onto.java” and “Onto2Wiki.java”).

5.2.2. ChAO Ontology

The ChAO ontology allows the tool to detect any changes that have occurred in the EISB ontology and what exactly those changes were. The synchronization tool connects to the ontology via its location (URL, file path, etc...) and updates it directly by saving the ontology into a new file and overwriting the old one. Figure 5.2 shows an example of changes recorded in the ChAO ontology using a Protégé interface.



Action	Description	Author	Entity	Created
Type of Change	Details of the action	Person who made the change	Ontology entity on which the change occurred	Date and time the change was made
Composite Change	Delete simple instance SA.9 'Social Networks Inter...	goAlves	(Deleted) Social_Networks_Interoperability_115	09/07/2012 02:30:36 BST
Composite Change	Create individual: Scientific_Area_1 of Scientific_Ar...	goAlves	(not specified)	09/07/2012 02:30:38 BST
Property Value Changed	Property: Definition for instance: set to: Social Net...	goAlves	(not specified)	09/07/2012 02:30:49 BST
Property Value Changed	Property: Name for instance: 'Social Networks Inter...	goAlves	(not specified)	09/07/2012 02:30:52 BST
Property Value Changed	Property: Definition for instance: 'Social Networks I...	goAlves	(not specified)	09/07/2012 02:31:00 BST
Property Value Changed	Property: ID for instance: 'Social Networks Interop...	goAlves	(not specified)	09/07/2012 02:31:00 BST
Property Value Changed	Property: ID for instance: SA.9 'Social Networks Inte...	goAlves	(not specified)	09/07/2012 02:31:06 BST
Property Value Changed	Property: FINES_Page for instance: SA.9 'Social Net...	goAlves	(not specified)	09/07/2012 02:31:12 BST
Property Value Changed	Property: instanceOf for instance: SA.9 'Social Net...	goAlves	(not specified)	09/07/2012 02:31:15 BST

Figure 5.2 - Example of changes recorded in the ChAO ontology

5.2.3. Wiki DB

The FInES wiki will be extensively described in the next subsection and therefore won't be further discussed here. However the FInES wiki DB is very important to the project, because like the ChAO

ontology, it is what allows the synchronization tool uses to detect any changes that have occurred in the wiki via the “recentchanges” table. The developed tool connects to the wiki DB via its URL and updates its contents directly into specific tables of the wikis DB. Figure 5.3 shows an example of the wiki DB represented in the “phpMyAdmin” interface. It features the “page” table highlighted and shows some of its instances.

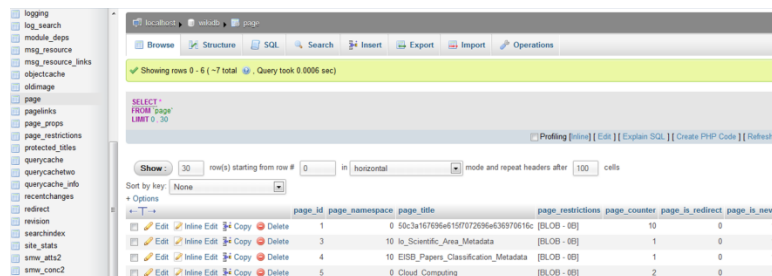


Figure 5.3 - Wiki DB example

5.2.4. FInES Wiki

As said beforehand, the FInES wiki serves as a source of knowledge more focused on the collaborative gathering of information from domain experts. It also serves as tool to maintain all the EI state of the art research. To that effect, the wiki, in its homepage is divided into 3 main parts, as seen in Figure 5.4, the FInES Research Roadmap, the FInES Task Forces and the EISB. However only the latter is relevant for this work and therefore is the only that will be described in detail.

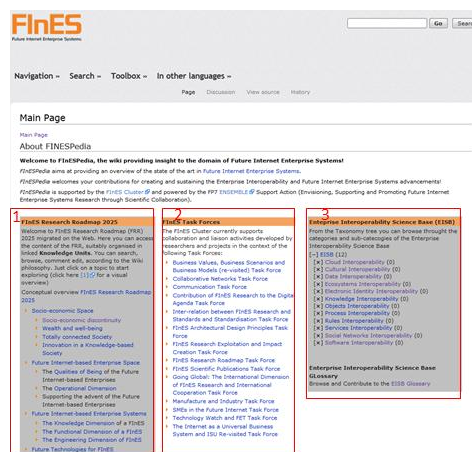


Figure 5.4 - FInES Wiki Homepage: 1 - FInES Reserach Roadmap; 2 - FInES Task Forces; 3 – EISB

Looking now, in detail, into the EISB portion of the wikis homepage, it can be seen in Figure 5.5 that it is composed of several links that represent and direct the user to the various scientific areas addressed by the EI community as well as the EISB Glossary.

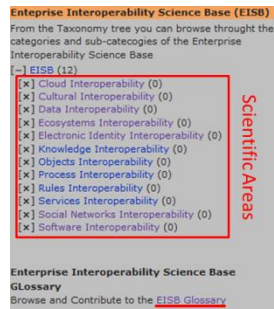


Figure 5.5 - FlNES Wiki: EISB Scientific Areas and Glossary

Going now into further detail, the EISB portion of the FlNES wiki is essentially composed of 5 types of pages, the category page type, the scientific area and sub scientific area description type, the EI ingredients page type and the publications page type.

- Category Pages** – These types of pages serve as an index since it lists all of the wiki pages that fall under a specific category. The links present in the wikis homepage direct the user into these pages that can either be the EISB glossary or a specific scientific area. In the EISB Glossary category page all the terms in the EI domain are listed. These terms are called the EI ingredients and they can be scientific areas, sub scientific areas, case studies, methods, experiments, tools, standards, a proof of concept, surveys or empirical data and concepts or positions. Regarding the scientific area category pages, these are very similar to the EISB Glossary page, however they contain a list of the EI ingredients and publications that particular scientific area addresses as well as the wiki page describing that same scientific area. A part of the EISB Glossary page and an example of a scientific area category page is shown in Figure 5.6.

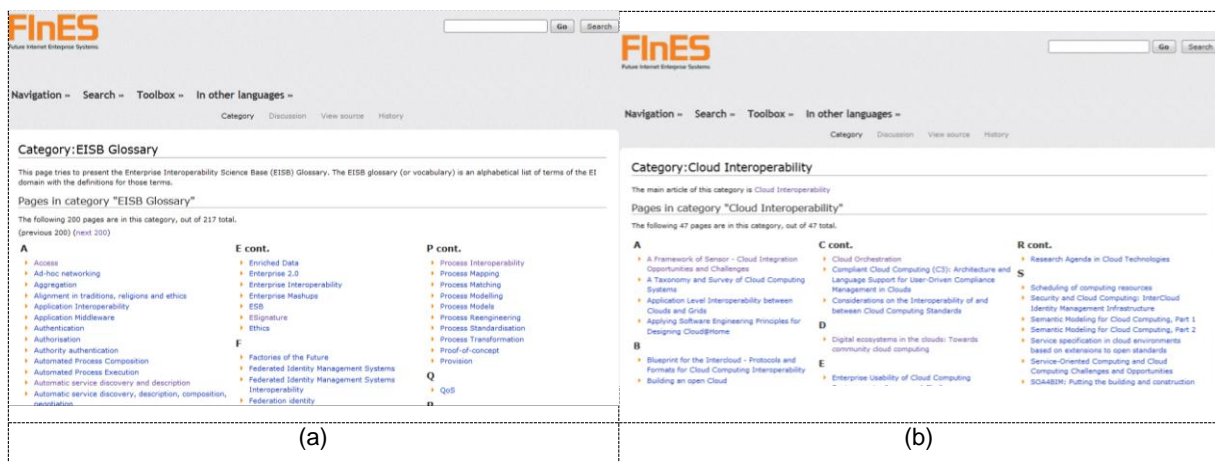


Figure 5.6 - FlNES Wiki: (a) EISB Glossary; (b) Scientific Area category page example

- Scientific Area Pages** – These types of pages have the purpose of describing the scientific areas addressed by the EI community. Each page contains a table that summarizes the scientific area. This table contains the name of the scientific area paired with its unique identifier, a small description, links to other scientific areas, a list of its sub scientific areas and a list of tags. Furthermore these types of pages contain the full general description of the

scientific area which is the main focus of the page, along with a section with the references that are identified along the text and a section that contains links to additional information relative to that scientific area. A scientific area page example is presented in Figure 5.7.

Cultural Interoperability												
<p>With the world evolving into a unified marketplace, the business context in terms of societal and company culture, language and various regional particularities prove to be a great challenge regarding the communication between organisations as well as between their underlying ICT systems [1]. For instance, even though English is used as a common, universal language for conducting business, not all employees are fluent English speakers, which results in the misuse of the language and to problematic communication [2]. In many cases documents and other electronic (or not) data resources need to be translated to English in order to be sent over to another organization. As a result, a great amount of effort and resources are allocated during this procedure, and transaction times are increased due to these transformations that have to be undertaken manually. Moreover, it is not uncommon that there is not a 1-1 data translation mapping and many literals have no equivalent terms in the foreign language.</p> <p>Global trading represents a good example of a domain in which cross-cultural information systems between enterprises are really required. Experts in the global trading domain are under a growing pressure to exchange actual and correct information on very local and unique regions. Different regions in the world share many business processes and data although each region still is unique in terms of different processes, data and business rules due to religion, cultural and social customs [3]. This makes it, on the one hand, difficult to generalize B2B solutions and present them centrally, but on the other hand it is also clear that knowledge about aspects of each situation can be shared. Due to continuous global changes, cultural aspects in local regions change rapidly and ICT experts likewise need to acquire up to date information about the local cultural status on a frequent basis.</p> <p>Interoperability between organisations, people and enterprise systems that have different languages and different cultural aspects such as politics, religion, regional art, Traditions and Social Customs defines the concept of Cultural Interoperability [4]. Cultural Interoperability is the degree to which knowledge and information is anchored to a unified model of meaning across cultures [5]. Enterprise systems that take into consideration Cultural Interoperability aspects can be used by transnational groups in different languages and cultures with the same domain of interest in a cost-effective and efficient manner. Cultural interoperability mechanisms are based on the assumption that both high-level (business processes, strategy, policies) and low-level layers of Enterprise Systems (ICT infrastructures, software, data models, etc.) reflect culture and that the linguistic encoding of knowledge and information is therefore culturally based. These interoperability mechanisms and considerations address the ability of enterprises to understand and co-manage context from any source and of any kind, therefore realising the cooperation between enterprises with major cultural differences.</p>	<table border="1"> <tr> <td>SA.4 - Cultural Interoperability</td> </tr> <tr> <td>Description</td> </tr> <tr> <td>The degree to which knowledge and information is anchored to a unified model of meaning across cultures</td> </tr> <tr> <td>Backlinks</td> </tr> <tr> <td>Hierarchical Links</td> </tr> <tr> <td>Outbound Links</td> </tr> <tr> <td> <ul style="list-style-type: none"> Knowledge Interoperability Social Networks Interoperability </td> </tr> <tr> <td>Indicative Scientific Sub-Areas</td> </tr> <tr> <td> <ul style="list-style-type: none"> Language interoperability Alignment on traditions, religions and ethics </td> </tr> <tr> <td>Tags</td> </tr> <tr> <td>Culture, Context, Language, Linguistics</td> </tr> </table>	SA.4 - Cultural Interoperability	Description	The degree to which knowledge and information is anchored to a unified model of meaning across cultures	Backlinks	Hierarchical Links	Outbound Links	<ul style="list-style-type: none"> Knowledge Interoperability Social Networks Interoperability 	Indicative Scientific Sub-Areas	<ul style="list-style-type: none"> Language interoperability Alignment on traditions, religions and ethics 	Tags	Culture, Context, Language, Linguistics
SA.4 - Cultural Interoperability												
Description												
The degree to which knowledge and information is anchored to a unified model of meaning across cultures												
Backlinks												
Hierarchical Links												
Outbound Links												
<ul style="list-style-type: none"> Knowledge Interoperability Social Networks Interoperability 												
Indicative Scientific Sub-Areas												
<ul style="list-style-type: none"> Language interoperability Alignment on traditions, religions and ethics 												
Tags												
Culture, Context, Language, Linguistics												
<p>References</p> <ol style="list-style-type: none"> 1. Helmen K, As H, Tåsen POB. Organizational barriers to Interoperability. <i>Management</i>. 2010;pp.1-9 2. Lim L, Liu Y. The role of cultural diversity and leadership in computer-supported collaborative learning: a content analysis. <i>Information and Software Technology</i>. 2006;48(3):pp. 142-153. 3. Kwon T. Identifying organisational culture clash in MIS implementation when is it worth the effort? <i>Information & Management</i>. 1991;21(2):99-109. Retrieved February 8, 2011, from http://linkinghub.elsevier.com/retrieve/pii/S037872069190041Y 4. Cayr S, Saaghi a N. Information technology interoperability awareness: A taxonomy model based on information requirements and business needs. <i>FGICT'08 - 2008 Portland International Conference on Management of Engineering & Technology</i>. 2008;(2):946-955. 5. KYOTO: a wiki for establishing semantic interoperability for knowledge sharing across languages and cultures <p>See Also</p> <ul style="list-style-type: none"> Anthonybyrd T, Lewis B, Bryan R. The leveraging influence of strategic alignment on IT investment: An empirical examination. <i>Information & Management</i>. 2006; 43(3):308-321. Camaby P. E-learning and digital library futures in New Zealand. <i>Library Review</i>. 2009; 54(6):344-354. 												

Figure 5.7 - FlNES Wiki: Scientific Area page example

- **Sub Scientific Area Pages** –They have the purpose of describing the sub scientific areas addressed by a specific scientific area. Similarly to the previous page type, each page contains a table that summarizes the sub scientific area. This table contains the name of the sub scientific area paired with its unique identifier, a small description, the scientific area it relates to and a list of tags. Furthermore these types of pages contain the full general description of the sub scientific area which is the main focus of the page. A sub scientific area page example is shown in Figure 5.8.

