

Collaborative Ontology Development for Geo-Information Sharing

Reza Kalbasi Khoramdashti
November, 2009

Collaborative Ontology Development for Geo-Information Sharing

by

Reza Kalbasi Khoramdashti

Thesis submitted to the International Institute for Geo-information Science and Earth Observation in partial fulfilment of the requirements for the degree of Master of Science in Geo-information Science and Earth Observation, *Geoinformatics*:

Thesis Assessment Board

Chair: Dr. R.A. de By

External examiner: Dr. K. Janowicz

Supervisor: Dr. R.L.G. Lemmens

Second supervisor: Dr. L.G.J. Boerboom



**INTERNATIONAL INSTITUTE FOR GEO-INFORMATION SCIENCE AND EARTH OBSERVATION
ENSCHDE, THE NETHERLANDS**

Disclaimer

This document describes work undertaken as part of a programme of study at the International Institute for Geo-information Science and Earth Observation. All views and opinions expressed therein remain the sole responsibility of the author, and do not necessarily represent those of the institute.

Abstract

Nowadays geospatial communities have adopted Geo-Information sharing to take advantage of other knowledge resources while reducing the efforts and costs of projects. Three levels of heterogeneity are the main obstacles for geo-information sharing: syntactic, structural, and semantic. Being resolved by some standards such as OGC and ISO, syntactic and structural heterogeneity are not the targets of this research. In contrast, the semantic heterogeneity is considered as the main bottleneck for geo-information sharing.

Ontology based geo-information sharing (integration) has been proposed to cope with the semantic conflicts. This research has focused on the development of ontologies through an enhanced collaborative methodology. In contrast to centralized methodology, collaborative ontology development aims at the collaboration of different roles such as ontology engineer, domain expert, and end-user. In this thesis two phases have been determined for the collaborative ontology development: building an initial ontology (is called Catalyst in this research) through a centralized methodology and refining the initial ontology through the collaboration (is called COD in this research). Corresponding to these phases, collaborative ontology development is a kind of centralized methodology which a collaborative extension is added to it. To develop the catalyst ontology, different centralized methodologies are evaluated in this research. Supporting phases of maintenance and evaluation, METHONTOLOGY is adopted to build the catalyst ontology in phase one. To investigate the methodologies and the platforms of phase two, COD requirements are determined in this research. Upon these requirements, firstly a state-of-the-art COD methodology which is enhanced by Google Maps and Geotagged images is proposed. The use of Google maps and its related geotagged images increases the expressivity of geospatial concepts to be understood by different roles. Secondly, the wiki-based and non-wiki-based platforms supporting COD are evaluated for the sake of implementation.

To test the proposed methodologies and adopted platform in phases one and two, two geospatial communities in the context of ForeStClim project are considered as the case-study in this research.

The ontology-based geo-information integration is possible in two ways: ontology mapping or shared ontology development. Due to ontology mapping bottlenecks, the latter one is considered in this research. The development of the shared ontology in two different methodologies (centralized and collaborative) brings about a better result through the collaborative one.

Keywords

Geo-Information sharing, Geo-Information integration, Shared ontology, Collaborative ontology development

Acknowledgements



This thesis grew out of a series of consultations with my supervisor Dr. R.L.G. (Rob) Lemmens. Meetings and discussions with Rob helped me to overcome the complexity of this research. I really appreciate his knowledge and cooperation throughout my thesis.

It is an honor for me to have a supervisor like Dr. L.G.J. (Luc) Boerboom. I never forget his supports, enthusiasm, and encouragements in my work.

I owe my deepest gratitude to Dr. Sven Schade and his colleagues in Münster University, semantic interoperability lab. Sven steered me in the right way of ontology engineering.

I would like to thank my advisors Dr. A. Alesheikh from KNT University and Dr. H. Abolhasani from Sharif University of technology.

I am heartily thankful to JKIP coordinators Dr. B. Vosooghi and Mr. Huurneman because of their support during the course.

Many thanks to my ForeStClim project colleagues Stephen, Julien, Laurent, and Paul. They were cooperative when I was working with them in France.

From the first days of my studies, my family was supporting and encouraging me warmly. Parvaneh, Arezoo, Azadeh, Mohammad, Arian, Baran, Babak, and Amin are my family. And Khosro who lives with me and is alive forever.

Many thanks to my best friend in Enschede Mohammad Karimi. He was always supporting with his advices and experiences during my thesis. I also thank Farhang, Shirin, Babak, Aidin my Iranian friends in the Netherlands.

I would like to thank my loyal friend Mohammad Reza, my JKIP classmate Reza Najari. Also I never forget Pooyan, Reza, and Mr. Abdollah Naroui.

... And the last one but the most important one is the only Love. Negar, I dedicate this thesis to you.

Table of contents

1. About the Research	1
1.1. Motivation and Problem Statement.....	1
1.1.1. Motivation	1
1.1.2. Research Problem	4
1.2. Research Identification	5
1.2.1. Research Objectives	5
1.2.2. Research Questions.....	5
1.2.3. Innovation Aimed At.....	6
1.3. Method Adopted	6
1.4. Thesis Outline	6
2. Ontology-based Geo-Information Sharing	9
2.1. Geo-Information sharing.....	9
2.2. Ontologies for explicating context knowledge	9
2.2.1. Shared ontologies upon Source ontologies.....	10
2.3. Ontology Engineering Process.....	12
2.3.1. Centralized ontology engineering.....	14
2.3.2. Collaborative ontology engineering	15
2.4. Summary	17
3. Collaborative Ontology Engineering Phases.....	19
3.1. Phase One: Catalyst ontologies.....	19
3.1.1. Source ontology (Answer one)	20
3.1.2. Shared ontology (Answer one)	22
3.2. Phase Two: COD	22
3.2.1. COD requirements (Answer two).....	22
3.2.2. The evaluation of COD methodologies (Answer three)	24
3.2.3. The platform supporting both phases (COD and Catalyst development)	31
3.3. Users of COD.....	32
3.4. Summary	33
4. Case study: Geo-Information sharing among ForeStClim communities	35
4.1. ForeStClim: an EU project.....	36
4.2. Centralized ontologies	37
4.2.1. Step one: Developing the ONF and SERTIT catalyst ontologies.....	37
4.2.2. Step two: Developing the catalyst shared ontology.....	39

4.3.	Collaborative ontologies	41
4.3.1.	Step three: Developing the ONF and SERTIT collaborative ontologies.....	41
4.3.2.	Step four: Developing the collaborative shared ontology	44
4.4.	The difference between collaborative and centralized methodologies	45
4.5.	The Support activities	46
4.6.	Summary	46
5.	Discussions, Conclusions, and Recommendations.....	47
5.1.	Discussion.....	47
5.1.1.	Geo-information sharing, obstacles, and solutions.....	47
5.1.2.	The development of ontologies: linking collaborative to centralized.....	48
5.1.3.	Catalyst ontologies	49
5.1.4.	COD methodology.....	49
5.1.5.	Protégé.....	52
5.1.6.	The role of Google maps and geotagged images in COD	55
5.1.7.	User feedbacks.....	55
5.2.	Conclusion	56
5.3.	General Recommendations:	57
Appendix 1:	Conceptualization phase	59
Appendix 2:	Client Server Protégé	65
Appendix 3:	OWL and RDFS – the example of ForeStClim Source Code	69
Appendix 4:	Centralized ontology engineering methodologies	73
Appendix 5:	Collaborative ontology engineering methodologies.....	77
Appendix 6:	The architecture of Collaborative Protégé.....	81
Bibliography	85

List of figures

Figure 1: Different forest definitions used in some countries [7].....	2
Figure 2: Conceptualizations of person 1 and person 2; each one paints her impression of reality	3
Figure 3: The knowledge structure of expert1 about a Temperate Needleleaf forest	4
Figure 4. Single ontology approaches [4].....	11
Figure 5. Source ontologies in multiple ontologies approaches [4]	11
Figure 6. The shared ontology in hybrid approaches [4].....	11
Figure 7. Ontology engineering process [16]; the support activities are parallel to development activities.	13
Figure 8. Collaborative ontology engineering and its major phases; the figure shows that COD is like a plug-in for centralized ontology development methodologies	16
Figure 9. Four essential questions in collaborative ontology engineering; this Chapter answers to these question one by one through its Sections	19
Figure 10. The extracted common ideas in ontology engineering Lemmens [5]	20
Figure 11. COD workflow in a client-server architecture; the sessions related to discussions threads (in blue color) and decision making (in red color) are illustrated in the lower part.	28
Figure 12. The whole process of ontology-based GI integration through the centralized and collaborative methodologies; these two methodologies are illustrated in two packages (left, right) to be comparable with each other.	35
Figure 13. Integrating two different datasets stored in ONF and SERTIT spatial databases based on two different classification systems	36
Figure 14. Linking the GI layers of ESRI ArcGIS to SERTIT hierarchy; “are” label represents the subclass/superclass relationship and was understandable for the domain experts as a kind of sub-category relationship.	38
Figure 15. The Catalyst shared ontology upon the ONF and SERTIT catalyst ontologies; OWLViz plug-in of Protégé is used for this tree representation of shared ontology	40
Figure 16. A discussion thread about Other_ONF class	42
Figure 17. The use of Google maps and geotagged images in COD.....	43
Figure 18. Inconsistency checking by the Pellet reasoner in Protégé; Here the Other_ONF class is semantically inconsistent.....	43
Figure 19. The collaborative shared ontology upon the ONF and SERTIT ontologies; OWLViz plug-in in protégé is used for this tree representation of shared ontology	45
Figure 20. The problem of Jambalaya for tree visualization when the ontology has many classes	53
Figure 21. Importing main concepts by the domain experts; COE interface	59
Figure 22. Grouping the concepts due to their similarity; this was easy in this research as the main hierarchy of ontology was determined by the classifications.....	60
Figure 23. KA in COE; the domain expert can do it in a distributed setting. In this figure one of the domain experts has annotated the concept	60
Figure 24. Representation of concepts and the properties among them in SWING [51]	61
Figure 25. Two different styles of representation; due to users’ point of view, the lower one is somehow messy.....	62
Figure 26. Defining each concept by its properties in NL; although the domain experts were not aware of any property, they understood these definitions well	62
Figure 27. Matrices for storing the information about ontology relations [51].....	63
Figure 28. Importing and storing the individuals in COE	63

Figure 29. Adding web URL of related Wikipedia pages in COE increased the speed of conceptualization phase.....	64
Figure 30. Adding web URL of related Wikipedia pages and Cyc ontology in COE increased the speed of conceptualization phase	64
Figure 31. The Protégé server is started	65
Figure 32. The defined operations for each group of collaborators in COD; the snapshot has been captured from the Protégé ontology editor	67
Figure 33. The COD policies for the interaction of different roles; this shows that domain experts do not have permission to edit the ontology components; the snapshot has been captured from the Protégé ontology editor	67
Figure 34. The available projects for the collaborators; the snapshot has been captured from the Protégé ontology editor	67
Figure 35. Login to the server by defining the Host Machine Name (here we have considered the local host), username and password	68
Figure 36. Increasing the amount expressiveness and computational complexity. From left to right, the right one include the left one both semantically and syntactically	70
Figure 37. Uschold and King's methodology	73
Figure 38. Grüninger and Fox's methodology	74
Figure 39. On-To-knowledge methodology [53]	74
Figure 40. The METHONTOLOGY development process; Due to this figure, the amount of KA in conceptualization phase is higher rather than other phases [16]	75
Figure 41. The proposed collaborative ontology engineering methodology by Holsapple and Joshi [24]	77
Figure 42. Evaluation Sheet [25].....	79
Figure 43. Different components of Collaborative Protégé and their relationships together	81
Figure 44. Provided methods by Java APIs in Protégé	82
Figure 45. Search Tab in Collaborative Protégé	83