Unified Cloud Interfaces (SaaS Io)								
<p>Clouds are becoming an everyday word of the IT and business world as they offer many advantages for enterprises. However, as with every IT innovation, the different options available have led enterprises in different paths regarding the clouds they choose and there have been already many problems recorded regarding the limited interoperability between the various deployed applications. In this context, cloud application interoperability is a very serious issue that needs to be tackled by the Enterprise Interoperability community, as there is a need to align clouds by building unified interfaces so that applications deployed in different clouds could communicate and interoperate.</p>	<table border="1"> <tr> <td>SSA.11.1 - Unified Cloud Interfaces (saas.io)</td> </tr> <tr> <td>Scientific Area</td> </tr> <tr> <td>Cloud Interoperability</td> </tr> <tr> <td>Description</td> </tr> <tr> <td>Unified Cloud Interfaces refers to the design of standard interfaces for direct communication and interoperation between clouds from different providers.</td> </tr> <tr> <td>Tags</td> </tr> <tr> <td>Cloud Application Interoperability, Cloud Mapping</td> </tr> </table>	SSA.11.1 - Unified Cloud Interfaces (saas.io)	Scientific Area	Cloud Interoperability	Description	Unified Cloud Interfaces refers to the design of standard interfaces for direct communication and interoperation between clouds from different providers.	Tags	Cloud Application Interoperability, Cloud Mapping
SSA.11.1 - Unified Cloud Interfaces (saas.io)								
Scientific Area								
Cloud Interoperability								
Description								
Unified Cloud Interfaces refers to the design of standard interfaces for direct communication and interoperation between clouds from different providers.								
Tags								
Cloud Application Interoperability, Cloud Mapping								

Figure 5.8 - FlNES Wiki: Sub Scientific Area page example

- **EI Ingredients Pages** – These pages aim to describe an EI Ingredient, i.e., a method, concept, tool, etc... They contain a table that contains the name and a small definition of the ingredient. It also contains a section (General) that contains the main text of the page, a references section which contains information about the citations that occur along the main text of the page. Finally it contains a section (See Also) that contains links to additional information relating to that particular ingredient. An example of this type of page is illustrated in Figure 5.9

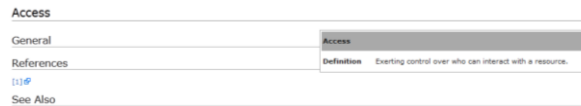


Figure 5.9 - FInES Wiki: EI Ingredient page example

- Publications Pages** – These pages contain information about publications pertinent to the EI community and that are referenced in several pages of the wiki as well as the ones that appear in the “See Also” section of many different pages. These types of pages contain a table that serves to classify the publication according to an EI Ingredient (Tool, Experiment, Standard, etc...). An example of this type of page can be seen in Figure 5.10

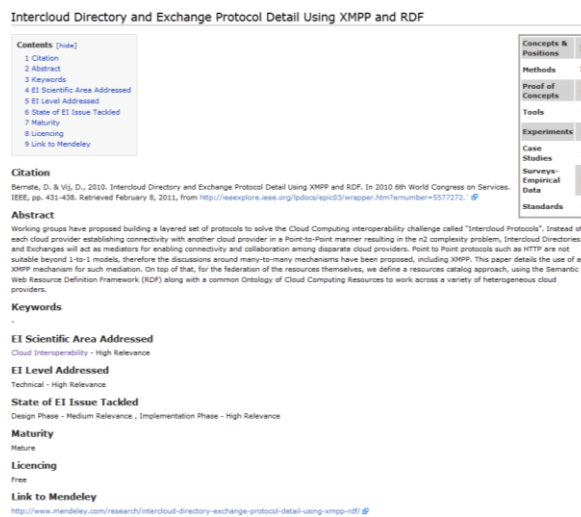


Figure 5.10 - FInES Wiki: Publications page example

5.2.5. EISB Reference Ontology

As previously stated the main goal of this ontology is to represent all the knowledge related to the EISB domain. Here the ontology will be presented fully and in detail. To have a better graphical understanding of the ontology, a good ontology visualization tool is needed. Taking into account the study of this visualizers conducted in section 2.3, the chosen tool to visualize this ontology was XMind. This is mainly because of this tools capability to represent the class hierarchy as well as the properties that serve to relate them in a perceptible way.

A general overview of the entire ontology can be seen in igure 5.11, where all the classes and respective subclasses are represented, as well as the relationships between them. Following, is the detailed description of the ontology.

Taking a top down approach to this description, the ontology, at the top (root) is composed of 5 classes, the Bibliography, Content_Classifier, EI_Contents_Categorization, EISB_Framework and EISB_Wiki classes.

- **Bibliography** - aims to represent all the publications that are featured in the EISB wiki and their authors. To achieve this goal, 4 properties were created that have this class as a domain, 2 datatype properties and 2 object properties. The datatype properties are the *Link* and *Citation* properties. The former aims to store the website from which the users can download or buy the respective publication. On the other hand the *Citation* property was defined to store the citation that is to be used by authors if they want to cite the respective publication in their work. The 2 object properties defined in this class are the *relatedTo_Publication* and the *AuthoredBy* properties. The first one has the goal of relating the instances defined in the Bibliography class to the instances defined in the Publications class (which will be discussed in detail shortly). This property was also defined as being functional to ensure that each instance in the Bibliography class has at most one corresponding entry in the Publications class. The second object property defined for this class aims to relate the authors to their corresponding publications. This means that each instance of the Bibliography class will be related to instances defined in the Researchers class (to be presented further along this description). Contrarily to the *relatedTo_Publication* property, the *AuthoredBy* property is not functional because a Bibliography instance can have more than one author.
- **Content_Classifier** - aims to store information relative to classifications of the EISB wiki contents. This class is subdivided into 4 other subclasses with the objective of storing specific classification types.
 - **EI_Barrier_Classifier** - holds the classification of a certain content regarding its interoperability barrier category. Instances in this class have 2 properties, *Relevance* which is a datatype property, and the *hasBarrier* object property. The first property holds the relevance of the classification and it must be one of three values, low, medium or high. The *hasBarrier* property has the objective of relating the classification with a respective barrier in the *Interoperability_Barriers* class. It is a functional property since a classification of this type must relate only to one type of barrier.
 - **EI_Maturity_Classifier** - stores information relative to the maturity of the wiki content. It only has an object property, *hasMaturity* that aims to relate the classification with an instance of the *Interoperability_Maturity* class.
 - **EI_Phase_Classifier** - has the goal of classifying wiki content relatively to its development lifecycle. Like the *EI_Barrier_Classifier* subclass, this one also has the *Relevance* property to rate the classification as being low, medium or high. This subclass also contains an object property, *hasPhase* that relates the instances of the classification to a certain instance that represents a phase of the *Development_Lifecycle* class.
 - **Scientific_Area_Classifier** - was created with the purpose of classifying certain wiki content with the relevance pertaining to a certain scientific area. Like the previous subclass, this one also has the *Relevance* datatype property to classify the content with low, medium or high relevance. Furthermore it also has an object property, *scientificArea* that relates the classification to a certain scientific area defined in the *EI_Scientific_Areas* or *EISB_Neighbouring_Scientific_Areas* classes.

- **EI_Contents_Categorization** - is tightly related to the previously described class (Content_Classifier). This class houses the information about the different categories that the content of the wiki can take. It is divided into 3 subclasses which will be individually discussed. Furthermore, this class has a single datatype property which is called *Name* and keeps the name of the category of the content. Also, this property is propagated to all subclasses under its domain.

- **Development_Lifecycle** – houses the information about the different development phases that certain content is in and it further divided into 3 subclasses.

- **Assessment**
- **Design**
- **Implementation**

The instances created in these subclasses are the ones that will be used to relate the content classification to its phase via the previously presented *hasPhase* property.

- **Interoperability_Barriers** – records the information regarding the barriers that a certain content can encounter. Like the previous subclass, the Interoperability_Barriers subclass is also divided into its own subclasses, representing the so called barriers.

- **Conceptual**
- **Organizational**
- **Techonological**

The instances created in these subclasses are the ones that relate the content classification to the interoperability barriers via the *hasBarrier* property

- **Inteoperability_Maturity** - intends to hold information about the various maturity classification categories. To this effect this subclass was also divided into several subclasses of its own.

- AIF
- C4IF
- Humanistic
 - Deprecated
 - Elder
 - Infant
 - Mature
- Interoperability_Classification_Framework
- LISI
- NC3TA_RMI
- OIM

Like the previous cases, the instances created within the various subclasses of the Interoperability_Maturity class are the ones that relate the maturity classification of content to their respective maturity category by means of the *hasMaturity* property.

- **EISB_Framework** - aim of this class is to hold information about the elements that compose the EISB universe. To that effect three subclasses were defined.

- **EISB_Knowledge_Base** - contains an object property named *instancedBy* which aims to illustrate a relation of origin, meaning that that an instance associated with this property is originated within this class. This property is also propagated to the subclasses and its goal is the same, however the contexts are different.
 - **EI_Scientific_Areas** - holds information about the various scientific areas represented in the EISB universe, and these subclasses are also divided into other subclasses that illustrate the scientific sub areas. These classes also contain the *Name* datatype property that stores the name of the scientific areas and scientific subareas. The instances created under these classes are the ones that are used to relate the scientific area classification of content to the respective scientific area via the *scientificArea* property.
 - Various Scientific Areas
 - Various Scientific Sub Areas
 - **EISB_Community** - Contains information about different researcher communities present in the EISB universe. This is why this class is also divided into different subclasses that represent each community respectively.
 - **Experts_Scientific_Committee**
 - **Related_Scientific_Disciplines_Community**
 - **Validation_Community**
 - **Other_Relevant_Communities**

These classes also contain the *Name* property to record the name of the communities

- **EISB_Neighbouring_Scientific_Areas** - is very similar to the *EI_Scientific_Areas* since it is also divided into subclasses that represent the scientific areas and scientific sub areas (if they exist), however in this context the scientific areas belong the EISB neighbouring domains instead of the EISB domain.
 - Various Neighbouring Scientific Areas
- **Tangible_Content** - contains information about the actual contents of the EISB universe. These contents are divided into 3 subclasses
 - **EISB_Ingredients** - is divided into subclasses that represent the ingredients themselves which are used in the classification of scientific publications, i.e. if it as case study, a standard, a method, etc...
 - Various Ingredients
 - **Expert** - The Expert subclass is used to classify the researchers involved in the EISB universe by relating them using the *instancedBy* property
 - **Scientific_Publication** - class is used to classify publications pertinent to the EISB, and they also relate via the *instancedBy* property.
- **EISB_Problem_Space**
- **EISB_Solution_Space**

- **EISB_Hypothesis**
- **EISB_Laws**
- **EISB_Wiki** – The objective of this class is to represent all elements that compose the EISB (FInES) Wiki. This class also holds *FINES_Page* datatype property that holds the direct web link to the wikis main page (in this case). This property is propagated to all the subclasses of this one with the same objective. However the links will obviously be different for each instance. This class also contains the object property *isInstanceOf* which is the inverse of a previously discussed property named *instancedBy*, meaning that the relation can now be seen as that instance x was originated by instance y. Instead of being instance y originates instance x. It is worthy of note that this property is also propagated to the subclasses but the instances contained in them will have a different values.
 - **EISB_Glossary** – This class contains the contents that are represented in the glossary page of the EISB wiki. To achieve that goal, the class is divided in the following subclasses. This class also contains some properties that are also propagated to its subclasses. One property is the *FINES_Page* property which holds the link to the respective wiki page. Another property is the *Name* property which contains the name of the respective content. The *Definition* property was also created and its aim is to hold small definitions of a respective content. Finally the previously described *isInstanceOf* property is also present.
 - **EI_Ingredients** – This class holds the detailed information about the various ingredients (concepts) that are represented in the EISB_Wiki. Therefore some properties, along with the ones inherited from the upper class, were defined. These properties are the *MainText* datatype property which holds all of the text in the wiki page of the respective ingredient. The *hasReference* object property holds the instances of the bibliography that is referenced along the text in the wiki page and that appear in the References section of the wiki page. The *hasSeeAlso* object property holds the instances of the Bibliography class that appear in the See Also section of the wiki page. The other properties that compose this subclass are the ones that were inherited from the upper class, and as such will not be described here.
 - **Scientific_Area** – This class holds all the details regarding the EISB scientific areas represented in the EISB wiki. Apart from the inherited properties (which won't be described here) this class presents the following properties. The *ID* datatype property holds the unique identifier of a certain scientific area. The *MainText* datatype property holds the text of the wiki page. The *hasReference* and *hasSeeAlso* object properties have the same purpose as in the EI_Ingredients class. The *hasSubArea* subclass relates the scientific areas to their corresponding scientific sub areas, so the range of this property is Scientific_SubArea class (to be described shortly). The *hasTags* object property holds the ingredients, publications or neighbouring ingredients that are represented in the wiki page of

the scientific area. The *includes*, *relatesTo* and *requires* object properties serve to relate a scientific area with other scientific areas or scientific sub areas.

- **Scientific_SubArea** – This class holds all the details regarding scientific sub areas represented in the EISB wiki. The inherited properties won't be described here. Apart from those properties, this class contains the *hasSuperArea* object property that is the inverse of the *hasSubArea* property and serves the purpose of relating the scientific sub areas with their respective scientific areas. The *ID* datatype property holds the unique identifier of the scientific sub area. The *MainText* datatype property holds the text of the wiki page. The *hasReference*, *hasSeeAlso*, *hasTags*, *includes*, *relatesTo* and *requires* object properties serve the same purpose as the ones describe for the *Scientific_Area* class.
- **EISB_Neighbouring_SDRG** – This class serves the same purpose of the *EISB_Glossary* class, however it refers to the Neighbouring domains instead of the EISB domain. Apart from the properties inherited from the root class, this class has 2 other datatype properties. The *Definition* property which holds a small definition of the content, and the *Name* property which records the name of the content.
 - **Core_Features** – This class holds the information about the core features of the EISB neighbouring domains. Apart from the inherited properties which won't be described again here, this subclass contains several other properties. The *hasReference*, *hasSeeAlso* and *hasTags* object properties serve the same purpose as the ones described for the *Scientific_Area* class. The *MainText* datatype property holds the text of the respective wiki page. The *relatedTo_EI_ScientificArea* relates the core features of the neighbouring domains with the EISB scientific areas. The *relatedTo_EISB_Neighbouring_Area* relates the core features with the scientific areas of the neighbouring domains. The *EISB_Relation* holds a small description as to how this feature relates to the EISB universe.
 - **Neighbouring_Ingredients** – This subclass holds the information regarding the ingredients of the EISB neighbouring domains. Apart from the inherited properties, this class contains the *hasReference* and *hasSeeAlso* object properties that serve the same purpose as the ones described in previous classes. It also contains the *MainText* datatype property that records the text of the corresponding wiki page. The *relatedTo_CoreFeature* object property relates the neighbouring domain ingredients to neighbouring domains core features.
 - **Neighbouring_Scientific_Area** – This class records all the data regarding the EISB neighbouring domains scientific areas. To that effect and separately from the inherited properties, this subclass has the *hasTags* object property and the *MainText* datatype property which has the same objective as the ones described for previous classes. Furthermore it also has the *hasCoreFeatures* object property which is the inverse of the *relatedTo_EISB_Neighbouring_Area* property and aims

to relate the EISB neighbouring domains scientific areas to their respective core features.

- **Publications** – This class aims to hold all the information regarding all the publications represented in the EISB Wiki. To that end various properties were defined. More specifically 6 datatype properties were defined along with 4 object properties (including the properties that were propagated from the root class). The datatype properties will now be presented in detail. *Abstract* property holds the textual form of the abstract section presented in the wiki pages representing publications. *HasLicence* property holds the value for the licencing section of the wiki page. *Keywords* property stores the value of the keywords section of the wiki page. The *linkMendeley* property holds the link to the mendeley website of the respective publication. The *Name* property holds the name of the publication. And the *FINES_Page* property holds the link to the wiki page of the publication. Now, the object properties will be described. The *hasIngredient* property relates the publications with none, some or all the ingredients in the *EISB_Ingredients* class. The *isClassifiedAs* property relates the publications with the classifications regarding its barrier, phase, maturity and scientific area. This means that this property will have instances that were created in *Content_Classifier* class. The *relatedTo_Bibliography* property is the inverse of the *relatedTo_Publication* property that was previously presented. This property relates the instances of the *Publications* class with the corresponding ones in the *Bibliography* class. The *isInstanceOf* property in this case, will relate the instances of the *Publications* class with the ones in the *Scientific_Publication* class.
- **Researchers** – This class handles detailed information about the researchers of the EISB universe. To this effect 5 datatype properties and 3 object properties were defined. The datatype properties are the *FINES_page*, which holds the link to the researcher's wiki page. The *FirstName* and *LastName* hold the first name and the surname of the researcher, respectively. The *Contact* property holds various contacts of the researcher (e-mail, phone, etc...). The *Organization* property holds the organization(s) which the research is affiliated with. Regarding the object properties, they are, the *belongsToCommunity* property that relates the researcher with the community or communities which he is associated with via the instances created in the *EISB_Community* subclasses. The *workedOn* property is the inverse of the *AuthoredBy* property that was previously described. This property holds the instances of the *Bibliography* class in which the researcher has participated. The *isInstanceOf* property in this case will hold the instance created in the *Expert* class.

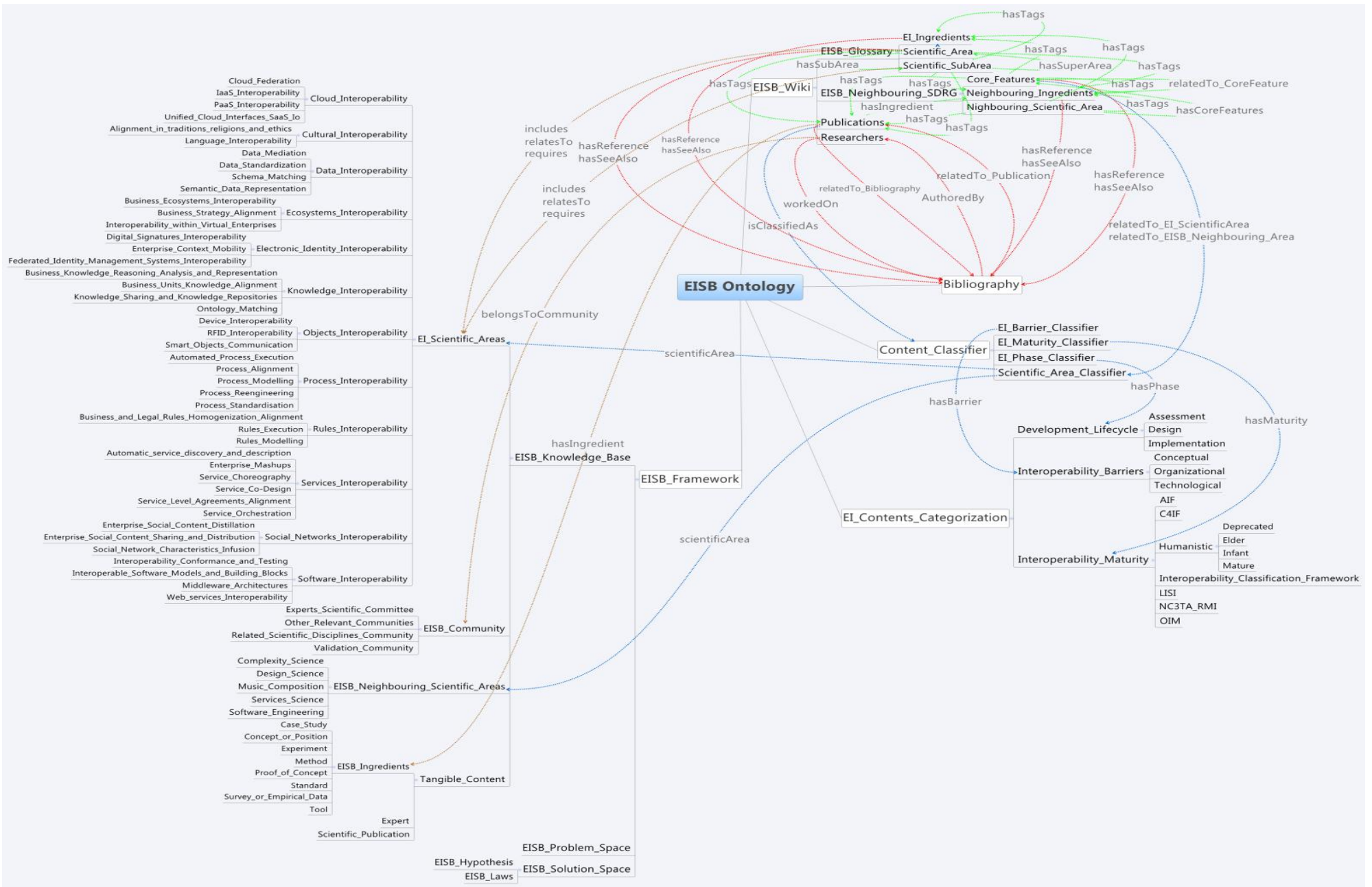


Figure 5.11 - EISB Reference Ontology overview

5.3. Synchronization execution flows

To have a better general understanding of how the synchronization tool works, the flow of execution and how the information is processed are presented in this subsection. Firstly, the flow of execution of the synchronization tool from the EISB Ontology to the FInES wiki is introduced and after, the reverse route is presented. These sequence diagrams, serve to complement the previously shown in architecture in a sense that it is shown in detail the flow of execution of the system.

5.3.1. EISB Ontology to FInES Wiki Synchronization Execution Flow

As can be observed in Figure 5.12, the user first activates the tool which allows it to perform some initializations, such as loading the ontology to prepare for editing and constructing the required java classes for synchronization. When these initializations are complete the program signals the user and it's at that point that the user can instruct the tool to begin synchronizing. At this moment the program connects to the wiki DB to verify that synchronization is, indeed, possible. When the connection is established the developed tool then proceeds to perform the actual synchronization. It starts by getting the changes recorded in the ChAO ontology, which is associated with the EISB Ontology. It is worthy of note that the EISB ontology isn't directly involved in this procedure because all the changes that are made in it are recorded in the ChAO ontology and therefore all the information required for synchronization can be accessed directly from the ChAO ontology. After obtaining the modifications perpetrated in the ontology, the program processes them in order to maintain the consistency of the extracted contents, and places them in specific tables of the wiki DB. Finally, before signalling the user that the synchronization process has been completed, the program deletes all the changes from the ChAO ontology to ensure that on the next execution of the software, the same contents won't be resynchronized.

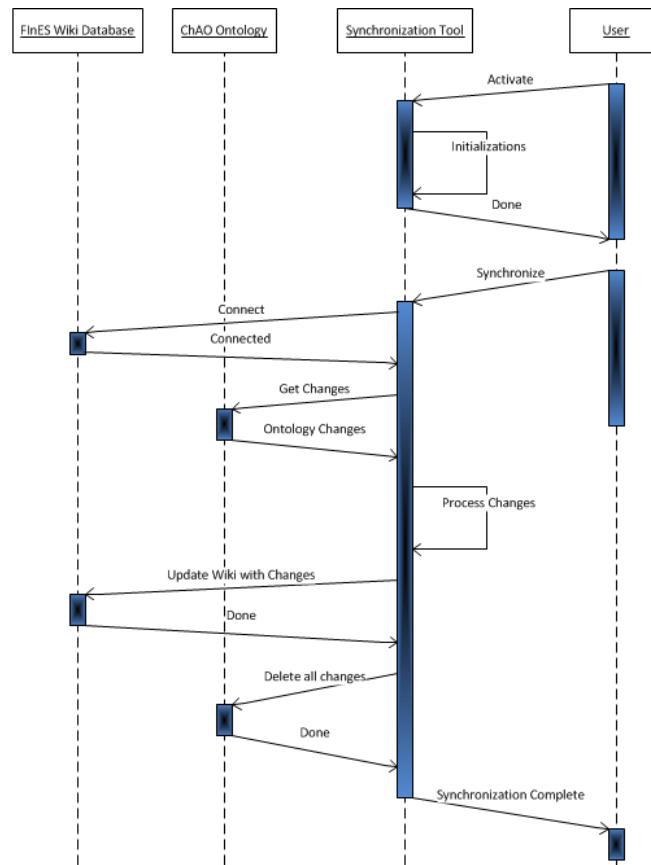


Figure 5.12 - Ontology to Wiki Synchronization execution flow

5.3.2. FInES Wiki to EISB Ontology Synchronization Execution Flow

In this subsection it's intended to describe the execution flow of the developed tool regarding the synchronization between the EISB ontology and the FInES wiki. As can be perceived in Figure 5.13, the process starts in the same manner as when the synchronization is between the EISB ontology and the FInES wiki. The user activates the program and it begins by performing the same initializations as it did in the previous scenario. After the user gives the command to begin synchronization, the program connects to the wiki DB and proceeds to get the modifications that have occurred in the wiki. Upon obtaining those changes, the program processes them, once again to ensure that the information remains consistent, and updates the EISB ontology accordingly. Finally, the program saves the ontology file, that ensure that the update isn't lost and erases all the changes in the wiki DB to guarantee that the next time the program is run, the same changes won't be synchronized again. When all those steps are complete, the developed tool signals the user that the synchronization process is complete.

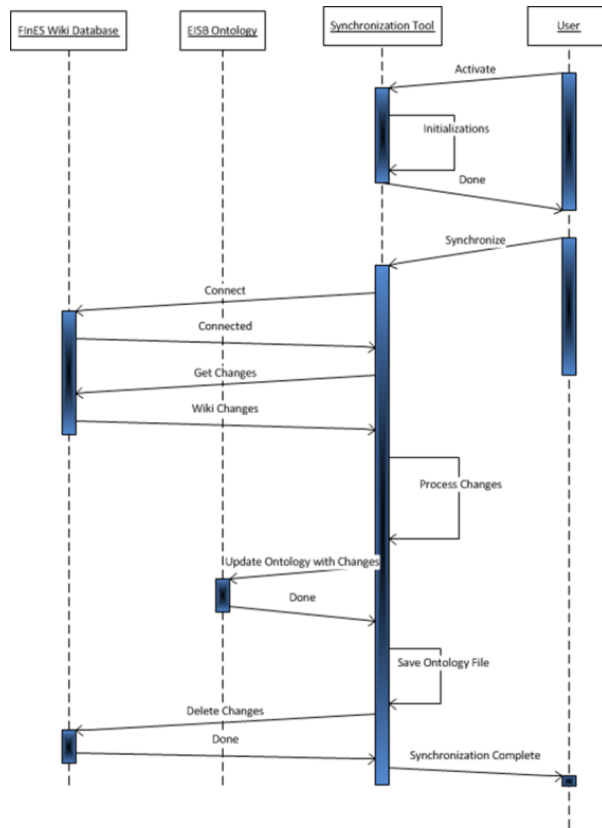


Figure 5.13 - Wiki to Ontology synchronization execution flow

5.4. Concluding Remarks

The work conducted throughout this chapter features the study behind the development of the implementation of the synchronization process between the EISB reference ontology and the FInES wiki. This study is what allows an effective implementation of the synchronization, since as it was referred in 3.4 regarding item 5, the knowledge of the synchronization procedure facilitates the semantic checking process. Furthermore the study conducted in this section enabled a better understanding of the system and how its components interact with each other, and with the aid of the sequence diagrams, a visual and temporal understanding of how the synchronization process is done is facilitated.

6. SYNCHRONIZATION TOOL DEMONSTRATION

This chapter of the document shows a demonstration example related to the multiple structural semantic checking scenario introduced in section 4.2.2 and features the results of the developed synchronization tool which was implemented according to the architecture presented in Figure 5.1.

The examples presented here intend to demonstrate how the synchronization tool works in detail. Firstly an example of synchronization from the EISB reference ontology to the FInES wiki is presented in subsection 6.1. Following, an example of synchronization from the FInES wiki to the EISB Reference Ontology is presented in subsection 6.2. However, before going in to the examples, it is important to demonstrate the common steps that always take place when running the synchronization tool. Upon executing this tool, the users find a GUI, shown in Figure 6.1, from which they can control the synchronization process. In this user interface, the users first have to specify some information such as the wiki DB name, username and password, in order to allow the program to access it. Furthermore, users need to specify the ontology project location as well as the project name for the program to know which ontology will be involved in the synchronization process. After all that information is specified, users need to click the activate button in order for the program to perform some initializations. Once these steps are complete, the users are then allowed to click the synchronize button, and what happens afterwards will be described in following two subsections.

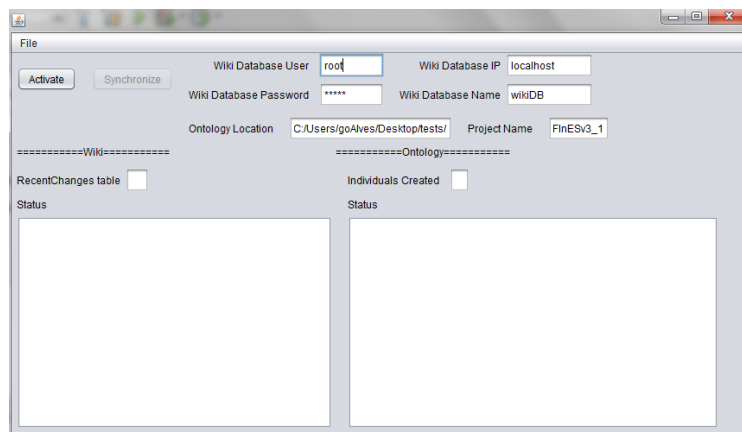


Figure 6.1 - Synchronization tool GUI

6.1. Ontology to Wiki Synchronization Demonstration

Two examples of ontology to wiki synchronization are going to be presented in this subsection. The first example chosen here to demonstrate how the synchronization tool works features a scientific area instance created in the EISB ontology being synchronized into the FInES wiki. The second example features the removal of a scientific area of the ontology and its synchronization with the wiki. However before going into the specific examples, a thorough analysis of all the cases that may occur when synchronizing the two elements was made.

Table 6.1 shows in the first column the cases that may occur when the synchronization process is between the EISB ontology and the FInES wiki. The middle column denotes the recommended course of action (if any) to take part in the wiki for each specific case that ensues in the ontology. Finally, the third column denotes which cases have been implemented in the synchronization tool prototype.