List of tables

Table 1. The evaluation of current COD methodologies with respect to the COD requirements	25
Table 2. Annotation types in COD	30
Table 3. Evaluation of COD with respect to its requirements; some of the requirements related to tools have not been discussed here	31
Table 4. The ONF and SERTIT discussion threads summary: Collaborative development of ONF and SERTIT ontologies	41
Table 5. The discussion threads summary: Collaborative development of shared ontology	44
Table 6. COD collaborators profile	66
Table 7. Collaborative Protégé operations for COD	66

List of Acronyms

AI	Artificial Intelligence
COD	Collaborative Ontology Development
CQ	Competency Question
DB	Data Base
DBMS	DataBase Management System
DE	Domain Expert
DL	Description Logic
FAO	the Food and Agriculture Organization
FOL	First-Order Logic
GI	Geo-Information
GIS	Geospatial Information System
GPS	Global Positioning System
HTML	HyperText Markup Language
IEEE	Institute of Electrical and Electronics Engineers
ISO	International Standards Organization
JRE	Java Runtime Environment
KA	Knowledge Acquisition
NL	Natural Language
OE	Ontology Engineer
OGC	Open Geospatial Consortium
OWA	Open World Assumption
OWL	Web Ontology Language
RDF	Resource Description Framework
RDFS	RDF Schema
RS	Remote Sensing
UN	the United Nations
UNA	Unique Name Assumption
URL	Uniform Resource Locator
XML	eXtensible Markup Language

Chapter One

About the Research

1.1. Motivation and Problem Statement

1.1.1. Motivation

The spurt of Geospatial Information System/Science/Service¹ [1] from the mainframe systems to distributed and dynamic GIS (Internet GIS) delineates the need for the interaction among multiple GIS components [2]. Each of these components located in different locations is like a piece of Lego which should be associated together via the interaction and cooperation. Simply, the interaction and cooperation of these communities without obstacles is called interoperability which is not achievable through heterogeneous environments. IEEE Computer Society Standards Coordinating Committee² defines interoperability as: *“The ability of two or more systems or components to exchange information and to use the information that has been exchanged.”* [3]

Besides exchanging (sharing) information is an essential task, the usability of information is important too. It should be mentioned that information sharing includes two parts: information finding and information integration [4]. The intention of this research from information sharing is information integration while assuming information has been found before. Therefore in this research the terms information sharing and information (also data) integration are using interchangeably. Because of three levels of heterogeneity, geospatial communities have difficulties to share and use information. The heterogeneity problems hindering the information sharing can occur at the following three levels:

1. The syntactic heterogeneity in two types: one is related to the logical data model and its underlying DBMS³. For instance, the relational and the Object Oriented DBMS where there should be a mapping between different schemas. The other one is related to the representation of data such raster data or vector data.
2. The structural heterogeneity (Schematic).
3. The semantic heterogeneity, which involves conflicts on the intended meaning of different resources [5]. As the semantic heterogeneity is difficult to grasp easily, this is explained better by some examples below.

¹ GIS: Geographic Information Systems (1980s), Geographic Information Science (1990s) and from 2000 until now Geographic Information Service

² Institute of Electrical and Electronics Engineers; <http://www.ieee.org>

³ Database management system

What is a “Forest”? You can find many definitions for this concept, corresponding to different communities. Lund [6] has surveyed the definitions of forest, deforestation, afforestation and reforestation and classifies them in four types (administrative, land cover, land use, ecological/miscellaneous) and four scopes (general, international, national, and local). His survey has resulted in 953 definitions until 30th April 2009. For instance, UN-FAO has defined Forest land as:

“... *A land used mainly for wood production and other forest products, protection...*”⁴.

Or the definition of European Environment Agency for the forest:

“... *a vegetation community dominated by trees and other woody shrubs, growing close enough together that the tree tops touch or overlap, creating various degrees of shade on the forest floor. It may produce benefits such as timber, recreation, wildlife habitat, etc...*”⁵.

These examples manifest various points of view and criteria to describe “Forest”, albeit with some similarities and overlaps. Among these criteria, miscellaneous definitions can be observed in different countries if only two parameters of *tree height* and *canopy cover* are considered for the forest definition [7]. These definitions can be similar or completely different from each other. This issue is illustrated in the Figure 1:

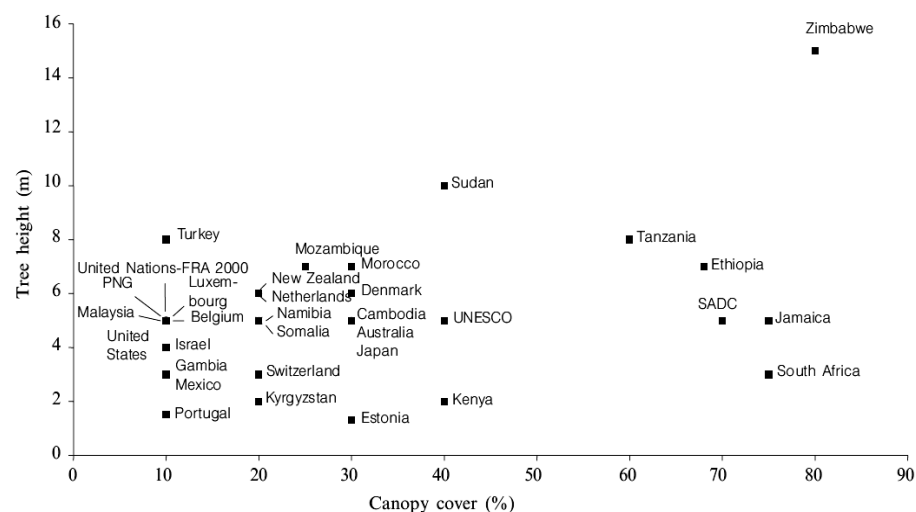


Figure 1: Different forest definitions used in some countries [7].

E.g. in Belgium and South Africa the criteria to determine forest are different, but forest has the same meaning in both Belgium and Luxemburg. Heterogeneity or the quality of being diverse and not comparable in kind in “Forest” word is a semantic one in the mentioned examples.

Standards and solutions

OGC⁶ is an international voluntary consensus standard organization that has been successful to resolve the problems of syntactic and structural heterogeneities. But this is only the first step, as the semantic heterogeneity still presents an obstacle on the way to achieve full interoperability [8].

⁴ <http://www.fao.org/DOCREP/005/Y4357E/y4357e20.htm>

⁵ European Environment Agency (EEA) <http://www.epa.gov/trs>

⁶ Open Geospatial Consortium; <http://www.opengeospatial.org/>

Tim Berners-Lee expressed his opinion about the Semantic Web in 2001 when he believed that it should be decentralized and capable to comprehend semantic data for the machines [9]. In other words, computers and communities working on the semantic web infrastructure can understand each other's messages and cooperate by resolving semantic conflicts. In semantic web, many technologies and tools help the semantic enrichment of current web contents. From the other point of view, semantic web technologies try to augment and make up the current HTML⁷ documents for a better understanding of operations and analysis on the web contents. These technologies and theories are XML/XML⁸ Schema, RDF/RDF⁹ Schema, Ontology, Logics et cetera.

Ontology is a collection of concepts and relationships among them in a special structure. The studies about ontology are not only for the semantic web exclusively but also discussed in other fields rather than web. One of these fields is Artificial Intelligence (AI) where the ontologies are used to resolve the semantic conflicts among Agents' interactions. The basis of ontology is conceptualization which is simply a kind of painting when one tries to represent some parts of the world. Paintings are the frames for depicting the complex reality and the painter adopts a number of parameters to paint her painting. For instance, two different persons after gazing on a same area of forest can contemplate and draw it disparately. In the following figure, each person's conceptualization about the same area of a forest has been depicted:



Figure 2: Conceptualizations of person 1 and person 2; each one paints her impression of reality

Each person only considers some aspects of reality and one can say that the conceptualization relates to the human point of view to extract the entities and the relationships among them. If a third person looks at these paintings, he may not interpret them as the same area. This means that they have conceptualized differently.

Let's suppose each person has some background knowledge about the forest; e.g. person 1 is also a forester and has some knowledge about the forest types. She claims that this forest is a *Temperate Needleleaf forest* which exists in higher latitude regions of the northern hemisphere, higher altitude zones and warm temperate areas.¹⁰ This knowledge will be hidden and implicit from the other people (like person 2) if they intend to get additional information about this region. The first solution to explicate the mentioned implicit knowledge is the use of natural language (NL) but the others may not understand it as their perspectives about the domain are different (e.g. they have different meaning for Forest; see Figure 1). Briefly NL is a language that has developed naturally in use, as opposed to an artificial language or computer code [10]. To cope with this problem and explicate his knowledge about the domain, person 1 has done some tasks. Firstly, constructing the structure of his knowledge (see Figure 3):

⁷ Hypertext markup language

⁸ Extensible markup language

⁹ Resource Description Framework

¹⁰ UNEP-WCMC http://www.unepwcmc.org/forest/fp_background.htm; Accessed at 24 June 2009

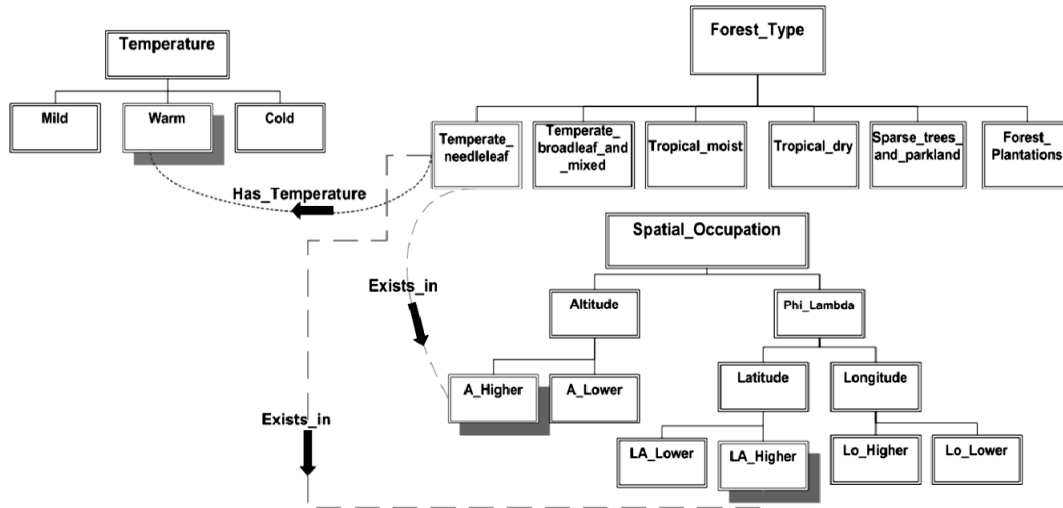


Figure 3: The knowledge structure of expert1 about a Temperate Needleleaf forest

By this structure, some of the implicit knowledge of the mentioned scenario has been explicitly stated; e.g., the relationship between the temperate needleleaf class of forests and the higher altitude spatial occupation. Also, corresponding to the existence of temperate needleleaf forest in the higher latitudes as a necessary rule, person 1 supports more semantics to this knowledge structure through some constraints; e.g. asserting the following necessary constraint for `Temperate_needleleaf` class:

- All the instances of `Temperate_needleleaf` class exist at least in one of the higher latitudes (defined before e.g. latitude > 50 degree). It can be written as an axiom via logical quantifiers such as

existential quantifier (\exists): $\exists \text{ Exists_in LA_higher}$

Through this procedure, this person could express and explicate her knowledge gradually. And if she expresses this structure and axioms in a formal language, the computers can process and conceive the semantics of background knowledge. The formal and explicit specification of a conceptualization is called ontology [11].

1.1.2. Research Problem

The process of extracting knowledge of a domain and constructing its structure is called ontology engineering (development). Most of the current ontology development processes are performed through a centralized methodology. It is the development of ontologies only by the ontology engineers (familiar with ontology tools, methodologies, reasoning and inference services and modeling) and not engaging the domain experts and the end-users so much. The role of domain experts in this methodology is the explication of domain knowledge for the engineers via conceptualization. Although the domain experts are the most important knowledge resources in ontology engineering, other knowledge resources such as databases, encyclopedias, domain thesauruses, wikis are also considered as knowledge resources. And the

role of end-user is to use the ontology-based application. This methodology's deficiency is revealed: first, when the ontology engineer cannot grasp the domain expert's knowledge well. It can have many reasons such as engineer's inaptitude or the expert's inability to describe the domain. Second, when the applicability of application has been hindered by some system faults. The best solution is to learn ontology principles by the domain experts and end-users. Then they can resolve the engineer's (or domain expert) mistake and application's fault by interacting with its ontology. This solution is entirely illogical and impossible because of interest and costs aspects. But there are some ways to engage them in the process of ontology engineering like getting feedbacks from the end-user and etc [12]. An example of end-user's participation is an ontology-based knowledge base that results in wrong conclusion and reasoning. Or in the mentioned scenario of forestry experts, it can be strange for another person how an area in the higher latitudes and higher altitudes has a warm weather. This can be because of inconsistent knowledge (expert or engineer's mistake) or the existence of some volcanic area around the temperate needleleaf forest.

1.2. Research Identification

1.2.1. Research Objectives

The super objective of this research is to propose an enhanced methodology to develop geospatial ontologies collaboratively, which supports the cooperation of geospatial communities such as geo-information sharing among them. Also the following sub-objectives related to the super objective are:

- Identifying the need for collaborative ontology engineering corresponding to the centralized methodology deficiencies
- Defining the requirements related to collaborative ontology engineering
- Adopting a technology supporting the proposed methodology
- Enhance the results of methodology via geospatial information or services
- Test the results of methodology through the development and evaluation of a shared ontology in two different methodologies: collaborative and centralized

1.2.2. Research Questions

1. How to share geo-information among geospatial communities without any obstacle?
2. What is the state-of-the-art in ontology engineering?
 - 2.1. What is centralized ontology engineering?
 - 2.2. What is collaborative ontology engineering?
3. What are the phases of collaborative ontology engineering?
 - 3.1. How to develop the catalyst ontology?
 - 3.2. What is COD?
 - 3.3. What are COD requirements?
 - 3.4. How to propose a methodology for COD?
 - 3.5. What is the proper tool fulfilling the proposed methodology?
 - What are the characteristics of wiki-based COD?
 - What are the characteristics of non-wiki-based COD?
 - 3.6. Who can be contemplated as the users of COD?

4. How to enhance COD by geospatial information and services?
5. How to test the methodology?

1.2.3. Innovation Aimed At

The novelties in this research are:

- Ontology-based geo-information sharing in a collaborative methodology
- Proposing a new methodology related to COD
- The enhancement of COD with Google maps and Geotagged images
- Integrating centralized and collaborative ontology engineering methodologies through the determination of their joint point

1.3. Method Adopted

Firstly, a solution for sharing information among different communities should be determined. This needs the evaluation of ontology-based information integration methodologies after addressing the problems related to other solutions such as schema mappings. Secondly, the methodologies to develop ontologies should be evaluated. In this step, the collaborative and centralized methodologies are considered. Because of centralized methodology deficiencies, the collaborative is adopted for the development of ontologies and its requirements will be determined. Third, corresponding to the phases of collaborative ontology engineering (two phases), a new methodology comprehending phase one and phase two will be proposed. Fourth, wiki-based and non-wiki-based ontology development technologies are evaluated by COD requirements and a proper one for the proposed methodology will be chosen for the implementation. Fifth, the proposed methodology and its related technology will be tested through the ForeStClim¹¹ Project. In this project two geospatial communities have been adopted for the case-study as they share geospatial information. In this step a shared ontology will be developed collaboratively. In a nutshell, this shared ontology is constructed for easing geo-information sharing among these communities. The shared ontology will be compared with another shared ontology among these communities. The latter shared ontology will be developed in a centralized methodology.

1.4. Thesis Outline

The whole ideas in this research have been discussed in the following Chapters:

Chapter One- The motivation, research problem, research objectives, research questions, and the adopted method to solve the research problem and to achieve the objectives are discussed.

Chapter two- The fundamentals and scientific principles of this research have been discussed. Due to the geo-information sharing, the obstacles and the solutions such as ontology-based information integration are explained. The use of shared ontologies (upon communities' source ontologies) is introduced as the main solution in this research. To build these ontologies, the collaborative and centralized ontology engineering methodologies have been discussed in this Chapter. Two general phases are determined for collaborative ontology engineering: building an initial ontology and collaboration extension. At the end of

¹¹ <http://www.forestclim.eu/> ForeStClim is a EU-funded environmental project addressing forests and climate change. The short name stands for "Transnational Forestry Management Strategies in Response to Regional Climate Change Impacts".

this Chapter, four questions have been defined which their answers will determine collaborative ontology engineering phases in detail.

Chapter three- In this Chapter by answering to the questions of Section 2.3.2, the phases of collaborative ontology engineering will be determined in detail.

Chapter four- In the context of ForeStClim project, two geospatial communities are considered for sharing geo-information. The proposed ontology engineering phases will help to share geo-information through the development of source ontologies of these communities and a shared ontology upon them. Also these ontologies will be developed through a centralized methodology to test the results of collaborative ontology engineering.

Chapter five- The test results of case-study implementation in Chapter four will result in some discussions, conclusion, and recommendation about the context of this research.

Chapter Two

Ontology-based Geo-Information Sharing

2.1. Geo-Information sharing

Nowadays, geospatial data become the key solution for many applications such as decision making, significant analysis and disaster management. After the process of spatial data acquisition, people concentrate on the use of geo-information in the next step. However, as the data acquisition is expensive and wearisome, the GI¹² communities have adopted GI sharing. In these scenarios, some of the communities have adjusted themselves to provide the GI and some to use it and the others may act as both provider and user. As each community has its own domain terminology, the sharing and integration of different terminologies become a strenuous task. One of the solutions to resolve this problem is the use of shared terminologies between the communities. Although there are many common terms among them, the bottleneck of this task is the implicit and background knowledge related to a term when it is intended to be interpreted and used by the other community. Stuckenschmidt and Van Harmelen [4] has called this kind of knowledge as “Context Knowledge” which should be explicated while sharing information. In the process of explicating and representing¹³ the context knowledge, some critical touchstones should be considered: firstly, not to represent the knowledge erroneously, and secondly, not to define the non-related knowledge (more detailed or more general).

There are several solutions for information sharing (data integration) such as database and XML Schema mappings. The use of XML helps for structuring data, but different terminologies by different communities bring about the semantic conflict between communities. If they adopt a common vocabulary for the definition of common terms, the context knowledge (stored in their databases or XML schema) will be exchanged well. Also in applications such as distributed databases systems where an administrator designs in a centralized way, no semantic heterogeneity occurs [13]. However, in reality each community has its own terminology of the domain.

2.2. Ontologies for explicating context knowledge

Studies of being, existence, the essence of things and so many points of view about ontology can be noticed which are not the exact meanings of ontology. Ontology is the translation of a Greek word

¹² Geo-Information

¹³ Knowledge Representation

(Οντολογία). Using a Greek-to-English translator¹⁴, the following remarks are considered: “Οντο” means “of Being” and “λογία” means “the reasons, the systematic study and science”. When you are studying ontology in philosophy this word “science of being” is of scientists interest like Aristotle (384-322 B.C). It is essential to mention that ontology is a branch of metaphysics related to being in this domain. However philosophy is not the only science which applies ontology. Other applications of ontologies are [14]: knowledge management, natural language processing, e-commerce, intelligent integration information, information retrieval, database design and integration, bio-informatics, education and in new emerging fields like the Semantic Web. In the field of semantic web and computer science, the following ontology definitions are noticeable: “*An ontology is an explicit and formal specification of a conceptualization*” [11]. Each word of this definition has a scientific background that helps the reader to understand ontology:

1. An ontology: although ontology is an uncountable noun, in computer science studies it is considered as a countable noun. For example the plural form of ontology is ontologies.
2. Explicit: explicating the implicit knowledge results in semantics of knowledge which is needed for the information sharing. On the other hand, defining the types of concepts and corresponding constraints for their uses [14].
3. Formal specification and degree of formality: it is rigorously related to “machine reasoning”. The lowest degree of formality is in the natural language expressions. By structuring the contents of these expressions, they become more formal which will result in hierarchies. The next steps can be contemplated as giving semantics in terms of first order logic to achieve formal ontologies [15].
4. Conceptualization: the sufficient principles of conceptualization are discussed in Section 1.1.1 through some examples. Nevertheless, a brief but complete definition is: “*an abstract model of some phenomenon in the reality by having identified the relevant concepts of that phenomenon.*”[14]

The ontologies can be used to explicate the context knowledge in the process of information sharing. The advantage of using ontology for this task is to meet the mentioned critical touchstones: firstly, not to represent the knowledge wrongly; by supporting inconsistency checking through formalizing the context knowledge. And secondly, not to define the non-related knowledge (more detailed or more general); with the help of conceptualization, it is the model of domain while concentrating on the relevant concepts of domain.

2.2.1. Shared ontologies upon Source ontologies

Many methods for ontology-based information integration amid different communities have been proposed. Among them, three main approaches are distinguished [4]: single-ontology approaches, multiple-ontology approaches and hybrid approaches. In below, these approaches are explained in brief.

Single ontology approaches- exploiting a global ontology supporting the shared vocabulary of all communities as the information sources; in this approach the information sources must be in a same domain. This approach is not flexible for the future changes in the information sources such as domain databases (see Figure 4).

¹⁴ such as an online translator: <http://translate.google.com>; Accessed at 2 July 2009

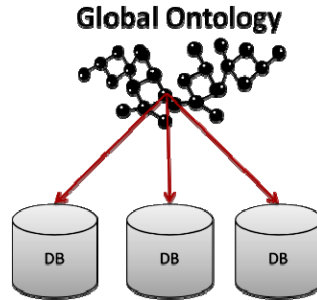


Figure 4. Single ontology approaches [4]

Multiple ontologies approaches- in these approaches, one ontology for each information source will be developed and no need for global ontology is met. From one aspect, this is a flexible approach as there is no concern about the changes in the source ontologies (and its impact on the general ontology). From the other aspect, the main bottleneck is the comparison of the source ontologies (see Figure 5). Because the terminology of source ontologies may be different from each other and a way to map these ontologies should be found. In practice, mapping of different local ontologies to spot the corresponding concepts (same or similar) is very difficult [4].