Table 6.1 - Ontology to Wiki synchronization cases analysis

Ontology Case	Action	Implemented
New instances	If new instances are part of any of the subclasses of the EISB_Glossary class or in the publications class then a wiki page has to be created for each of them, with the contents built using the values of the instances properties. Else no wiki related action is needed.	Yes
New classes	If the new classes are a subclass of the EI_ScientificAreas class then a wiki category page must be created for each of them. Else if the new classes belong to the EISB_Ingredients subclass then the EISB Papers Classification Metadata template page must be altered to accommodate the new ingredient. Else no wiki related action is needed.	-
New proprieties	If the new proprieties have the Publications class or the EISB_Glossary class as domain then the wiki page contents must reflect these new proprieties. Else no wiki related action is needed.	-
Edit instances	If the edited instance is part of the EISB_Glossary or Publications classes then the corresponding wiki page must be edited to reflect the changes recorded. Else no wiki related action is needed.	-
Edit classes	If the edited classes belong to the EI_ScientificAreas class then the corresponding wiki category page must reflect the changes. Else if the edited classes belong to the EISB_Ingredients class then the EISB Papers Classification Metadata template page must be altered to accommodate the changes. Else no wiki related action is needed.	-
Edit proprieties	If the edited proprieties have the Publications class or the EISB_Glossary class as domain then the wiki page contents must reflect these proprieties Else no wiki related action is needed.	-
Remove instances	If the deleted instances are part of the EISB_Glossary or Publications classes then the corresponding wiki pages must also be deleted. Else no wiki related action is needed.	-
Remove classes	If the removed classes belong to the EI_ScientificAreas class then the corresponding instances in ScientificAreas and subScientificAreas must also be deleted which in turn will remove the corresponding wiki pages. If the removed classes are subclasses of EISB_Ingredients then the EISB Papers Classification Metadata template page must be altered to accommodate the changes and the corresponding instances in the EI_Ingredients class must also be removed. Else no wiki related action is needed.	Yes (for scientific areas and sub scientific areas)
Remove proprieties	If the removed proprieties have the Publications class or the EISB_Glossary class as domain then the wiki page contents must reflect these changes. Else no wiki related action is needed.	-

6.1.1. New Scientific Area instance

Upon performing the previously presented and required initializations the user can then start the synchronization process. After the user clicks the synchronize button on the tools interface, the program checks the ChAO ontology to get the changes that have occurred in the EISB ontology. In this case, the program verifies that a new scientific area instance has been added to the EISB ontology, as shown in Figure 6.2.

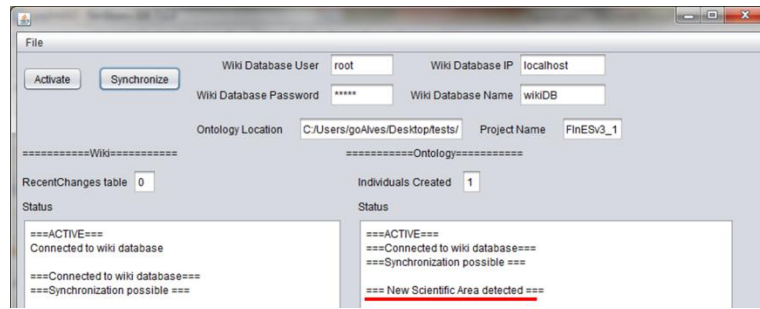


Figure 6.2 - Ontology to Wiki Synchronization - New Scientific Area instance detection

More specifically, in this demonstration, the instance created in the EISB ontology is of the “Social Networks Interoperability” scientific area, which can be observed in Figure 6.3, on the Protégé interface.

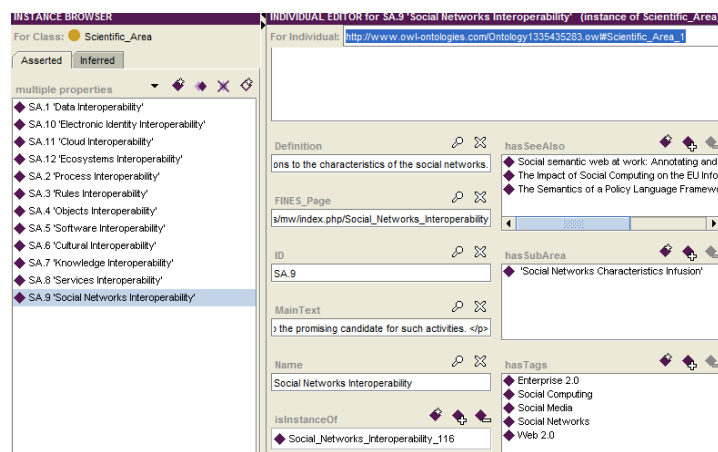


Figure 6.3 - Ontology to Wiki Synchronization - Scientific Area instance

The synchronization tool then proceeds to obtain the values of the properties associated with that instance and builds a string from those values to form the wiki page content. Afterwards, three entries are added to three different tables of the wiki DB. Firstly an entry is added to the page table that the wiki uses to identify each page using its title [67]. Then an entry is added to text table of the DB, which is where the wikitext of individual page revisions are stored [68]. Lastly, an entry is added to revision table which is needed because this table holds the metadata for every edit done to a page within the wiki [69] (including the creation of pages). When these entries are made, the synchronization process for this particular instance is finished and the result on the wiki can be seen on part (a) of Figure 6.4, while the finished process on the java interface can be observed in half (b).

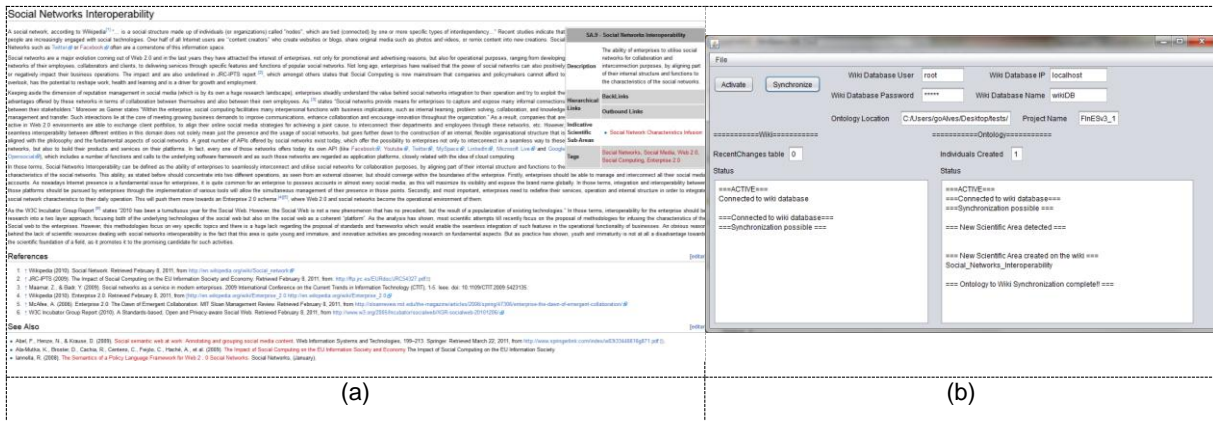


Figure 6.4 - Ontology to Wiki Synchronization - New Scientific Area instance finished synchronization

Afterwards, the ontology program deletes all references to that instance in the ChAO ontology to ensure that this particular instance won't be resynchronized in the future.

Finally, the top portion of Figure 6.5 shows the representation of the "Social Networks Interoperability" scientific area in the EISB ontology (Protégé interface), whereas the bottom portion shows the "Social Networks Interoperability" scientific area page on the FInES wiki. As seen, the various properties of the instance have a correspondence in the wiki page, ensuring that the contents are well transferred.

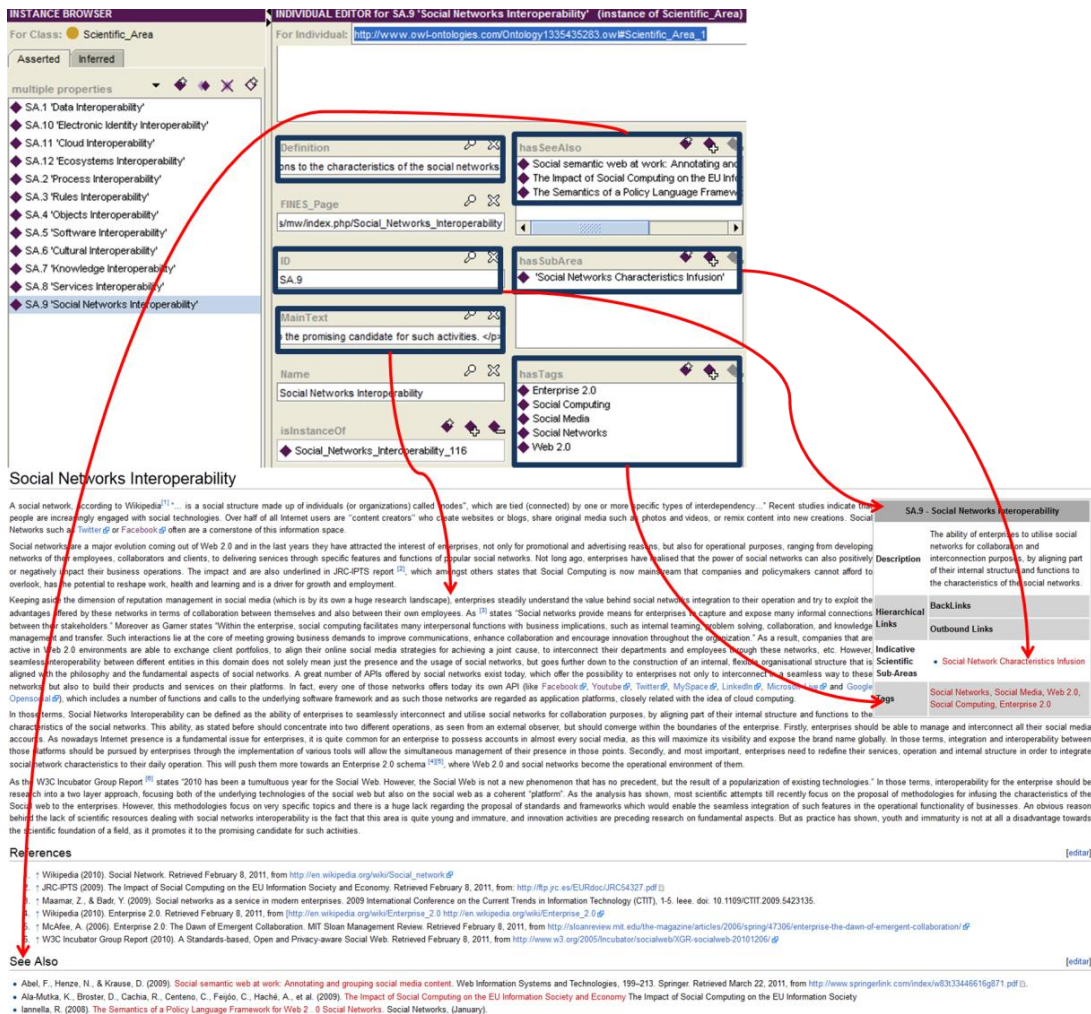


Figure 6.5 - Ontology to Wiki new Scientific Area synchronization example

It is worthy of note that a portion of the java code used to perform this synchronization is present in appendix 9.1.

6.1.2. Remove Scientific Area class

Similarly to the previous demonstration, the user starts by performing the required initializations of the synchronization tool. The user then presses the synchronization button on the tools interface to begin the process. The program starts by checking the ChAO ontology for any changes that may have occurred in the EISB ontology. Specifically in this case, the program detects that a class has been removed, namely the “Cloud Interoperability” scientific area, as shown in Figure 6.6.

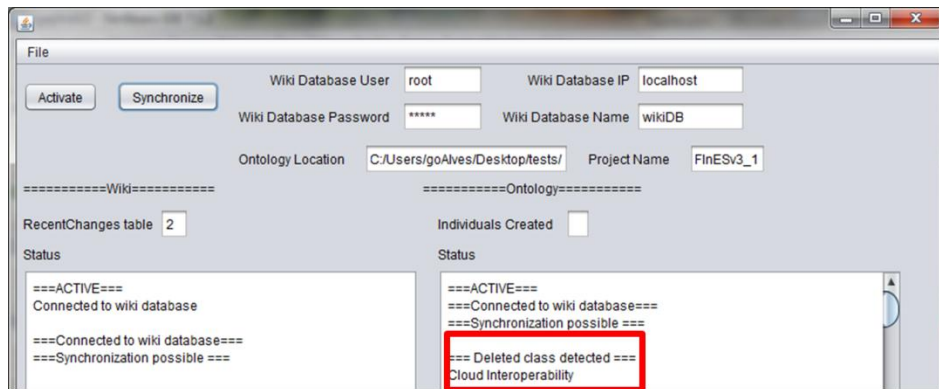


Figure 6.6 - Ontology to Wiki Synchronization - Deleted Class detection

As a side note, to demonstrate that the “Cloud Interoperability” scientific area class was indeed erased from the ontology, Figure 6.7 is presented, where part (a) shows the structure of the ontology before the deletion, while part (b) the resulting class hierarchy of the ontology after deletion.

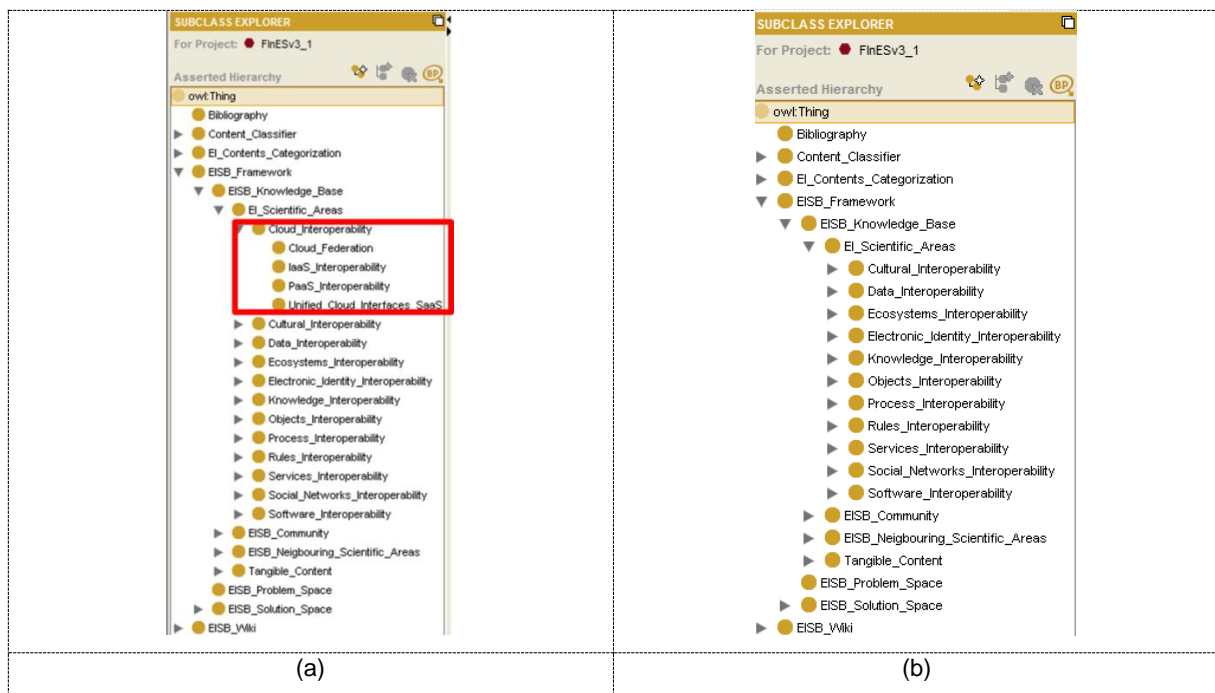


Figure 6.7 – EISB Reference Ontology (a) Before class deletion; (b) After class deletion

It can also be observed Figure 6.7 (a) that the “Cloud Interoperability” scientific area contains four subclasses that compose its sub scientific areas. Since the scientific area was removed, consequently, all of its sub scientific areas were also erased. Therefore, the synchronization tool will also have to deal with them.