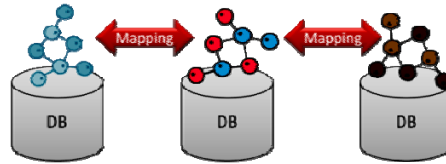


Figure 5. Source ontologies in multiple ontologies approaches [4]

Hybrid approaches- these approaches are the combination of previous approaches (single and multiple). The semantics of each information source is explicated by its own ontology independently. To ease the process of comparing these ontologies, a global ontology (like single approach) is constructed upon the source ontologies (see Figure 6).

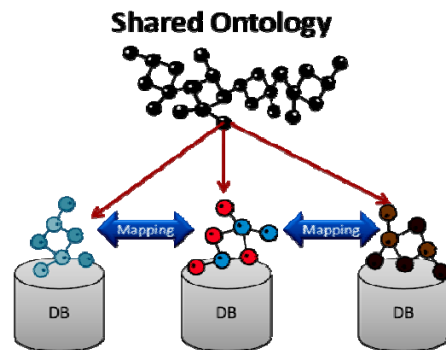


Figure 6. The shared ontology in hybrid approaches [4]

This ontology is called shared ontology or shared vocabulary. By doing these tasks, the obstacles of flexibility for change and aggregation/granularity in mapping approaches will be overcome. Before finishing this Section, two important remarks should be mentioned. Firstly, this approach has its own

drawback: the existing ontologies cannot be reused easily unless they are re-developed. Secondly, sometimes in literature the terms shared ontologies and shared vocabularies are misused unintentionally. Actually, the difference between these terms are explained concisely by Stuckenschmidt and Van Harmelen [4]:

“The shared vocabulary is a tuple $\langle W, l \rangle$, where W is a set of words and $l: W \times W \rightarrow [\text{syno}, \text{hyper}, \text{hypo}]$ is a partial function from the set of all pairs of terms into a set of identifiers specifying whether the first term is a synonym, a hypernym or a hyponym of the second. The axioms define the synonym, hypernym and hyponym relations between terms in terms of the subsumption relation.”

Where synonym means equivalent term, hypernym means a word with a broad meaning constituting a category under which more specific words fall and hyponym means a word of more specific meaning than a general or superordinate term applicable to it [10].

“And a shared ontology is a tuple $\langle ST, T, R, A \rangle$, where $ST = \langle W_L, l_L \rangle$ is a shared terminology, T is a basic set of terms, R is a set of relations $R \subseteq T \times T$ and A is a set of axioms of the form $T_i \sqsubseteq T_j$ if the following conditions hold:

- $T \subseteq W_L$
- For each pair of words (W_i, W_j) ,
 - If $l((W_i, W_j)) = \text{hyper}$ then $W_j \sqsubseteq W_i$ is in A ,
 - If $l((W_i, W_j)) = \text{hypo}$ then $W_j \sqsubseteq W_i$ is in A ,
 - $l((W_i, W_j)) = \text{syno}$ then $W_i \sqsubseteq W_j$ and $W_j \sqsubseteq W_i$ are in A .

Shared ontologies provide us with a vocabulary we can use in order to specify the semantics of information in different sources. The definition shows that the expressivity of shared ontologies can be considered as RDFS¹⁵ (see Appendix 3)”

The development of source and shared ontologies refers to the ontology engineering methodologies which are discussed in the forthcoming Section.

2.3. Ontology Engineering Process

The whole activities to develop ontologies such as ontology life cycle (some coherent and ordered blocks to create the ontology), the methodologies, and languages to build them are called ontology engineering/development [16]. Based on the IEEE standards for software engineering, an ontology development process has been extracted and proposed by Fernandez-Lopez et al [17]. This process

¹⁵ Resource Description Framework Schema (see Appendix 3)

includes three main activities such as: ontology management activities, ontology development-oriented activities, and ontology support activities. The sub activities of this process are illustrated in the Figure 7. In the following these activities will be described in detail [16].

Ontology management activities

Scheduling: identifying the essential tasks to be done, their arrangement, time, and the resources needed for the ontology engineering.

Control: it guarantees the completeness and performance of the tasks determined in the previous step.

Quality assurance: qualifying the ontology engineering outputs such as ontologies, documents and any prototype.

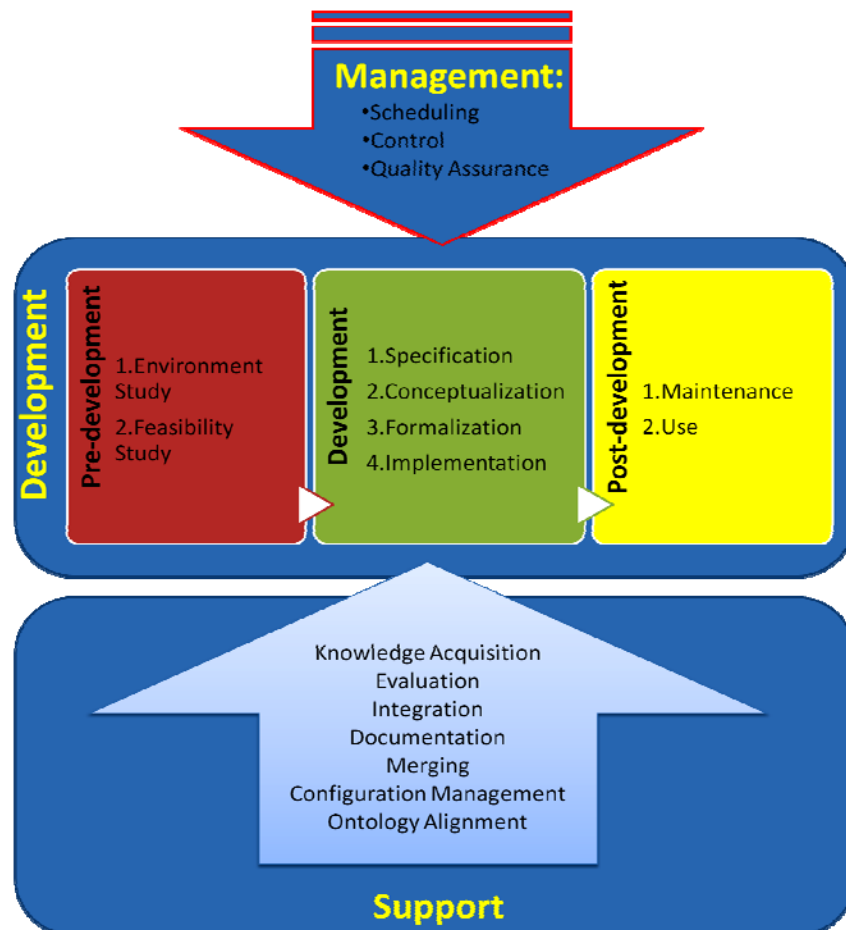


Figure 7. Ontology engineering process [16]; the support activities are parallel to development activities.

Development-oriented activities

These activities are mostly focusing on the construction of ontology components through three phases:

Pre-development

- Environment study: investigating the applications and platforms where ontology should be used
- Feasibility study: assessing the possibility and suitability of ontology building

Development

- Specification: determining the reasons for ontology building, its usage, and users in the future
- Conceptualization: as mentioned in Section 1.1.1, the prominent domain entities and relationships will be structured
- Formalization: transformation of the conceptual model into a formal or semi-computable model
- Implementation: using ontology languages to make the formalized models computable

Post-development

- Maintenance: refining and updating the current version (state) of ontology; this is a very important phase for the applications which are not static.
- Use: the exploitation of ontology for different applications or reusing for the other ontologies

Ontology support activities

The success of the development-oriented activities is strictly depended on these activities:

- Knowledge acquisition (KA): extracting knowledge from the domain experts (also the other knowledge resources) or through the ontology learning; the use of competency questions (CQ) is one of the useful techniques to extract knowledge from the domain experts. These are a list of questions which the ontology-based application should be able to answer to them [18]. Also the use of matrices, concept map tools, card sorting technique is useful in this phase.
- Evaluation: how much we can rely on the created ontology? How much the ontology represents the domain of discourse? These are some questions that the ontology evaluation should answer and validate the applicability of ontology for the domain. This evaluation can be done by the experts, developers, and the users of ontology based application. Five main parameters are defined to assess the quality of ontology [12]: technological (platforms and tools), structural, conceptual (taxonomy related), functional (intended use) like inconsistency checking and finally user-oriented (usage-related). Evaluation phase and maintenance phase are complementary of each other.
- Integration: reusing existed ontologies to develop new ontologies
- Documentation: after the completion of each activity and reaching to a result, every detail should be recorded.
- Merging: combining and merging different ontologies as the input to build an integrated ontology as the output which covers the components of input ontologies' components.
- Ontology Alignment: establishing the mappings between the common concepts in different ontologies which do not establish a new ontology
- Configuration management: an activity which records all the versions of documentation and ontology codes to control the changes

From the literature two inclinations in engineering of ontologies exist: centralized ontology engineering and collaborative ontology engineering. These inclinations are described in Sections 2.3.1 and 2.3.2 .

2.3.1. Centralized ontology engineering

As discussed in Section 1.1.2, the common methodology to develop ontologies is “engineer-oriented” or “centralized” which the role of ontology engineers to design and develop the ontologies is prominent.

And, it seems that the created ontology will be well-defined if more accomplished ontology engineers (OE) are engaged in the process of ontology development. In this methodology, there are two other roles: domain experts (DE) and end-users (see Section 1.1.2). Not being familiar and motivated about the other phases of methodology, the domain experts are mostly engaged in the knowledge acquisition phase via some techniques like competency questions and etc. The role of users in this methodology is the use of ontology-based application and providing feedbacks about the efficiency of application. But not more efficient feedbacks can be made by the users, due to the lack of proper infrastructure to communicate with the other roles. Totally the centralized ontology engineering has some disadvantages related to the quality of ontology:

1. The risk of ontology engineer inaptitude or the expert inability to describe the domain
2. Not taking the advantage of end-user's feedbacks
3. Static ontologies (not being up-to-date and adaptive to the current state of application) as the development of ontology is done once [19].
4. Difficult cooperation when the ontology engineers and domain experts are distributed geographically
5. No possibility to find some inconsistencies which are not recognizable by the reasoners, such as incompleteness of concept names
6. Not taking the advantage of more domain experts in some situations such as development of shared ontologies and vocabularies (e.g. taking the advantage of domain experts in the other organizations)

2.3.2. Collaborative ontology engineering

From the first days of semantic web emergence (2001), the significance of collaboration was considered by Tim Berners-Lee who confessed that the small group can innovate rapidly, but the others may not understand the created concepts and their relationships properly [9]. Similarly, three roles of centralized ontology engineering are the main roles of collaborative ontology engineering. A number of crucial characteristics of this kind of engineering which some of the researches and related projects have considered are:

The collaboration in ontology engineering intends to engage the mentioned roles while considering their "*skills and experience*" with respect to the ontology and domain. For instance, the end-user as an unfamiliar person with the principles of ontology is not forced to learn these basics. Instead, he can make comments about the applicability of ontology-based application through a proper way. Or the domain experts can be more engaged in this process if the ontological components can be represented for them in a more understandable way. Related to the ontology engineers, there is no concern about the representation of ontology for them. But, they can cooperate together in a decentralized infrastructure instead of working in specific groups and communicating via email or telephone calls. For instance if they are distributed geographically or they are from different organizations, a client-server architecture can link them together for the collaboration and communication. The collaborative ontology engineering infrastructure should ease the enrollment of these roles to participate in this process.