After detecting the “Cloud Interoperability” scientific area class removal the synchronization tool proceeds to deleting the wiki page that corresponds to that scientific area. Subsequently, the wiki pages corresponding to the scientific sub areas of the “Cloud Interoperability” scientific area are also deleted, since they no longer figure in the ontology. These page deletions are signalled by the synchronization tools interface as seen in Figure 6.8.

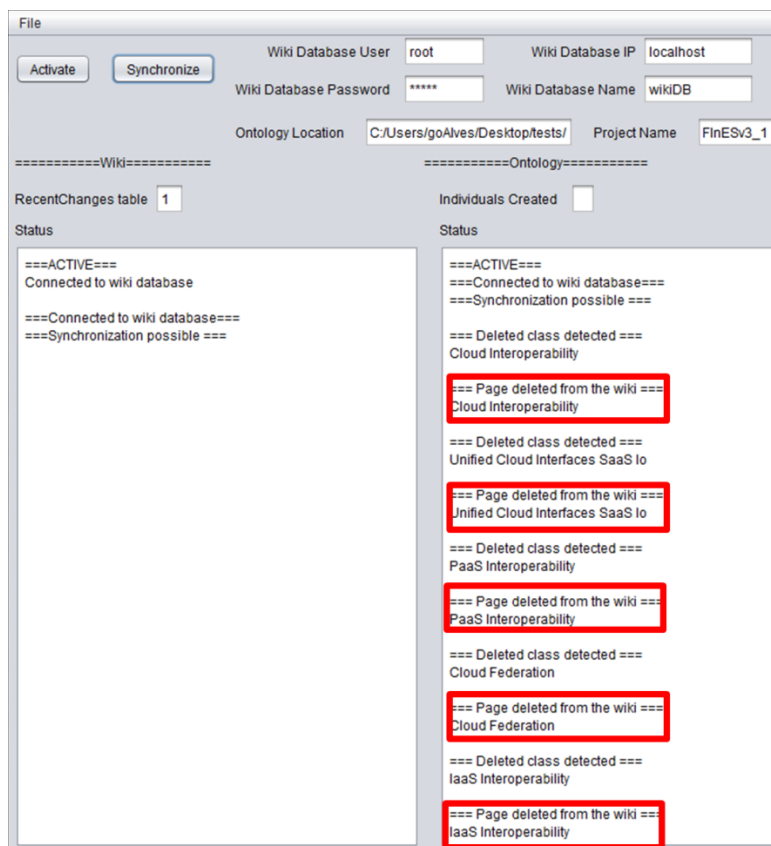


Figure 6.8 - Ontology to Wiki Synchronization - Wiki page deletion (Java GUI)

After deleting the wiki pages the synchronization process is finished and the application also erases all references to the deleted classes to avoid conflicts in future synchronizations. The results of this specific process can be observed in Figure 6.9, where half (a) illustrates the wiki page before deletion whereas part (b) denotes the wiki page after deletion. As seen, the wiki page was ,in fact, erased ensuring that the synchronization process was successful.

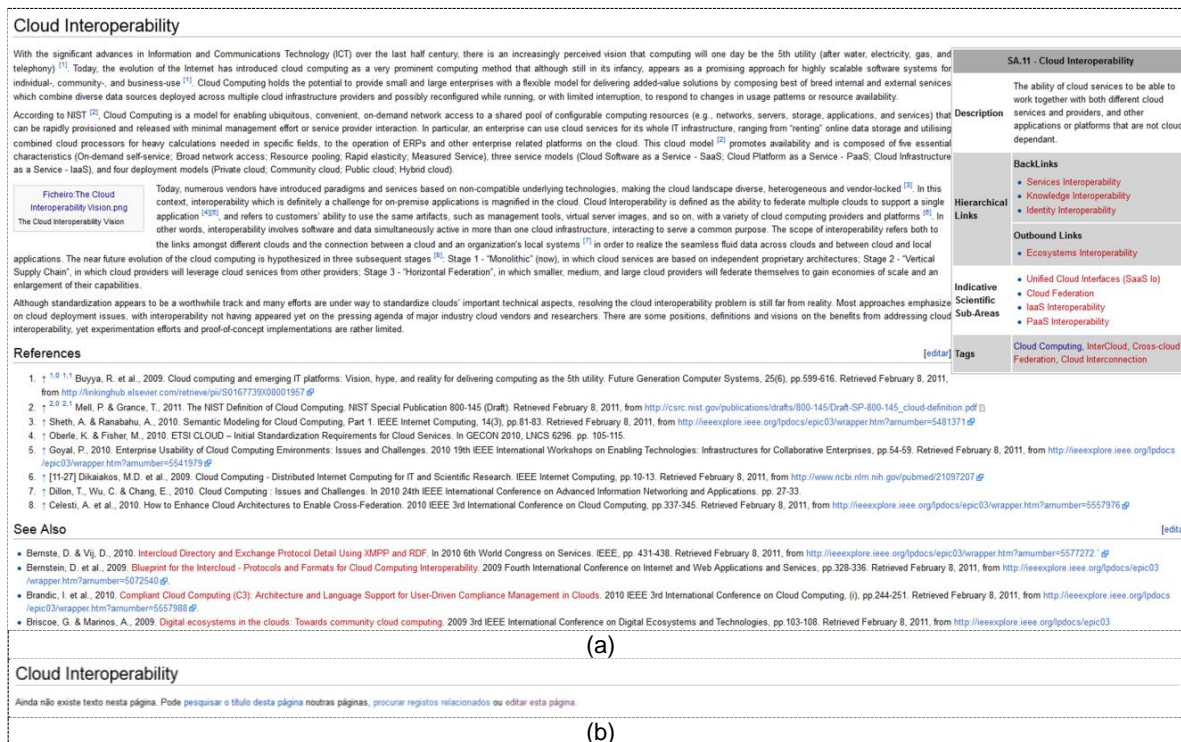


Figure 6.9 - Ontology to Wiki Synchronizaton. (a) Wiki page before deletion; (b) Wiki page after deletion

As happened with the previous example, some of the java code developed to perform this synchronization task is illustrated in appendix 9.2.

6.2. Wiki Ontology Synchronization Demonstration

In this subsection, firstly an example of the synchronization process between the wiki and the ontology is the creation of a new publication page on the wiki will be presented. Next an example of editing a scientific area page in the wiki and posterior synchronization with the ontology will be presented. However before going into the specific demonstration examples, a study of the cases that can occur when synchronizing the wiki with the ontology was made.

Table 6.2, on the first column, shows the identified cases when the synchronization is between the wiki and the ontology. The second column indicates the recommended action to take in the ontology for each specific case that occurs in the wiki. Finally, the third column specifies which cases have been implemented in the developed synchronization tool prototype.

Table 6.2 - Wiki to Ontology Synchronization cases analysis

Wiki Case	Action	Implemented
New publication	New instance in Publications class under the EISB_Glossary class with properties filled according to wiki text New instance in bibliography class with properties filled according to wiki text. Create instances for new researchers that don't yet exist.	Yes
New Ingredient	Create a new instance in EI_Ingredients class with properties filled according to wiki text.	Yes
New SA	1. Create new sub-class in the EI_ScientificAreas class;	Yes

Wiki Case	Action	Implemented
	<ol style="list-style-type: none"> 2. Create new instance in that same subclass 3. Create new instance in ScientificAreas class under the EISB_Glossary class 	
New SSA	<ol style="list-style-type: none"> 1. Create new sub-class in scientific area that this sub area is part of in the EI_ScientificAreas class; 2. Create new instance in that same subclass 3. Create new instance in <u>subScientificAreas</u> class under the EISB_Glossary class 	Yes
Edit Publication	Get the respective instance in the publications class and edit the values of the properties according to the changes verified in the wiki text Also edit the corresponding bibliography instance filling the values of the properties according to the wiki text. Edit the respective researchers instances (if needed)	-
Edit Ingredient	Get the respective instance in the EI_Ingredients class and edit the values of the properties according to the changes verified in the wiki text	-
Edit SA	Get the respective instance in the ScientificAreas class and edit the values of the properties according to the changes verified in the wiki text	Yes
Edit SSA	Get the respective instance in the subScientificAreas class and edit the values of the properties according to the changes verified in the wiki text	Yes
Remove publication	Remove the respective instance from the Publications class and also remove the respective bibliography instance	-
Remove ingredient	Remove the respective instance from the EI_Ingredients class	-
Remove SA	Remove the respective instances from the ScientificAreas and EI_ScientificAreas class. Also remove the corresponding subclass from the EI_ScientificAreas class	-
Remove SSA	Remove the respective instances from the subScientificAreas and EI_ScientificAreas class. Also remove the corresponding sub-subclass from the EI_ScientificAreas class	-

6.2.1. New Publication

In this example, the “Cloud Computing” publication was created on the wiki and then the synchronization tool was run to perform the synchronization with the ontology.

The user begins by executing the synchronization tool and performing the previously described initializations. Once the user clicks the synchronization button of the java application GUI, the program checks the “recentchanges” table of the wiki DB for any changes that have occurred in the wiki since the tool was last run. In this particular situation, as referred earlier, the tool detects a new publication, as shown in Figure 6.10.

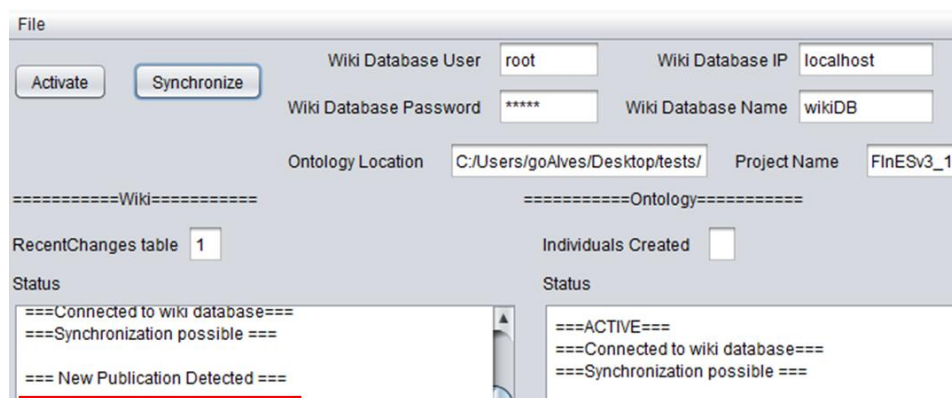


Figure 6.10 - Wiki to Ontology Synchronization example - New publication detection

Subsequently, the program retrieves the page content from the wiki DB and proceeds to breakdown

the different sections of the page. In this example the newly created page refers to the “Cloud Computing” publication which is illustrated in Figure 6.11.

Cloud Computing

Contents [hide]

- 1 Citation
- 2 Abstract
- 3 Keywords
- 4 EI Scientific Area Addressed
- 5 EI Level Addressed
- 6 State of EI Issue Tackled
- 7 Maturity
- 8 Licencing
- 9 Link to Mendeley

Citation

Cunsolo, V.D. et al., 2010. Cloud Computing N. Antonopoulos & L. Gillam, eds. Computer Communications, pp.93-111. Retrieved February 8, 2011, from <http://www.springerlink.com/index/10.1007/978-1-84996-241-4>.

Abstract

Computing as you know it is about to change, your applications and documents are going to move from the desktop into the cloud. I'm talking about cloud computing, where applications and files are hosted on a "cloud" consisting of thousands of computers and servers, all linked together and accessible via the Internet. With cloud computing, everything you do is now web based instead of being desktop based. You can access all your programs and documents from any computer that's connected to the Internet. How will cloud computing change the way you work? For one thing, you're no longer tied to a single computer. You can take your work anywhere because it's always accessible via the web. In addition, cloud computing facilitates group collaboration, as all group members can access the same programs and documents from wherever they happen to be located. Cloud computing might sound far-fetched, but chances are you're already using some cloud applications. If you're using a web-based email program, such as Gmail or Hotmail, you're computing in the cloud. If you're using a web-based application such as Google Calendar or Apple Mobile Me, you're computing in the cloud. If you're using a file- or photo-sharing site, such as Flickr or Picasa Web Albums, you're computing in the cloud. It's the technology of the future, available to use today.