The other important point is the "*amount of collaboration*" among these roles. In some projects, the policies restrict the collaboration of other roles. For instance, the accessibility of a number of ontology engineers to the source ontology is forbidden while not being for the others. Or in one project, there may be only the collaboration of ontology engineers and domain experts, while being the collaboration of end-users and ontology engineers in another one.

Besides discussions about the skills and amount of collaboration, most of previous works have separated this methodology from the centralized one (no commonalities) while declaring that the collaboration should be started from an initial ontology. So there is a vague point here: are the centralized and collaborative methodologies two completely different methodologies? Or they have common phases. The answer of this question will be achieved if the answer of another question becomes clear: why collaborative ontology engineering? It means that there should be some logical reasons to apply collaborative methodology (instead the centralized). Actually these reasons are not the deficiencies which are explained in Section 2.3.1. Because if the ontology which is created by the centralized methodology has no any problem, there will not be any need for collaborative methodology (doing more efforts). From this statement, one can infer that before the use of collaborative ontology engineering, the results of centralized methodology have shown the need for ontology refinement. And in this sense, collaboration of different roles can be extended to the results of centralized methodology.

The definition of collaborative ontology engineering

Linking a collaboration extension to the centralized methodology is called collaborative ontology engineering. The reason for such a link is the need for the maintenance of ontology through its evaluation. So if the ontology is qualified enough, there is no need for the collaboration plug-in¹⁶.

Collaborative ontology engineering has two major phases: the development of an initial ontology through a centralized methodology, and collaboration of different roles to enhance the quality of initial ontology. The initial ontology is denominated as “Catalyst”¹⁷ ontology and the process of second phase as “COD” in this research (see Figure 8). COD is an acronym for Collaborative Ontology Development which is only phase two. So, *COD is not Collaborative Ontology Engineering*.

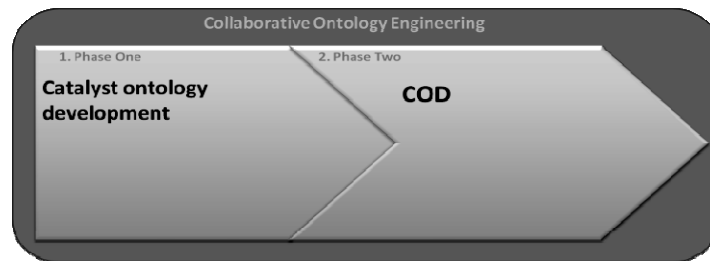


Figure 8. Collaborative ontology engineering and its major phases; the figure shows that COD is like a plug-in for centralized ontology development methodologies

In this architecture, COD can be considered as the extension of centralized ontology engineering methodologies to enhance their results. However, the challenges related to these phases are:

1. What is the proper methodology for developing catalyst ontology which is capable to link to COD (phase one)?
2. What are the requirements of COD (phase two)?
3. Which methodology can be considered for COD (phase two)?

¹⁶ Although plug-in term is used in software development, this is used here for the methodology to make it comprehensible for the reader.

¹⁷ Something that motivates other things or events to happen; this means that the immaturity of initial ontology motivates the collaborators to start discussion and cooperation on it.

4. Which platform can support the implementation of collaborative ontology engineering phases?

Therefore, answering to these questions will determine the collaborative ontology engineering. Also these questions are the main motivations for the further investigations in the forthcoming Chapter (Chapter 3).

2.4. Summary

For integrating information, the terms in one community should be transformed to the other community terminology. This transformation is only possible when the related context knowledge of each term becomes explicit. Information integration can be done through database and XML Schema mappings without any obstacle, if the used terminology among the communities is the same. As each community has its own terminology and policy in reality, the semantic heterogeneity hinders the information integration where the context knowledge cannot become explicit. Ontology as the formal and explicit specification of conceptualization copes with the mentioned problem.

Ontology-based information integration can be done through ontology mapping or the use of shared ontologies. Because of ontology mapping bottlenecks, the use of shared ontologies is adopted in this Chapter. To build ontologies, different methodologies are proposed such as centralized and collaborative. Collaborative ontology engineering is applied when the centralized developed ontology needs for refinement. This methodology has two phases: developing catalyst (initial) ontology and COD (collaboration plug-in for catalyst). At the end of this Chapter, four questions have been introduced which their answers will determine collaborative ontology engineering phases in detail.

Chapter Three

Collaborative Ontology Engineering Phases

To answer the questions related to the determination of collaborative ontology engineering phases (see Figure 9), this Chapter aims at: firstly, determination of proper methodology for developing catalyst shared and catalyst source ontologies in Section 3.1. Secondly, some requirements as the touchstones for the evaluation of COD methodologies and platforms will be determined in Section 3.2.1. Thirdly, upon these requirements the current methodologies which support COD are evaluated. And due to the deficiencies of them, a new methodology will be proposed in Section 3.2.2 which is called COD. Fourthly, upon the COD requirements a proper platform supporting COD will be adopted in Section 3.2.3.

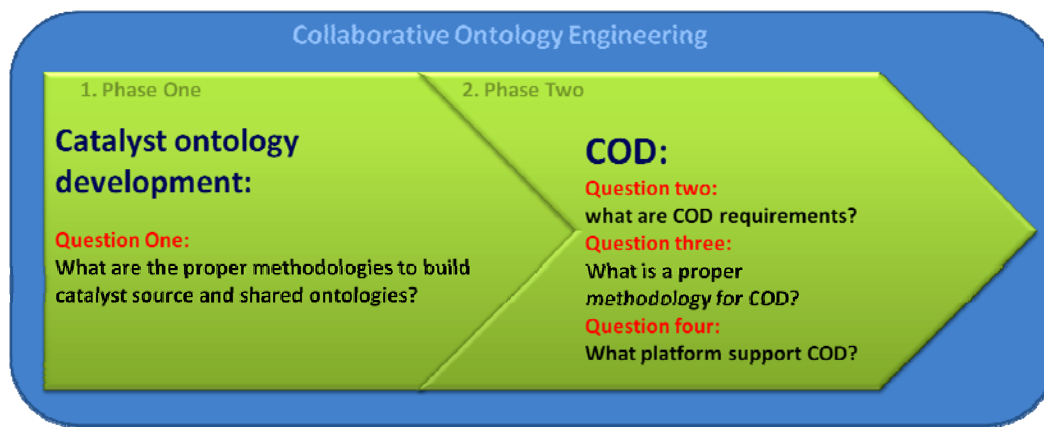


Figure 9. Four essential questions in collaborative ontology engineering; this Chapter answers to these question one by one through its Sections

3.1. Phase One: Catalyst ontologies

As introduced in Section 2.2.1, in this research shared and source ontology are two forms of ontology which are intended to be built for geo-information sharing.

3.1.1. Source ontology (Answer one)

Many centralized methodologies have been proposed until now which a kind of common idea among them is observable. Lemmens [5] has extracted these common perspectives in ontology engineering (see Figure 10):

1. Determine the scope of the ontology
2. Determine how it will be used
3. Making the sketch of ontology (in this phase the most general concepts and relationships are designed)
4. Create a list of all important concepts (at all detail levels) to be included
5. Create the ontology concepts and their relationships with axioms in OWL¹⁸ [20] with an ontology editor. Repeat this step for any newly added concepts, check every new axiom to see if the ontology remains consistent and is well-formed
6. Instantiate the concepts with individuals

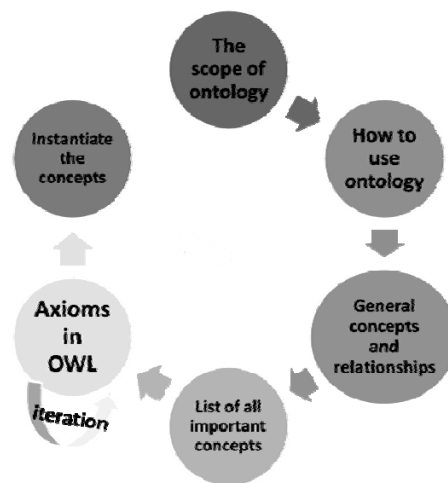


Figure 10. The extracted common ideas in ontology engineering Lemmens [5]

To illustrate that different methodologies have common phases, another perspective recommended by Van Harmelen and Antoniou is compared with the previous one [21]:

1. Determine scope (what is the domain that the ontology will cover? why are we going to use the ontology? for what types of questions should the ontology provide answers? who will use and maintain the ontology?)
2. Consider reuse (not to start from scratch if other people have been designed the initial phases of our development)
3. Enumerate terms (to write down in an unstructured list of all the relevant terms)
4. Define taxonomy (organizing the identified terms in a taxonomic hierarchy which subclass relationships among terms exist)

¹⁸ Web Ontology Language (see Appendix 3)

5. Define properties (the properties that link classes together)
6. Define facets; an important phase as five preceding phases only support expressiveness of RDFS, but in this phase the expressiveness of ontology will be increased by OWL primitives (enriching the previously defined simple properties with cardinality, required values (e.g. owl:hasValue), relational characteristics (symmetry, transitivity, inverse properties, and functional values). The advantage of this phase is inconsistency checking as RDFS (also OWL lite) expressiveness is unable for inconsistency checking.
7. Define instances (the ultimate desire of ontology engineering is to apply ontology for individuals in reality)
8. Check for anomalies (the populated individuals assist the ontology with inconsistency checking again)

These two examples illustrate the general agreement in the development of ontologies and both of them cover the same procedure in different statement.

The evaluation of centralized methodologies

The mentioned methodologies by Lemmens and Van Harmelen are indicating the general ideas and accepted steps among a lot of ontology engineering methodologies. From these methodologies, Gomez-Perez et al. [16] has investigated and evaluated the most applied ones such as Cyc, KACTUS, SENSUS, Uschold and King, Grüninger and Fox, METHONTOLOGY, On-To-Knowledge. To answer the question one (see Section 2.3.2), one of these methodologies has to be adopted. This methodology must support maintenance phase for ontology refinement (see Section 2.3). At the first step of evaluation some of these methodologies have been overlooked as they do not support maintenance and evaluation phases such as Cyc, KACTUS, and SENSUS. Through the evaluation of Uschold and King, Grüninger and Fox, METHONTOLOGY, On-To-Knowledge methodologies (see Appendix 4), the METHONTOLOGY is adopted in this research as the main methodology to build the catalyst source ontology. What come after are the reasons for this decision:

Common phases- the METHONTOLOGY has covered most of the phases of other methodologies.

Completeness of phases- Grüninger and Fox's methodology lacks maintenance phase. Also Uschold and king methodology has not mentioned anything about the conceptualization before the implementation [16]. The lack of conceptualization will make problem for the domain experts, and end-users to understand the ontologies in ontological languages [22].

Evaluation phase- it has addressed the evaluation phase in detail.

Accessible best practices- accessing to the experience and scientific outputs of ontology engineers engaged in SWING¹⁹ project. The main methodology to develop ontologies in this project was METHONTOLOGY. One of these scientific results is "Similarity as a Quality Indicator" [23] which is also related to the evaluation phase. Besides these, the use of collaborative ontology engineering was one of its recommendations:

¹⁹ <http://swing-project.org/index.html>;

*“... Semantic inconsistencies due to false assumptions, e.g. wrong classification, illogical conceptualizations, or contradicting axioms occur as well. Some of these errors might be discovered by automatic testing, others require **collaboration** with the domain experts or user feedback to be located...”*

After defining the METHONTOLOGY, it should be determined that which kind of ontologies can be built by this methodology in this research? As mentioned in Section 2.2.1, two kinds of ontologies will be developed for the sake of geo-information integration: some source ontologies (related to GI communities) and a shared ontology upon the source ontologies. For building the source ontologies, METHONTOLOGY can fulfill this research's needs. As through the development of shared ontologies the state of source ontologies should be considered, a different methodology has been proposed by Stuckenschmidt and Van Harmelen [4]. In below this methodology is described.

3.1.2. Shared ontology (Answer one)

Stuckenschmidt and Van Harmelen have proposed a centralized methodology adaptive to the building of shared ontologies [4]. The phases of this methodology are: first, defining the bridge concepts between the source ontologies concepts; bridge concepts subsume the pair of concepts that should be translated to each other. Second, the bridge concepts properties and their fillers to be used for the semantic definition of each concept should be defined. Third, the vocabulary should be adapted to the source ontologies (to invent some concepts, to specialize a concept more than its standard terminology). Fourth, evaluating and revising the ontology; if any concept has been overlooked (or redundant and inconsistency), it should be considered as the refinement.

Consequently, two centralized methodologies are determined to develop two forms of catalyst ontology in this research. The METHONTOLOGY for the development of source ontologies, and Stuckenschmidt and Van Harmelen for the development of shared ontology.

3.2. Phase Two: COD

As discussed in Section 2.3.2, the answers to questions two and three will determine the phase two. So in this Sections the COD requirements, COD methodology, and finally a platform supporting both COD and phase one will be determined.

3.2.1. COD requirements (Answer two)

Due to the investigation of implemented projects and studied researches, the specific requirements for supporting COD are [24-30]:

R.1) Integration, representation, and storing of discussions and annotations and changes in ontology development- annotation, discussion, and change are important actions in collaborative ontology engineering. W3C [31] has described annotation as: *“... Comments, notes, explanations, or other types of external remarks that can be attached to any Web document or a selected part of the document without actually needing to touch the document. When the user gets the document he or she can also load the annotations attached to it from a selected annotation server or several servers and see what his peer group thinks.”* There is no need to define discussion and change, but it should be mentioned that

they are about ontology components such as classes. The important task is to integrate them both in methodologies and platforms. The next important task is the representation of them for the collaborators. Collaborators should understand the meaning and the order of them. Use of user-friendly interface, forums and chats can enhance the representation and interface of discussions. The use of special signs, highlighting, and tagging by flags are good ways for the annotation representation. Changes can also be represented, but, the representation of them is a difficult task. Use of NL expressions, tables, and some interfaces such as PromptViz²⁰ can be useful to represent changes. Also all of these annotations, discussions, and changes should be stored for the future. For instance the thread of discussions should be kept for the other participants willing to enroll the discussions after some days or months. Or, other people can understand the rationale behind an ontology component by reviewing them.

R.2) Expressiveness (or expressivity) and ontology visualization- different roles should understand the ontology as it can have different levels of expressiveness. Specially, when the ontology is more expressive such as OWL DL²¹, there should be a way for the representation of ontology for the naive participants. Ontology visualization techniques (like tree view, component-based view, and cloud view) are useful in this case.

R.3) Continuous editing- not saving and archiving the ontology versions but keeping the track of changes in a continuous manner

R.4) Periodic archiving of different versions (versioning)- in this case, different versions of ontology are existed due to its changes during the time dynamically (usually one of periodic and continuous editing is adopted)

R.5) Determination of Jury- (as the panel of judges and administration) to manage the collaboration and control the decisions

R.6) Support for evaluation such as semantic/syntactic inconsistency checking- sometimes a user edits a class, which this change should be invoked for its subclasses inherently. So, there should be a facility to check the state of ontology after each change. Inconsistency checking as a machine-based evaluation indicator of functional parameters [12] is a useful way to evaluate ontology (see Section 2.3).

R.7) Provenance of information- it is critical for the participants to be capable to apprehend and search where information is coming from and the persons who make creation, change, vote and other activities. For instance, a comment from a person who is a professional expert in a domain can have good/bad influences on the other participants' idea.

R.8) Scalability (both in the size of ontology and the number of collaborators)- this is not only significant for the tool, but also the methodology should be flexible enough to support the scalability of both ontology components and number of ontology users.

R.9) Reliability- not to lose data

²⁰ visual representations of the differences between two versions of an ontology; <http://www.thechiselgroup.org/promptviz>

²¹ OWL Description Logic (see Appendix 3)

R.10) Robustness- ontology development tools should be robust enough as well as other tools applied by the participant.

R.11) The methods to reach to the consensus when different ideas exist

R.12) Access control- which is about user's restrictions and liberty to interact with the ontology components. A large number of policies can be considered for the accessibility of the collaborators. For instance, a user should only have permission to modify some parts of ontology where is her profession. So, the user can have the capabilities to edit/argument or only review the ontology.

R.13) Workflow support- workflow specification may include different tasks that editors are charged with, the process for proposing a change and reaching to the consensus, and the roles that different users play.

R.14) Supporting synchronous collaboration on ontologies (notifying other collaborators when a person annotates or changes an ontology component instantaneously)

R.15) Supporting asynchronous collaboration on ontologies (not synchronous; like periodic/continuous requirements, one of these requirements is considered for collaborative ontology engineering)

Some of these requirements such as determination of jury and the consensus methods are exclusively for the COD methodologies (not for platform), and the requirements such as reliability and robustness are only related to COD platforms. Now, after the determination of these requirements, the proper methodology and platform can be found with respect to these requirements.

3.2.2. The evaluation of COD methodologies (Answer three)

In contrast to centralized methodologies, only a few methodologies have been proposed related to COD. From them, three most referred methodologies are: Holsapple et al, Karapiperis et al, and DILIGENT. These methodologies are evaluated through COD requirements which are determined in Section 3.2.1 (see Table 1).

No	Parameter	Holsapple et al	Karapiperis et al	DILIGENT
1	Integration, representation, and storing of discussions and annotations and changes in ontology development	No	Yes (but no annotation)	Yes
2	Expressiveness and ontology visualization	Not discussed	Not discussed	Not discussed
3	Continuous editing	Yes	Yes	Yes
4	Periodic editing	No	No	Not discussed
5	Jury	Yes	Yes	Yes
6	Checking inconsistencies	No	Yes	Yes
7	Provenance of information	No	No	Yes
8	Scalability (size of users)	No	No	Yes
9	Consensus achievement	Yes	Yes	Yes
10	Access control	Not discussed	Not discussed	Yes
11	Workflow support	Yes	Yes	Yes (but not sufficient discussion about the roles and activities)
12	Synchronous/Asynchronous	Not discussed	Not discussed	Not discussed

Table 1. The evaluation of current COD methodologies with respect to the COD requirements

From the mentioned requirements, the workflows of these methodologies are evaluated in detail. Because, this is the most important factor in collaboration as it organizes collaboration. All in all, the following pros and cons in these methodologies are considerable (the similarity between Holsapple and Karapiperis caused to integrate them as one methodology):

Holsapple and Karapiperis advantages:

Firstly, the role of moderator helps for the control of collaborative ontology engineering process and not becoming divergence from the targets of ontology engineering. Secondly, supporting workflow for reaching to consensus and making decision is an advantage in these methodologies.

Holsapple and Karapiperis disadvantages:

Firstly, they are only applicable for the collaboration of ontology engineers while overlooking the other roles (domain experts and end-users). Secondly, in these methodologies the reason for collaboration has not been determined. They have mentioned that creating an initial ontology is sufficient to start the collaboration. Here the critical point is that what is the need for collaboration as one can develop an ontology without any need for maintenance (in centralized methodologies)? So, they have missed a very important point: “The collaboration should start when a lot of efforts have been put into the development of ontology, but the evaluation of ontology shows the need for maintenance. A wrong structured ontology will mislead its maintenance and there is no guarantee to reach to a right ontology”. Third, the required tools for collaboration in these methodologies (Questionnaire and Evaluation Sheet; see Appendix 5) are not integrated into the tools of ontology building. This makes problem when the ontology has many concepts and relationships where finding the corresponding concepts becomes difficult in discussions. Fourth, they have not discussed sufficiently about the development of catalyst ontology (only Karapiperis

has enumerated the general ideas of ontology engineering). Fifth, some of important requirements of COD are not supported by these methodologies regarding to Table 1.

DILIGENT advantages:

Firstly, the role of moderator helps for the control of collaborative ontology engineering and not becoming divergence from the targets of ontology engineering. Secondly, supporting workflow for reaching to consensus and making decision is an advantage of this methodology.

DILIGENT disadvantages:

First, the situations for the collaboration of participants are the same. This methodology does not mention anything about the different scenarios which may occur for different roles such as end-user, ontology engineer, and domain expert. Second, although the formalization of arguments helps for finding inconsistent discussions among the collaborators, this task makes workflow inflexible in situations that the collaborators want to change or modify their idea. In these situations, other collaborators can find inconsistent discussions themselves and there is no need for formalizing them. This research believes that if a person changes his idea, the collaboration is in a good state as he could enhance and revise his idea through the collaboration. It is predictable that ontology engineer has not modeled the domain carefully. Or the domain expert cannot convey his knowledge to the ontology engineer completely. Third, this methodology has not discussed anything about the expressiveness and ontology visualization, periodic editing and the importance of versioning (see Table 1).

Proposing a methodology for COD

Due to the mentioned deficiencies in the current methodologies (in previous paragraphs), a methodology for COD, based on “Consensus Decision Making”²² [32, 33], is proposed in this Section which addresses the COD requirements while not having the mentioned disadvantages. In below, COD methodology is described through its main components:

Roles

Administrator: the person who controls the process of collaboration. He registers some participants to join to the collaboration (each person has a profile). He considers the project’s policies, discussions, annotations, changes, emotions, and all activities related to the participation of collaborators. He cannot make decision by himself and must let the group to reach to the consensus. If the group does not reach to the consensus or any decision, he will stop the discussions and announce that no decision has been made due to this discussion. In contrast if the group have consensus on a decision, the administrator should announce it. If reaching to the consensus needs to start a voting process, he should ask the collaborators to vote due to the existence of different ideas. The announcement of decision by the administrator has some advantages: every participant will be informed about the made decisions (that is a valid decision), the new participants can review the decisions which have been made legally. He can ignore other roles activities if they break the rules and policies of project. Another responsibility of administrator is to transmit the results of evaluation phase by himself or the group of evaluators to the other participants. As the minimal

²² “Instead of simply voting for an item, and having the majority of the group getting their way, the group is committed to finding solutions that everyone can live with. This ensures that everyone’s opinions, ideas and reservations are taken into account.”; <http://seedsforchange.org.uk/free/consens#guide>; accessed at 20 Oct 2009

need, they must do semantic inconsistency checking as a machine-based functional evaluation of ontology.

He is also responsible for configuring server and related policies. The ontology should be uploaded on the server and visualized for the collaborators by: the translation of ontology concepts and their definitions in NL expressions, tree view of ontology hierarchy, and OWL file (or other ontology languages).