Keywords

-

EI Scientific Area Addressed
Cloud Interoperability - High Relevance

EI Level Addressed
Conceptual - High Relevance

State of EI Issue Tackled
Design Phase - High Relevance

Maturity
Mature

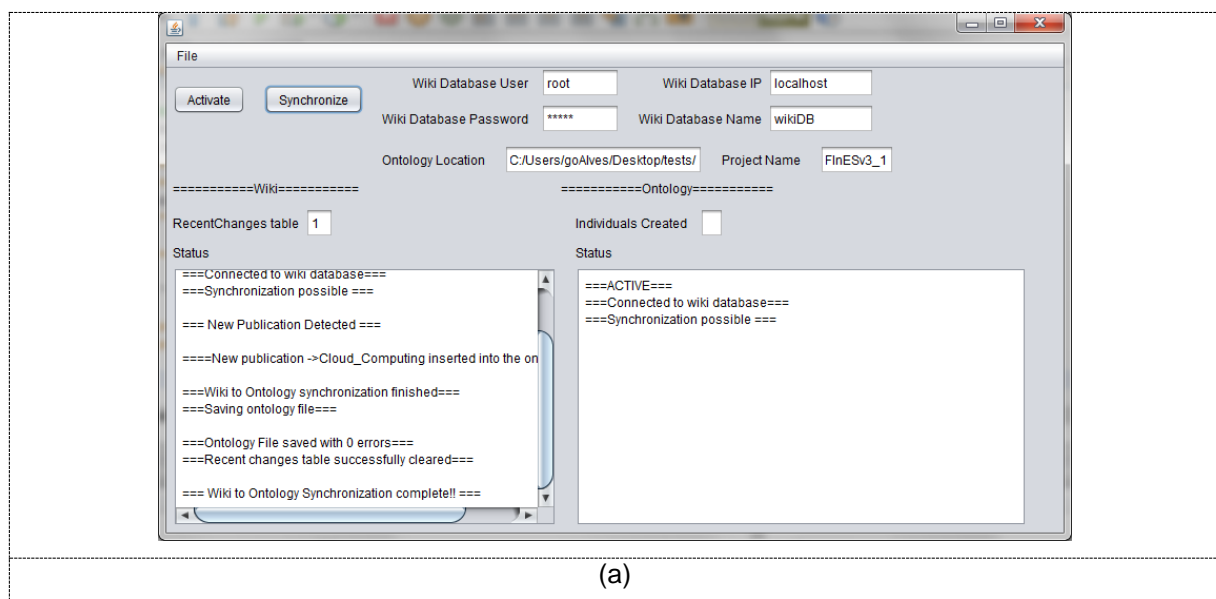
Licencing
Free

Link to Mendeley
<http://www.mendeley.com/research/cloud-computing/>

Concepts & Positions	X
Methods	X
Proof of Concepts	X
Tools	-
Experiments	-
Case Studies	-
Surveys- Empirical Data	-
Standards	-

Figure 6.11 - Wiki to Ontology synchronization example - Publication to be synchronized

Then the tool creates a publication instance in the ontology and fills the respective properties with the previously broken down sections of the wiki page. Finally, the tool saves the ontology with the new publication and the synchronization process is finished, with the results being shown in Figure 6.12. Part (a) of that same figure, shows the result of the finished synchronization process in the developed tool. Part (b) illustrates the created instance in the ontology, viewed here in the Protégé editor.



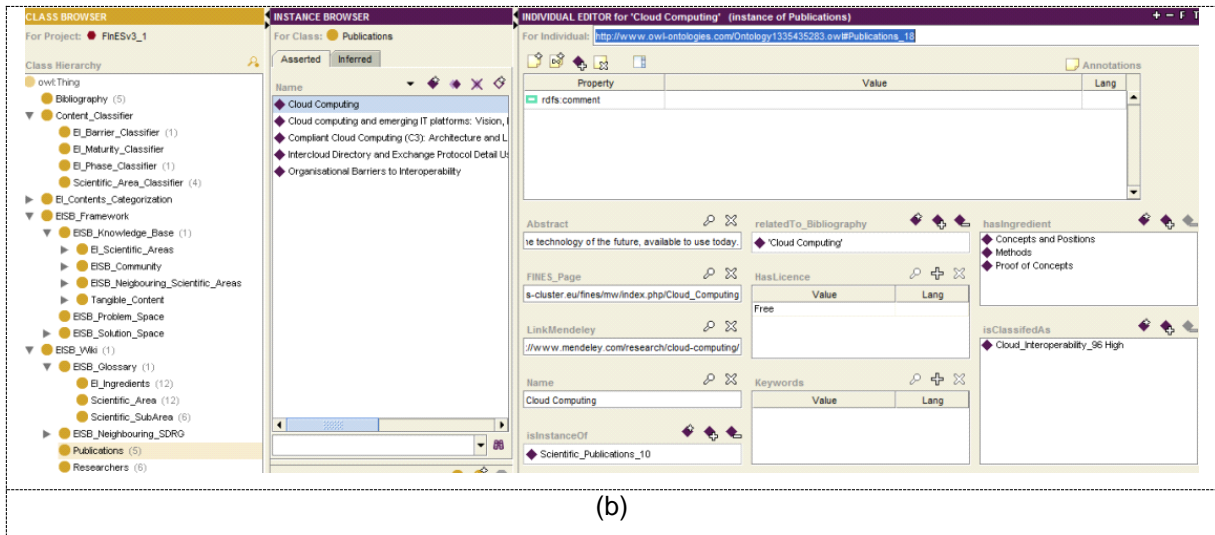


Figure 6.12 – Finished wiki to ontology synchronization process: (a) - java GUI; (b) Created instance

In conclusion, the top part of Figure 6.13 features the wiki page with its various sections and contents whereas the bottom part features the version of the same publication represented in the ontology (Protégé interface). The various sections of the wiki page have a direct correspondence in the ontology, and all of the contents are therefore well migrated.

An excerpt of the java code used to perform this synchronization task is shown in appendix 9.3.

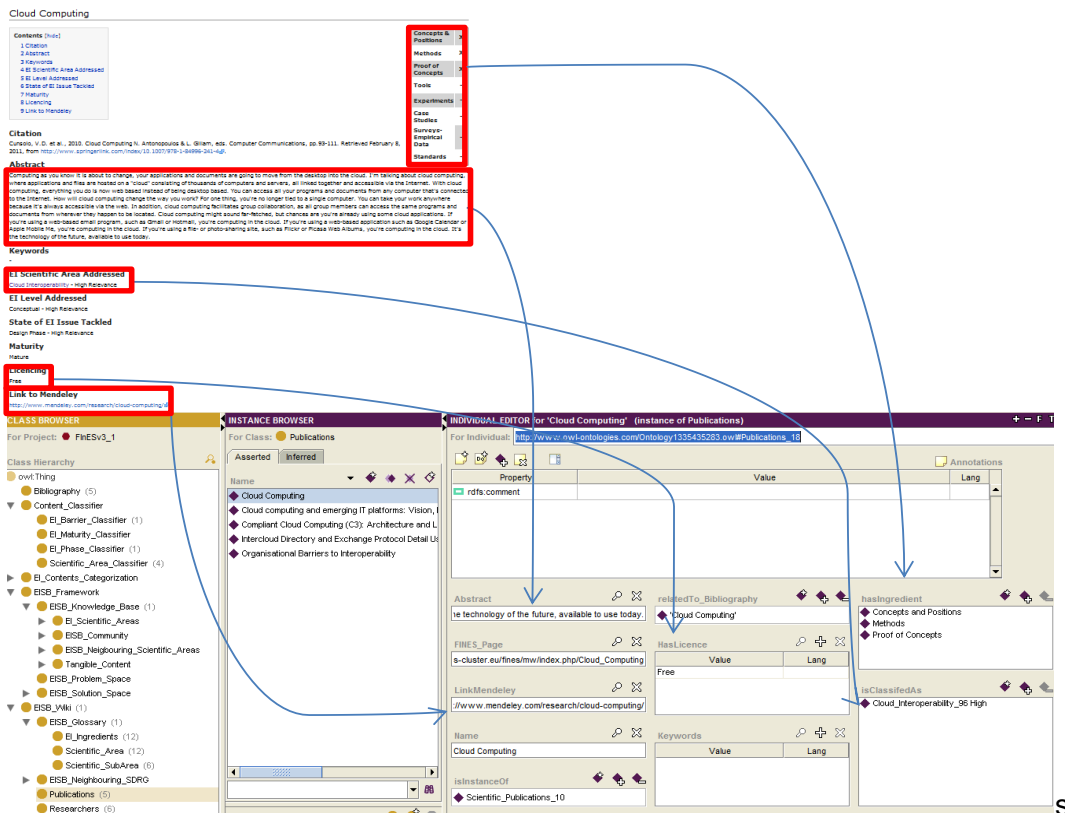


Figure 6.13 - Wiki to Ontology new publication synchronization example

6.2.2. Edit Scientific Area

In this example, the “Cloud Computing” publication was created on the wiki and then the synchronization tool was run to perform the synchronization with the ontology.

As with previous examples the users start by executing the synchronization tool and performing the required initializations. Then the users press the synchronization button to begin the process. Once again the application starts by checking the “recentchanges” table of the wiki DB and retrieves any changes that may have occurred in the wiki since the synchronization tool was last executed. In this particular example, the synchronization tool detects that a scientific area page was modified (edited), as illustrated in Figure 6.14.

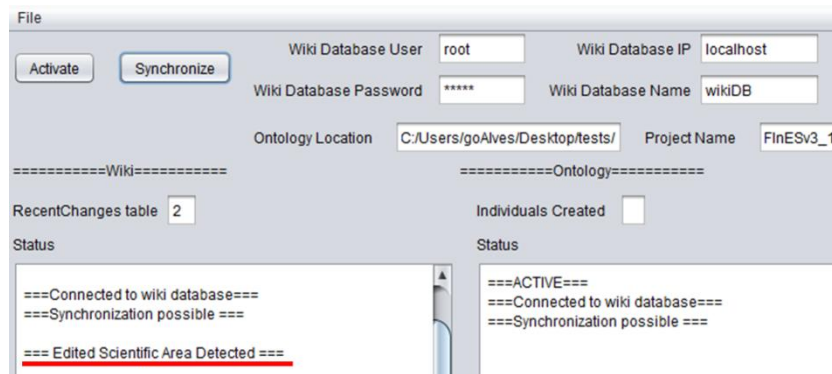


Figure 6.14 - Wiki to Ontology Synchronization example - Edited Scientific area detection

In this example, the edited scientific area is the “Social Networks Interoperability” area. Half (a) illustrates a fragment of the page before editing, while part (b) shows some the scientific area wiki page after editing. The edited items are circled for a better visualization. Subsequently, the synchronization tool breaks down each section of the edited page.

Social Networks Interoperability

A social network, according to Wikipedia^[1] "... is a social structure made up of individuals (or organizations) called "nodes", which are tied (connected) by one or more specific types of interdependency..." Recent studies indicate that people are increasingly engaged with social technologies. Over half of all Internet users are "content creators" who create websites or blogs, share original media such as photos and videos, or remix content into new creations. Social Networks such as [Twitter](#) or [Facebook](#) often are a cornerstone of this information space.

Social networks are a major evolution coming out of Web 2.0 and in the last years they have attracted the interest of enterprises, not only for promotional and advertising reasons, but also for operational purposes, ranging from developing networks of their employees, collaborators and clients, to delivering services through specific features and functions of popular social networks. Not long ago, enterprises have realised that the power of social networks can also positively or negatively impact their business operations. The impact and are also underlined in JRC-IPTS report^[2], which amongst others states that Social Computing is now mainstream that companies and policymakers cannot afford to overlook, has the potential to reshape work, health and learning and is a driver for growth and employment.

Keeping aside the dimension of reputation management in social media (which is by its own a huge research landscape), enterprises steadily understand the value behind social networks integration to their operation and try to exploit the advantages offered by these networks in terms of collaboration between themselves and also between their own employees. As^[3] states "Social networks provide means for enterprises to capture and expose many informal connections between their stakeholders." Moreover as Garner states "Within the enterprise, social computing facilitates many interpersonal functions with business implications, such as internal teaming, problem solving, collaboration, and knowledge management and transfer. Such interactions lie at the core of meeting growing business demands to improve communications, enhance collaboration and encourage innovation throughout the organization." As a result, companies that are active in Web 2.0 environments are able to exchange client portfolios, to align their online social media strategies for achieving a joint cause, to interconnect their departments and employees through these networks, etc. However, seamless interoperability between different entities in this domain does not solely mean just the presence and the usage of social networks, but goes further down to the construction of an internal, flexible organisational structure that is aligned with the philosophy and the fundamental aspects of social networks. A great number of APIs offered by social networks exist today, which offer the possibility to enterprises not only to interconnect in a seamless way to these networks, but also to build their products and services on their platforms. In fact, every one of those networks offers today its own API (like [Facebook](#), [Youtube](#), [Twitter](#), [MySpace](#), [LinkedIn](#), [Microsoft Live](#) and [Google OpenSocial](#)), which includes a number of functions and calls to the underlying software framework and as such those networks are regarded as application platforms, closely related with the idea of cloud computing.

SA.9 - Social Networks Interoperability	
Description	The ability of enterprises to utilise social networks for collaboration and interconnection purposes, by aligning part of their internal structure and functions to the characteristics of the social networks.
BackLinks	<ul style="list-style-type: none"> ▶ Data Interoperability ▶ Cultural Interoperability
Hierarchical Links	<ul style="list-style-type: none"> ▶ Outbound Links ▶ Ecosystems Interoperability
Indicative Scientific Sub-Areas	<ul style="list-style-type: none"> ▶ Social Network Characteristics Infusion
Tags	Social Networks, Social Media, Web 2.0, Social Computing, Enterprise 2.0

(a)

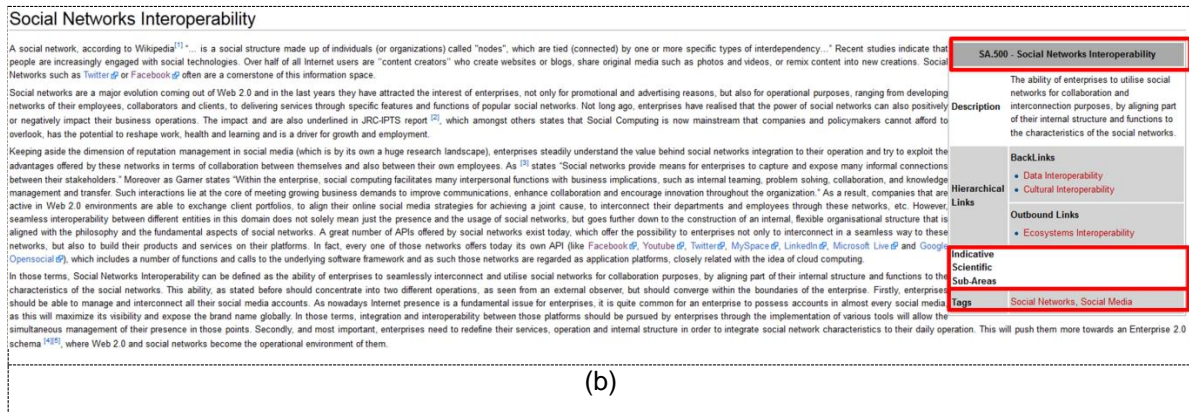


Figure 6.15 - Scientific area page - (a) Before editing; (b) After editing

Since this is merely a modification of page contents, it is assumed that an instance corresponding to the wiki page already exists in the ontology. That being said, the synchronization tool then proceeds to retrieving the instance associated with the “Social Networks Interoperability” scientific area and resets its properties to the new values, gotten from the previously broken down sections of the page. Finally, the tool saves the ontology with the edited scientific area and the synchronization process is finished. The results of this synchronization process are shown in part (b) of Figure 6.16, while part (a) illustrates the scientific area instance before the modifications, and part (c) indicates the finished synchronization process in the java interface.

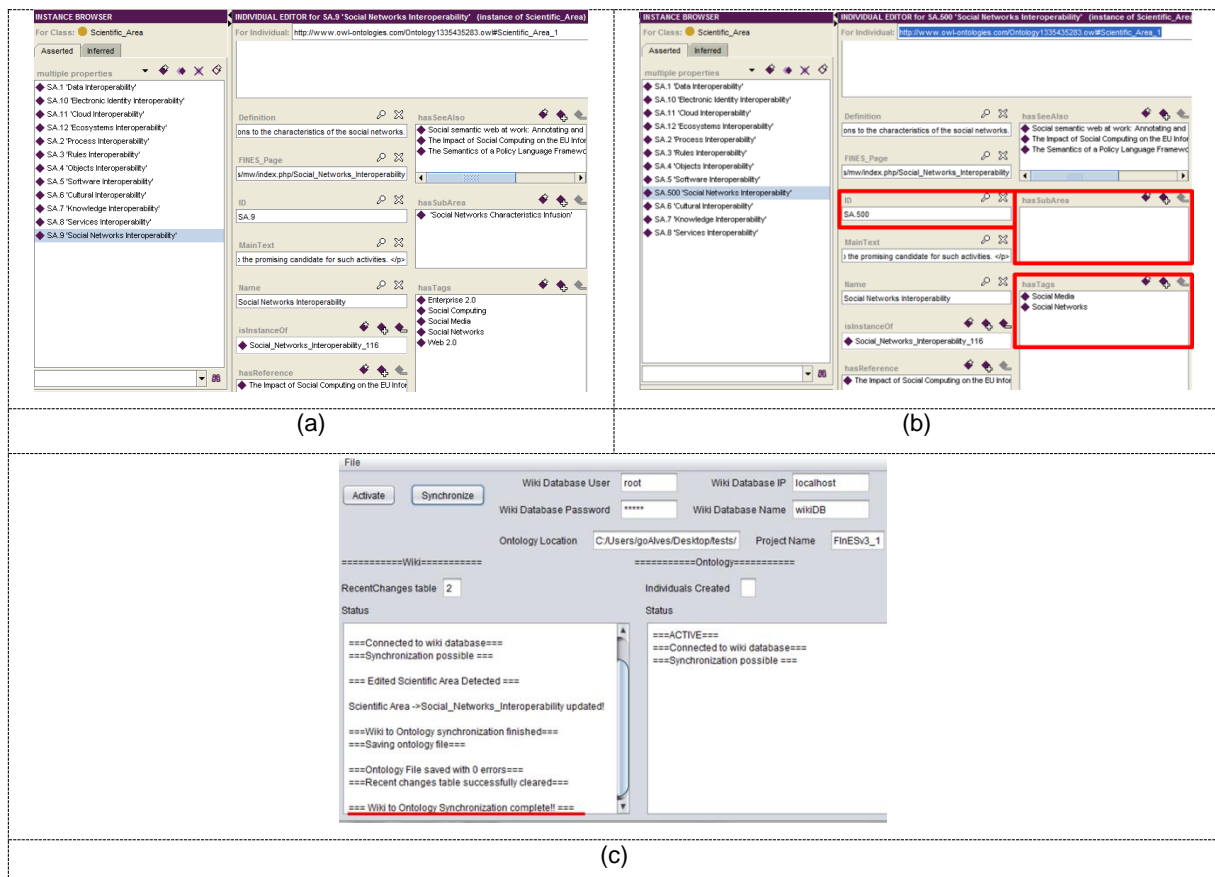


Figure 6.16 - Edited Scientific Area Synchronization - (a) Instance before editing; (b) Instance after editing; (c) Finished process - Java GUI

In conclusion, Figure 6.17 features the edited wiki page and ontology instance with the correspondence of the modified sections in each one, ensuring that all of the contents are therefore well migrated.

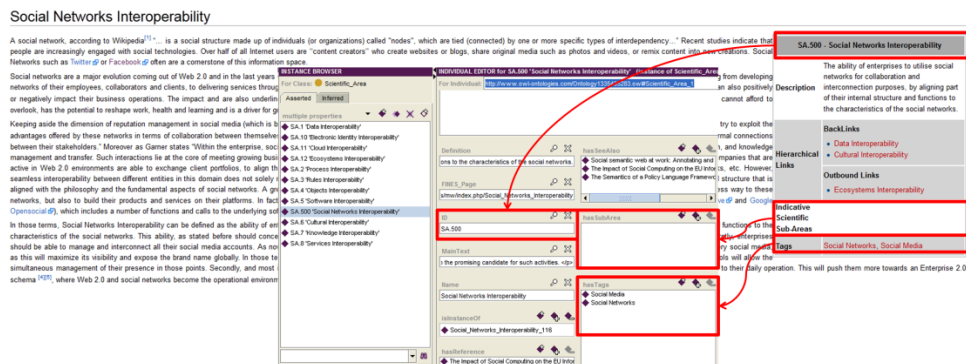


Figure 6.17 - Wiki to Ontology Edited Scientific Area example

Similarly to the previous examples, a part of the java code used in the implementation of this synchronization process is presented in appendix 9.4.

6.3. Synchronization Tool Demonstration Concluding Remarks

Regarding the synchronization process between the EISB reference ontology and FInES wiki it is important to have semantic checking because it ensures that, as both systems evolve, the data represented in them remains consistent. This was demonstrated in this chapter by presenting some use case examples of the synchronization process, showing that the synchronization was successful and that the data remained consistent and well represented in both systems.

This chapter also serves to reiterate the idea expressed when the proposed framework was presented in section 3.4. The idea is that in complex systems like this one, the prior knowledge of the synchronization method facilitates the semantic checking process. This became apparent in these demonstrations, because the knowledge represented in both elements was properly aligned, therefore allowing the modifications on one side to be properly reflected in the other.

7. CONCLUSIONS AND FUTURE WORK

Today's demanding world is inciting small enterprises to think of new ways to do business in order to survive and keep up with market requirements. Such enterprises started to realize that in order to grow they needed to target a larger market to reap more benefits. To achieve this goal, enterprises must seek collaboration with one another in order to be able to compete with the larger enterprises that dominate the bigger markets. However, collaboration does not come easy since there is usually a price to pay and some enterprises are reluctant to cooperate since they feel they have to change their way of doing business. Regardless, interoperability is key in today's world and should be seen as an opportunity instead of a barrier.

To achieve interoperability, enterprises need to communicate and collaborate with each other in order to achieve a common understanding. However, it is often the case that these communications are unsuccessful due to semantic interoperability issues.

The proposed framework was developed with the idea to provide general solutions to various contexts and situations, allowing organizations to effectively assess if their KREs are consistent, specifically, on a semantic level. Following its guidelines it was possible to assess the semantic consistency of the involved ontologies on a small case study scenario that comprises a bolt retailer and a manufacturer. The framework also enables companies to evaluate if there are losses in the information exchanges that occur between the knowledge elements. In addition, the framework indicated a possible solution through a reasoning process, more specifically using the HermiT and Pellet reasoners, to assess the conceptual consistency of ontologies. Furthermore, this framework can also be used for enterprises to evaluate the consistency of their own KREs before attempting to communicate with others. Concerning the structural point of view of the semantic checking issue, a prototype was developed for an ad hoc synchronization mechanism for multiple ontologies under the ENSEMBLE project work, between a wiki and an ontology. This prototype for a synchronization mechanism demonstrated that it is possible to maintain the structural consistency of the involved KREs, by seamlessly exchanging data from one system to another without tampering with their architectures.

In conclusion, the proposed framework could prove to be a valuable asset in helping, as a guideline, in the semantic checking of knowledge repositories.

7.1. Research Validation

To accomplish the research validation of this work it was followed a research method presented in section 1.3. Aligned to this is the research question presented in the beginning of this dissertation, and in response, it was verified that it is possible to check the semantic consistency of data exchanges between enterprises information systems by resorting to the guidelines provided by the proposed framework. The understanding between the systems can be preserved, thus maintaining semantic interoperability. This was demonstrated along the course of this document, specifically in the scenario

concerning the data exchange between a client system and a bolt manufacturer and retailer systems in section 4.1.5.1. With this situation it was possible to demonstrate the capability of the framework to help detect conceptual inconsistencies between the different KREs.

Also regarding the research question presented in the beginning of this dissertation, the demonstration of a synchronization process in section 6 helped validate one of the guidelines proposed in the framework, namely in items 5 and 6, where multiple KREs are involved. This scenario contributed to demonstrate that knowing the synchronization process indeed facilitates the maintenance of the semantic checking process. Through the demonstrated examples it was shown that this knowledge ensured that the contents between the elements of the system remained well aligned and consistent.

With both these scenarios, it can be established that both, reasoning and synchronization processes, when used separately or together, are extremely important when validating and maintaining the semantic consistency of data exchanges between the enterprises information systems.

Regarding the research question presented in the beginning of this dissertation, it was verified that it is possible to check the semantic consistency of data exchanges between enterprises information systems by resorting to the guidelines provided by the proposed framework. The understanding between the systems can be preserved, thus maintaining semantic interoperability.

For intentional purposes of the research results of this dissertation, a scientific publication was published in the proceedings of the Fifth Interop-Vlab.It Workshop on the 28th of September 2012 in Rome – Italy:

- Alves, G., Sarraipa J., Silva, J. P. M. and Jardim-Gonçalves R. A Framework for Semantic Checking of Information Systems, Accepted In: Fifth Interop-Vlab.It Workshop, 28th of September 2012 in Rome, Italy (2012).

7.2. Future Work

The main purpose behind the developed solution is to have seamless synchronization between knowledge representation systems, and in order to fulfil that goal all cases that can be identified need to be implemented. Therefore, in terms of future work, more features of the prototype can be implemented such as, the cases of “new classes”, “edit properties”, etc... (Table 6.1) or the “Remove publication”, “Edit Ingredient”, etc... features (Table 6.2).

On a different note, validation scenarios for items 3 and 4 of the framework, regarding composite ontologies could be devised.

Yet another topic of future work regarding the proposed framework is to test its items with more scenarios to further demonstrate its effectiveness.

8. REFERENCES

- [1] Silva, J. P. M , Cavaco F., Sarraipa, J. and Jardim-Gonçalves, R. (2011). Knowledge Based Methodology Supporting Interoperability Increase in Manufacture Domain. Proceedings of the ASME Congress 2011, November 11-17, Denver, CO, USA.
- [2] Sarraipa J., Jardim-Gonçalves,R., Gaspar, T. and Steiger-Garção, A. (2010). Collaborative Ontology Building using Qualitative Information Collection Methods. International Conference on Intelligent Systems, IEEE. Jul 7-9, London, United Kingdom, (2010).
- [3] The Free Dictionary (2010). Framework Meaning. Retrieved from the web at May 2012: <http://www.thefreedictionary.com/framework>.
- [4] Work Package – A4.2 Participants (2007). Athena Deliverable Number: D.A4.2: Specification of Interoperability Framework and Profiles, Guidelines and Best Practices – version 1.0; March, 2007.
- [5] Charalabidis, Y; Gionis, G; Hermann, K; Martinez, K.: Enterprise Interoperability: Research Roadmap. Update Version 5.0 (2008).
- [6] Camarinha-Matos L. (2010). Scientific Research Methodologies and Techniques - Unit 2: Scientific Method, PhD Program in Electrical and Computer Engineering (2010).
- [7] Grimm, S., Hitzler, P. and Abecker, A. (2007). Knowledge Representation and Ontologies – Logic, Ontologies and Semantic Web Languages. University of Karlsruhe, Germany (2007).
- [8] Sarraipa, J. (2004). Uma solução para a Interoperabilidade Semântica em ambientes globais de negócios. Universidade Nova de Lisboa, Faculdade de Ciências e Tecnologia.
- [9] de Bruijn, J., Ehrig, M., Feier, C., Martín-Recuerda, F., Scharffe, F. and Weiten, M. (2006). Ontology mediation, merging and aligning.
- [10] Noy, N. and Musen, M. An Algorithm for Merging and Aligning Ontologies: Automation and Tool Support. Stanford Medical Informatics, Stanford University.
- [11] Shamsfard, M. and Barforoush, A. The State of the Art in Ontology Learning: A Framework for Comparison. Intelligent Systems Laboratory, Computer Engineering Department, Amir Kabir University of Technology.
- [12] Cimiano, P., Madche, A., Staab, S. and Volker, J. Ontology Learning. Handbook on Ontologies Second Edition, International Handbooks on Information Systems pp 245-268.
- [13] Sarraipa, J. Semantic Adaptability for the Systems Interoperability. PhD Dissertation in Electrical and Computer Engineering, New University of Lisbon, Science and Technology Campus, to be presented in 2012.
- [14] Alfaries, A., Bell, D. and Lycett, M. Ontology Learning for Semantic Web Services. School of Information Systems, Computing and Mathematics, Brunel University, Uxbridge, United Kingdom.
- [15] Youn, S., Arora, A., Chandrasekhar, P., Jayanty, P., Mestry, A. and Sethi, S. Survey about Ontology Development Tools for Ontology-based Knowledge Management. University of South California.
- [16] Stanford Center for Biomedical Informatics Research (2012). Protégé Overview, available from <http://protege.stanford.edu/overview/index.html>. Accessed on February 2012.
- [17] About Ontopia, available from <http://www.ontopia.net/page.jsp?id=about>, accessed on February 2012.
- [18] Pepper, S. (2000). The TAO of Topic Maps – Finding the Way in the Age of Infoglut, available from <http://www.ontopia.net/topicmaps/materials/tao.html#d0e140>, accessed on September 2012.
- [19] Ontopia The editor, available from <http://www.ontopia.net/page.jsp?id=ontopoly>, accessed on February 2010.
- [20] Ontopia The browser, available from <http://www.ontopia.net/page.jsp?id=omnigator>, accessed on February 2012.
- [21] Dicheva, D. and Dichev, C. TM4L: Creating and Browsing Educational Topic Maps. Winston-Salem Stat University, Computer Science Department.
- [22] Katifori, A., Halatsis, C., Lepouras, G., Vassilakis, C. and Giannopoulou, E. (2007). Ontology Visualization Methods – A Survey. ACM Computing Surveys, Volume 39, Number 4, Article 10, October 2007.
- [23] FinES Wiki, available from http://www.fines-cluster.eu/fines/mw/index.php/Main_Page, accessed on February 2012.
- [24] Ontopia Graphical Visualization, available from <http://www.ontopia.net/page.jsp?id=vizigator>, accessed on February 2012.