Ontology engineer: this role is familiar with ontological studies and languages. A very important help of ontology engineer is to check the ontology components and specially the correctness of axioms (as the most difficult and error-prone part of ontology engineering). Also the representation of ontology for the other roles may have some deficiencies which only an ontology engineer can recognize them. For example if the axioms (in logical representation) are translated into the Natural Language (NL) sentences, they can evaluate the correctness of this translation. In COD, they are capable to annotate, but not to edit the ontology.

Domain expert: They should not only be engaged in knowledge acquisition phase, but also in maintenance and evaluation. They can discuss with the other collaborators about the domain and how the domain knowledge is structured by the ontology. They cannot discuss about the ontology languages but COD translates the ontology into NL expressions to be understandable for the domain experts. In COD, they are capable only to annotate, but not to edit the ontology (like ontology engineer).

User (end-user): in COD methodology they cannot be registered directly to interact with the other roles. This is because of their familiarity with ontology and domain. Instead, they can give feedbacks about the applicability of application (which is based on ontology). Although the end-users like the domain experts do not have the knowledge of ontology engineering, the domain experts have been taken part in KA (conceptualization) phase and are familiar with the rationale behind the ontology. In this methodology they should send feedbacks to the administrator if any bug or problem is recognized by them.

The architecture

The architecture of COD is client-server for distributed collaboration. The ontology is stored in server and the collaborators can login to it via username, password. Also every event by each collaborator must be seen by the other collaborators (clients).

The scheduling

For the sake of discussions, it is better to define an exact time for logging in and collaborate simultaneously. This can make an organized schedule for the collaboration and project. But this may not be possible if the number of collaborators is numerous as they are busy sometimes. However, the best case is that after taking place an event (a new annotation, discussion thread, or change) the other collaborators should be notified via email or other communicating services. COD suggests periodic editing to have versioning.

Workflow for collaboration

The proposed workflow is illustrated in Figure 11.

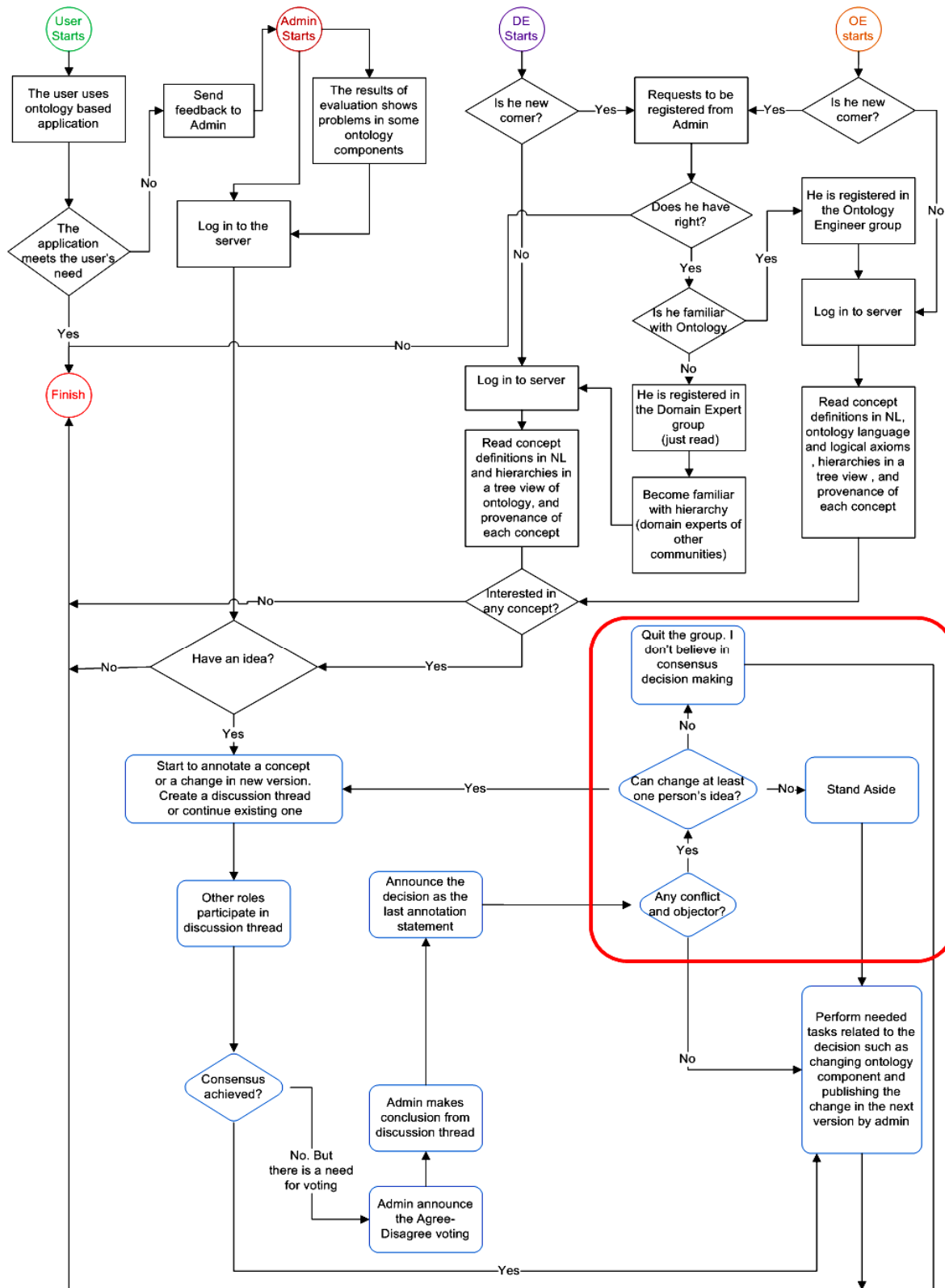


Figure 11. COD workflow in a client-server architecture; the sessions related to discussions threads (in blue color) and decision making (in red color) are illustrated in the lower part.

The collaboration starts in four situations: first, the administrator gets feedbacks from the end-user and if it is related to the ontology, he reflects this issue to the other collaborators by annotating an ontology component. Second, the administrator starts a discussion thread with respect to the results of evaluation phase. The problematic issues should be annotated for starting the collaboration by other collaborators. Third, when the domain experts login to the server and after reviewing the definition of ontology components in NL, hierarchies in a tree view of ontology, provenance of each concept, annotates a concept due to his interest and idea. Or he continues a discussion thread which has been started before (annotated before). In the latter case, he should be notified when the discussion thread had been started before. Fourth, the ontology engineer logs in to the server and acts the same as described for domain expert. But he can read also the ontology language or logical axioms and compares them with their NL definitions.

Then the discussion (between ontology engineers and domain experts) will result in two situations (see Figure 11; blue sessions): a consensual decision is achieved, no consensus is achieved and there is a need for voting. In the first case, the collaboration will be finished after administrator's announcement of result. In the second case, a voting process (agree or disagree votes) will be performed and the result will be announced by the administrator as the final decision. But here, an exception exists (as an advantage in consensus decision making; see Figure 11, red box). If one or more persons object and do not accept the results of voting, they can do one of the following tasks: they can change at least one of the other participant's idea; in this case, the discussion should be started again and the mentioned steps. Or they cannot change at least one person's idea; in this case, they are free to set aside from the discussion (become as a neutral person) or quit from the group collaboration.

Annotation types and GoogleMap annotation type

Some annotation types are adopted from [30] (based on Annotea [34]) which the collaborators can use them in COD. These types are the core annotation types for steering the discussions and not to converge. Also, other annotation types can be added due to the application like GoogleMap type in this research. The use of GoogleMap type is to represent geospatial concepts like a forest in specific location (geospatial object) and representing the geotagged images (linked to the Google maps) of some vague concepts. These techniques are used to improve the expressivity of a concept. For instance, if a collaborator cannot understand the semantics of a concept, other collaborators can represent the map or the geotagged images of this concept.

In contrast to DILIGENT, COD does not restrict the use of annotation types for the collaborators as it ignores the automatic finding of inconsistencies in the annotations (with respect to be flexible and not pushing the collaborators). These inconsistencies can be easily found by the collaborators and administrator themselves. In the following the annotation types in COD are explained (see Table 2):

No	Annotation type	Description
1	ArrivedVerdict	For the announcement of reaching to a decision; it is used only by administrator
2	Comment	All of the roles can use it to make comment about anything related to the ontology. This is a very common annotation type to start the discussion threads. This can be inferred that the feedbacks from the users are in this type; which are made by administrators.
3	Example	If any role wants to give an example in his discussion.
4	Explanation	Any response and reaction to a Question or Comment
5	GoogleMap	When a user can enhance the expressivity of a concept meaning by representing its map (if this is a geographical concept) or a geotagged image on the Google map.
6	AgreeDisagreeVoteProposal	When a person wants to propose something which needs the other persons' agreement or disagreement in a voting process. The response can be agreement or disagreement of the others. It is also possible for the others to use other annotation types to make the discussion flexible for collaboration. E.g. a person proposes to add a concept, but the other one responds him by Question. It is may be because of vague point in this proposal for him.
7	Question	If a participant has a question.
8	SeeAlso	To refer to the other ontology components

Table 2. Annotation types in COD

As claimed at the beginning of this Section, COD fulfills the requirements discussed in Section 3.2.1. To prove this claim, COD is evaluated in Table 3:

No	Parameter	COD
1	Integration, representation, and storing of discussions and annotations and changes in ontology development	Yes, each discussion thread is about an annotation in current version or a change in new version.
2	Expressiveness and ontology visualization	Yes, the translation of ontology concepts and their definitions into NL expressions, tree view of ontology hierarchy, and the use geotagged images and Google maps help the collaborators (especially domain experts) to understand ontology in different level of expressiveness.
3	Continuous editing	No, this is not proposed by COD. Because of versioning advantages for evaluating the changes, discussions, and annotations in different versions.
4	Periodic editing	Yes, through the versioning of ontology
5	Jury	Yes, the administrator and its control
6	Evaluation (Checking inconsistencies)	Yes, by evaluation methodologies in METHONTOLOGY (reasoning services)
7	Provenance of information	Yes, each collaborator should login with his username and password related to his profile. So to each annotation or discussion, the name of the collaborator should be attached.
8	Scalable (size of users)	Yes, due to the project's policies, the administrator can register many collaborators. It is proved that projects based on consensus decision making are scalable for many users (e.g.

		Wikipedia)
9	Consensus achievement	Yes, Consensus Decision Making is the basis of COD. Consensus Decision Making has been successful in many projects such as Wikipedia (the biggest knowledge base in the world).
10	Access control	Yes, registration by the administrator through the username and password. Domain experts and ontology engineers are not permitted to edit the ontology. Only administrator can change and edit the ontology. In this way, there is no concern about the security and personal change of ontology components.
11	Workflow support	Yes, in Figure 11 different roles and tasks are determined for the sake of collaboration.
12	Synchronous	Yes; any event by any participant should be visible for the other participants
13	Asynchronous	No, it is not recommended in COD.

Table 3. Evaluation of COD with respect to its requirements; some of the requirements related to tools have not been discussed here

Also, COD does not have some deficiencies discussed about the previous methodologies such: not discussing enough about the development of catalyst ontology (by proposing METHONTOLOGY), restricting the use of annotation types in DILIGENT (although in Table 2 the use of these types are explained for the collaborators, they are free to use them for their discussion threads), not considering different scenarios for different roles in DILIGENT (in Figure 11 four different scenarios for four different roles are illustrated in Figure 11).