- [25] Storey, M., Musen, M., Silva, J., Best, C., Ernst, N., Ferguson, R. and Noy, N. Jambalaya: Interactive Visualization to enhance ontology authoring and acquisition in Protégé.
- [26] Protégé Wiki – OntoGraf (2011), available from <http://protegewiki.stanford.edu/wiki/OntoGraf>, accessed on February 2012.
- [27] DebateGraph, <http://debategraph.org/Stream.aspx?nid=61932&iv=05>, accessed on April 2012.
- [28] theBrain, <http://webbrain.com/about;jsessionid=83856289256DA49C2185DB5A05F725F7>, accessed on April 2012.
- [29] Xmind3 – Brainstorming and Mind Mapping – Google Project Hosting, available from <http://code.google.com/p/xmind3/>, accessed on September 2012.
- [30] Features – Xmind: Professional & Powerful Mind Mapping Software, available from <http://www.xmind.net/features/>, accessed on September 2012.
- [31] Reasoners, available from <http://owlapi.sourceforge.net/reasoners.html>, accessed on September 2012.
- [32] Motik, B., Shearer, R. and Horrocks, I. (2009). Hypertableau Reasoning for Description Logics. Published in Journal of Artificial Intelligence Research, Volume 36 Issue 1, September 2009, pp 165-228, Oxford, United Kingdom.
- [33] Gardiner, T., Tsarkov, D. and Horrocks, I. (2006). Framework for an Automated Comparison of Description Logic Reasoners. Proceedings of the 2006 International Semantic Web Conference (ISWC 2006). November 5 – 9, Athens, United States of America (2006).
- [34] University of Oxford, Information Systems Group (2012). Hermit OWL Reasoner Overview, available from <http://www.hermit-reasoner.com>. Accessed on February 2012.
- [35] Clark & Parsia (2012). Pellet Reasoner, available from <http://clarkparsia.com/pellet/protege/>. Accessed on February 2012.
- [36] FaCT++, available from <http://semanticweb.org/wiki/FaCT%2B%2B>, accessed on September 2012.
- [37] Racer Systems GmbH & Co. KG (2012). RacerPro, available from <http://www.racer-systems.com/products/racerpro/index.phtml>, accessed on September 2012.
- [38] FaCT++, available from <http://owl.cs.manchester.ac.uk/fact++/>, accessed on September 2012.
- [39] University of Oxford, Information Systems Group (2012). Hermit OWL Reasoner – Using Hermit, available from <http://www.hermit-reasoner.com/using.html>. Accessed on February 2012.
- [40] Clark & Parsia (2012). Pellet Features, available from <http://clarkparsia.com/pellet/features>, accessed on September 2012.
- [41] Racer Systems GmbH & Co. KG (2012). RacerPro Features, available from <http://www.racer-systems.com/products/racerpro/features.phtml>, accessed on September 2012.
- [42] Institute of Electrical and Electronics Engineers (IEEE) Standards Glossary, available from http://www.ieee.org/education_careers/education/standards/standards_glossary.html, accessed on February 2012. Carnegie Mellon University.
- [43] Kasunic, M. (2001). Measuring Systems Interoperability: Challenges and Opportunities.
- [44] Morris, E., Levine, L., Meyers, C., Place, P. and Plakosh, D. (2004). System of Systems Interoperability (SOSI): Final Report. Carnegie Mellon Software Engineering Institute, Pittsburgh, PA, USA (2004).
- [45] Chen, D. (2006): Framework for Enterprise Interoperability, IFAC TC5.3 workshop EI2N (2006), Bordeaux, France.
- [46] Li, D., Huang, L. and Li, M. (2004). Dynamic Semantic Consistency Checking of Multiple Collaborative Ontologies in Knowledge Management System. Proceedings of the 5th international conference on Parallel and Distributed Computing: applications and Technologies (PDCAT), December 8-10, Singapore, pp. 76-80, 2004.
- [47] Jardim-Gonçalves, R. (2012). Arquitetura de Integração de Sistemas – Aula 1, MsC program in Electrical and Computer Engineering (2012).
- [48] Park, J. and Ram, S. (2004). Information Systems Interoperability: What Lies Beneath? ACM Transactions on Information Systems, Volume 22, Number 4, pp. 595-632.
- [49] Sarraipa, J., Jardim-Gonçalves, R. and Steiger-Garcia, A. (2010). MENTOR: An enabler for interoperable intelligent systems. International Journal of General Systems, Volume 39, Number 5, July 2010, pp. 557-573.
- [50] Gaspar, T. (2011). Methodology for Collaborative Enterprise Reference Ontology Building.
- [51] Agostinho, C., Sarraipa, J., Gonçalves, D. and Jardim-Gonçalves, R. Tuple-based semantic and structural mapping for a sustainable interoperability. Proceedings of: Technological Innovation for Sustainability - Second IFIP WG 5.5/SOCOLNET Doctoral Conference on Computing, Electrical and Industrial Systems, DoCEIS 2011, Costa de Caparica, Portugal, February 21-23, 2011.
- [52] ISOFIN

- [53] Sarraipa, J. and Jardim-Gonçalves, R. (2011). Knowledge-based System for Semantics Adaptability of Enterprises Information Systems. Proceedings of IWEI 2011 Third International IFIP Working Conference “Interoperability and Future Internet for Next-Generation Enterprises”, March 22-24, Stockholm, Sweden, 2011.
- [54] Oxford Online Dictionary, Consistency Definition, available from <http://oxforddictionaries.com/definition/consistency>, accessed on February 2012.
- [55] Haase, P. and Stojanovic, L. (2005). Consistent Evolution of OWL Ontologies. Proceedings of the 2nd European Semantic Web Conference (ESWC), May 29 – 1 June, Heraklion, Greece (2005).
- [56] Haase, P., Harmelen, F., Huang, Z., Stuckenschmidt, H. and Sure, Y. (2005). A Framework for Handling Inconsistency in Changing Ontologies. Proceedings of the 4th International Semantic Web Conference (ISWC), November 6-10, Galway, Ireland, pp. 353-367, (2005).
- [57] Baclawski, K., Kokar, M., Waldinger, R. and Kogut, P. (2002). Consistency Checking of Semantic Web Ontologies.
- [58] CEN Workshop Agreement (2012). Testing Framework for Global eBusiness Interoperability Test Beds (GITB). European Committee for Standardization., February 2012.
- [59] Bergengruen, O., Fischer, F., Namli, T., Rings, T., Schulz, S., Serazio, L. and Vassiliou-Gioles, T. (2010). Ensuring Interoperability with Automated Interoperability Testing. White Paper, European Telecommunications Standards Institute (ETSI), Sophia-Antipolis, France, 2010.
- [60] Jardim-Gonçalves, R., Agostinho, C. and Steiger-Garcao, A. (2010). Sustainable Systems’ Interoperability: A reference model for seamless networked business. Proceedings of the 2010 IEEE International Conference on Systems Man and Cybernetics (SMC), October 10-13, Istanbul, Turkey, 2010.
- [61] Agostinho, C., Gonçalves, R., Sarraipa, J., Koussouris, S., Mouzakitis, S., Lampathaki, F., Charalabidis, Y., Popplewell, K. And Assogna, P. (2011). ENSEMBLE Deliverable D2.3 EISB Basic Elements Report.
- [62] Gosling, J., Joy, B., Steele, G. and Bracha, G. (2005). Introduction. In: *The Java™ Language Specification*. 3rd ed. Addison-Wesley. pp. 1-5, 2005.
- [63] About MySQL, available from <http://www.mysql.com/about/>, accessed on August 2012.
- [64] MySQL Connectors, available from <http://www.mysql.com/products/connector/>, accessed on August 2012.
- [65] Protégé-OWL API Programmer’s Guide, available from http://protegewiki.stanford.edu/wiki/ProtegeOWL_API_Programmers_Guide, accessed on August 2012.
- [66] Accessing the collaboration features programmatically (The Changes and Annotations API), available from http://protegewiki.stanford.edu/wiki/ChAO_API, accessed on September 2012.
- [67] Manual: Page Table, available from http://www.mediawiki.org/wiki/Manual:Page_table, accessed on September 2012.
- [68] Manual: Text Table, available from http://www.mediawiki.org/wiki/Manual:Text_table, accessed on September 2012.
- [69] Manual: Revision Table, available from http://www.mediawiki.org/wiki/Manual:Revision_table, accessed on September 2012.

9.1. Ontology to Wiki Synchronization – New Scientific Area instance code example

```
private void createScientificArea(){
    String title = getDataFromMap(saDataMap, "Name");
    String table = "{{Io Scientific Area Metadata\n|SA Code= " +
    getDataFromMap(saDataMap, "ID") + "|Title= " + title + "|Description= " +
    getDataFromMap(saDataMap, "Definition") +
    "|Backlinks=</p><p>|OutboundLinks=</p><p>|Indicative Scientific Sub-Areas=</p><p>\n" +
    getDataFromMap(saDataMap, "subAreas") + "|Tags = " + buildTags(saDataMap) + "}}\n\n";
    String text = getDataFromMap(saDataMap, "MainText");
    String references = "\n== References ==\n<p
align=\"justify\"><references/> </p>";
    String seeAlso = "\n== See Also ==\n" + getDataFromMap(saDataMap,
    "SeeAlso");
    title = title.replace(' ', '_').replace('\n', ' ').trim();
    title = title.substring(0, 1).toUpperCase() + title.substring(1);
    //Capitalize first letter of title
    String category = "[[Category:" + title + "]] [[Category:EISB Glossary]]";
    String wikiText = table + text + references + seeAlso + category;

    if (db.insertPage(title, wikiText.length()))
    {
        if (db.insertText(wikiText))
        {
            if (db.insertRevision(title, wikiText.length()))
            {
                root.setOntoStatus("\n=== New Scientific Area created on the
wiki ===\n" + title + "\n");
            }
        }
    }
}
```

9.2. Ontology to Wiki Synchronization – Scientific Area class removal code example

```
private void deleteInstances(String name) {
    RDFProperty rdfProperty = owlModel.getRDFProperty("Name");
    Collection results =
    owlModel.getRDFResourcesWithPropertyValue(rdfProperty, name);
    for (Iterator it = results.iterator(); it.hasNext();) {
        Object obj = it.next();
        if (obj instanceof RDFIndividual) {
            RDFIndividual ind = (RDFIndividual) obj;
            ind.delete();
        }
    }
}
```

9.3. Wiki to Ontology Synchronization – New Publication code example

```
private void createPublicationInstance(String citation, String link, String abstr,
String wikiURL, String mendeley, String title, ArrayList<String> keywordArray,
ArrayList<RDFIndividual> ingredients, String sa, String saRelevance, String phase,
String phaseRelevance, String level, String levelRelevance, String maturity, String
licence){

    ArrayList classifierList = new ArrayList();
    getClassifier(sa, saRelevance, "Scientific_Area_Classifier",
classifierList);
    getClassifier(phase, phaseRelevance, "EI_Phase_Classifier",
classifierList);
    getClassifier(level, levelRelevance, "EI_Barrier_Classifier",
classifierList);
    getClassifier(maturity, "", "EI_Maturity_Classifier", classifierList);

    RDFSNamedClass bibClass = owlModel.getRDFSNamedClass("Bibliography");
    RDFResource newBibliography = bibClass.createInstance(title + "_BIB");

newBibliography.setPropertyValue(owlModel.getOWLDatatypeProperty("Citation"),
citation);
    newBibliography.setPropertyValue(owlModel.getOWLDatatypeProperty("Link"),
link);

    RDFSNamedClass pubClass = owlModel.getRDFSNamedClass("Publications");
    RDFResource newPublication = pubClass.createInstance(title);
    //DATATYPE PROPERTIES

newPublication.setPropertyValue(owlModel.getOWLDatatypeProperty("Abstract"), abstr);

newPublication.setPropertyValue(owlModel.getOWLDatatypeProperty("FINES_Page"),
wikiURL);

newPublication.setPropertyValue(owlModel.getOWLDatatypeProperty("LinkMendeley"),
mendeley);
    newPublication.setPropertyValue(owlModel.getOWLDatatypeProperty("Name"),
title);

newPublication.setPropertyValues(owlModel.getOWLDatatypeProperty("Keywords"),
keywordArray);

newPublication.setPropertyValue(owlModel.getOWLDatatypeProperty("HasLicence"),
licence);

    //OBJECT PROPERTIES

newPublication.setPropertyValues(owlModel.getOWLObjectProperty("hasIngredient"),
ingredients);

newPublication.setPropertyValues(owlModel.getOWLObjectProperty("isClassifiedAs"),
classifierList);

newPublication.setPropertyValue(owlModel.getOWLObjectProperty("relatedTo_Bibliography"
), newBibliography);

newPublication.setPropertyValues(owlModel.getOWLObjectProperty("isInstanceOf"),
owlModel.getRDFSNamedClass("Scientific_Publication").getInstances(true));

    root.setWikiStatus("====New publication ->" + title + " inserted into the
ontology====\n");
```

```
}
```

9.4. Wiki to Ontology Synchronization – Edit Scientific Area code example

```
private void editScientificArea(String wikiURL, String title, String newText, String
oldText){

    ArrayList newSubAreaList = new ArrayList();
    ArrayList newTagsList = new ArrayList();
    ArrayList newSeeAlsoList = new ArrayList();
    Collection range;

    String newCode = getComponentFromText("SA.", "|", newText);
    newCode = "SA." + newCode;
    String newDefinition = getComponentFromText("Description=", "|", newText);
    String newMainText = getComponentFromText("}}", "=", newText);
    String newSubAreaNames = getComponentFromText("Indicative Scientific Sub-
Areas", "|", newText);
    String newAllTags = getComponentFromText("Tags =", "}}", newText);
    String newAllSeeAlso = getComponentFromText("See Also ==\n", "[[Category",
newText);

    getMultipleComponents(newAllSeeAlso, newSeeAlsoList, "[[", "]]");
    getMultipleComponents(newSubAreaNames, newSubAreaList, "[[", "]]");
    getMultipleComponents(newAllTags, newTagsList, "[[", "]]");

    String oldCode = getComponentFromText("SA.", "|", oldText);
    oldCode = "SA." + oldCode;
    String oldDefinition = getComponentFromText("Description=", "|", oldText);
    String oldMainText = getComponentFromText("}}", "=", oldText);

    RDFIndividual editedSA = getInstanceFromClass(title, "Scientific_Area");
    if (editedSA != null)
    {
        if (newCode.length() != oldCode.length())
        {
            editedSA.setPropertyValue(owlModel.getOWLDatatypeProperty("ID"),
newCode);
        }
        if (newDefinition.length() != oldDefinition.length())
        {
            editedSA.setPropertyValue(owlModel.getOWLDatatypeProperty("Definition"),
newDefinition);
        }
        if (newMainText.length() != oldMainText.length())
        {
            editedSA.setPropertyValue(owlModel.getOWLDatatypeProperty("MainText"), newMainText);
        }
        range =
owlModel.getOWLObjectProperty("hasSeeAlso").getUnionRangeClasses();

        editedSA.setPropertyValues(owlModel.getOWLObjectProperty("hasSeeAlso"),
getListInstances(newSeeAlsoList, range));
        range =
owlModel.getOWLObjectProperty("hasTags").getUnionRangeClasses();
        editedSA.setPropertyValues(owlModel.getOWLObjectProperty("hasTags"),
getListInstances(newTagsList, range));
        range =
owlModel.getOWLObjectProperty("hasSubArea").getUnionRangeClasses();
```

```
editedSA.setPropertyValues(owlModel.getOWLObjectProperty("hasSubArea"),
getListInstances(newSubAreaList, range));

    root.setWikiStatus("Scientific Area ->" + title + " updated!\n");
}
else
{
    root.setWikiStatus("Error getting the edited instance from the
ontology!!!\n");
}
}
```