3.2.3. The platform supporting both phases (COD and Catalyst development)

Doing collaborative ontology engineering needs a proper platform (tool) supporting two phases of this methodology. Related to the METHOTOLGY and Stuckenschmidt (catalyst ontology), there is no need to evaluate the proper platforms as most of the ontology editors support centralized ontology engineering. But related to the second phase, some of the platforms supporting COD will be evaluated to adopt a proper one for this research. From the literature, two main categories of platforms support COD such as wiki-based and non-wiki-based.

Wiki-Based platforms

A Wiki Web (repository of ideas and a tool for collaboration) is a web site where users can contribute to modify data and share ideas by adding contents on any page [35]. The most famous wiki is Wikipedia²³ in which each concept has a unique identifier (its URL²⁴). Wikis can be contemplated as a kind of evolving graph which its pages are the nodes and the links among them are the edges [35].

The stimulating characteristic of wikis is their simplicity to be learnt and used which has motivated many people to apply wikis in the process of ontology engineering. Recently the wiki technologies and semantic web technologies have been merged to enhance the capabilities of each other. The first scenario is to apply semantic web technologies (like ontologies or RDFS) for the wikis to enhance wiki-based services e.g. enhanced information retrieval or making semantic queries (structured queries) on wikis. As

²³ www.wikipedia.org

²⁴ Uniform Resource Locator

the amount of stored data in wikis are becoming huge (e.g. nearly 3 million articles in Wikipedia), the search results are not so related to the intention of queries, while the help of semantic web technologies can lead to more related results [36]. Another scenario is to take the advantage of wikis to build and edit ontologies collaboratively as the wikis provide user-friendly interface, extensive versioning support and discussion pages to support collaboration [37]. The “semantic web for wikis” (aka ontologies for wikis) is not related to the objectives of this research. But for the second category, “wikis for ontologies”, some of the pertinent systems and their specifications will be evaluated. They are proper for the development of lightweight ontologies as they do not support complex semantic knowledge and inferencing [37]. The problem of wiki-based platforms is that they have missed the simplicity of wikis by using object properties or datatype properties among the pages of wiki. Some of the most discussed platforms of this category are: MyOntology [29], OntoWiki [38, 39], IkeWiki [28, 37], and AceWiki [40].

Non-Wiki-Based platforms

The non-wiki-based platforms mostly rely on ontology editors which have the capabilities for collaboration. They enable users to model more complicated ontologies (heavyweight) compared with models from the wiki-based platforms. However these platforms are not easy to be learnt and used by naive participants. Some of the most discussed platforms of this category are: Tadzebao [41], WebOnto [41], OntoEdit [42], COE [43], Ontolingua Server [44], Ontoverse [45], WebODE [46], and Protégé [30, 47].

Evaluation of platforms

Before evaluating them, it has to be defined which of these platforms are stable, active, and accessible now²⁵. Among these platforms only OntoWiki, COE, Ontoverse, and Protégé have the acceptable state²⁶. Also OntoWiki, COE do not support inconsistency checking (see Section 3.2.1) which is one of the most important requirements of COD (discussed in Administrator’s duties). However, only Ontoverse and Protégé satisfy the COD requirements better than the others. Each of these platforms can be used for this research, but Protégé is adopted because of the following reasons: firstly it is supported by the many developers, manuals, numerous plug-ins such as Jambalaya and OWLViz (powerful plug-ins for ontology visualization), collaborative Protégé, Client-Server Protégé (for distributed engineering). Secondly it can be called as the most live and up-to-date platforms among the others²⁷. It is because of its big community, clients, and interest of other developers to develop their own plug-ins in this platform. And the last one but not the least one, it is an open source software. More information about Collaborative Protégé is explained in Appendix 6.

3.3. Users of COD

With respect to the outputs of Sections 3.1 and 3.2, the target communities of COD are geospatial communities. Although the use of Google maps and geotagged images supports the expressivity of geospatial concepts, the main idea behind this technique can be extended to the other communities such as biomedical engineering and the studies related to social networks.

²⁵ November 2009

²⁶ Ontolingua server also fulfills these criteria, but its version is somehow outdated (<http://ksl.stanford.edu/software/ontolingua/>)

²⁷ <http://semanticweb.org/wiki/Tools>

3.4. Summary

Regarding to four answers for the defined questions in Section 2.3.2, collaborative ontology engineering phases are determined in this Chapter. First, METHONTOLOGY and On-To-Knowledge are suggested as the capable methodologies for creating catalyst source ontology. Nevertheless the METHONTOLOGY is selected in this research (see Section 3.1.1). Also Stuckenschmidt and Van Harmelen methodology is adopted to develop the catalyst shared ontology (see Section 3.1.2). Second, in Section 3.2.1, the COD requirements are defined to have touchstones to answers question three and question four (phase two). Third the current COD methodologies are evaluated in Section 3.2.2. Due to the deficiencies in these methodologies and not supporting the COD requirement sufficiently, a new methodology for COD is proposed in this Section. The evaluation of this methodology by COD requirements (see Table 3) proves its qualification for the usage in the second phase. Fourth, to implement COD for the sake of geo-information sharing, Protégé and some of plugins (OWLViz, Jambalaya, and Collaborative Protégé) have been adopted in Section 3.2.3.

Chapter Four

Case study: Geo-Information sharing among ForeStClim communities

As discussed in Section 2.2, the ontology is a solution to explicate the implicit knowledge while sharing and integrating multiple datasets of a domain. In this Chapter, the semantic problem in geo-information integration between two geospatial organizations (ONF, SERTIT) will be resolved through the development of a shared ontology upon ONF and SERTIT source ontologies (see 2.2.1). The shared ontology will be developed in two different methodologies: collaborative and centralized. At the end, the results of these methodologies will be compared to evaluate collaborative ontology engineering. The whole process is illustrated in Figure 12:

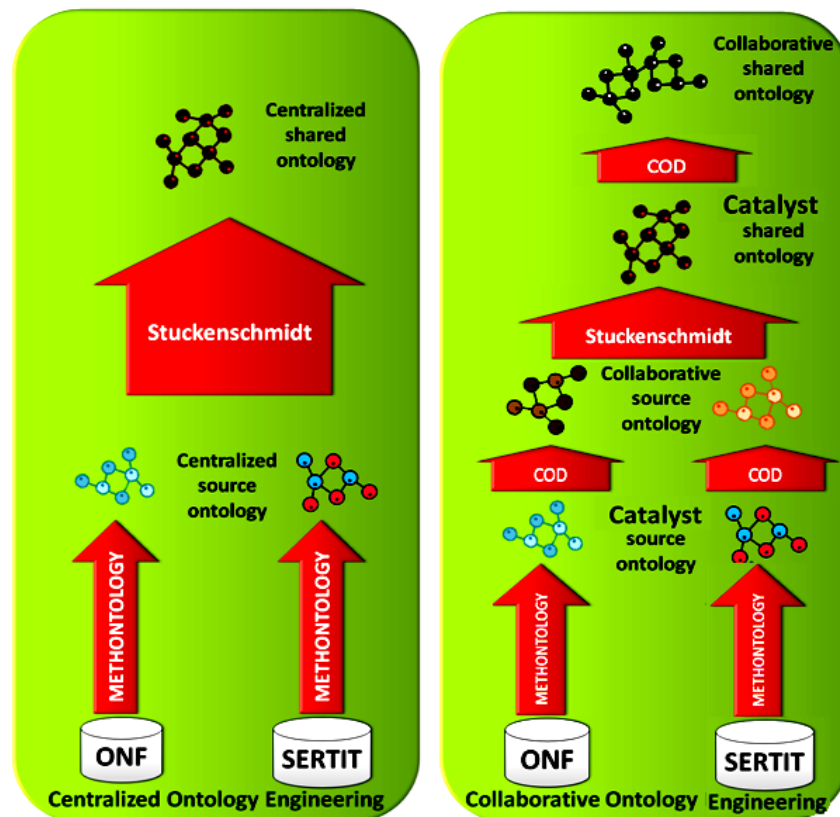


Figure 12. The whole process of ontology-based GI integration through the centralized and collaborative methodologies; these two methodologies are illustrated in two packages (left, right) to be comparable with each other.

4.1. ForeStClim: an EU project

Regards to the aim of this project to have cooperation among 21 communities in different European countries, the cooperation of two communities is considered for this research. ONF²⁸ as the data provider and SERTIT²⁹ as the user of ONF datasets have this cooperation for the sake of geo-information sharing. Applying Knowledge-Based classification techniques (besides the analysis of spectral characteristics), SERTIT is classifying the satellite images of Alsace-France. To improve the classification results, they use the GIS layers of ONF as one of knowledge resources. The domain of this cooperation is forestry and each of these organizations has a special classification system for itself. The ONF classification covers the tree species such as larch, oak, and etc, but, SERTIT classification focuses more on general types of trees and forests (coniferous and deciduous). The ordinary task to improve the classification results is to overlay the ONF and SERTIT GIS layers. Corresponding to the Figure 13, one may quickly claim that ONF layers (tree species) can be easily overlaid on SERTIT layers if the attributes of ONF features are changed to coniferous or deciduous. This solution for data integration among ONF and SERTIT is illustrated in Figure 13.

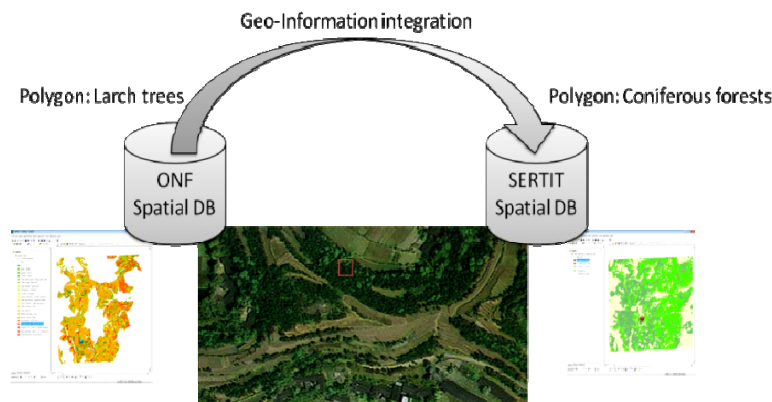


Figure 13. Integrating two different datasets stored in ONF and SERTIT spatial databases based on two different classification systems

Besides defining the coniferous or deciduous characteristic of species, some other important points should be taken into account by SERTIT such as: considering the spectral characteristics of each species. Therefore the complete and correct name of each species should be existed in ONF layers (different tree species have different spectral characteristics).

This translation from the ONF classification to SERTIT classification will not be feasible if the semantic heterogeneity isn't resolved. In the remained parts of this Chapter, the following steps will be done to resolve this problem (see Figure 12):

1. Developing ONF and SERTIT ontologies (related to their classification) as the source ontologies in a centralized methodology (METHONTOLOGY; see Section 3.1.1); they are called ONF and SERTIT catalyst ontologies in this research.

²⁸ <http://www.onf.fr/>; *Office National des Forêts*

²⁹ <http://sertit.u-strasbg.fr/>; a remote sensing and image processing service which is related to the University of Strasbourg

2. Developing a shared ontology upon the ONF and SERTIT catalyst ontologies through the Stuckenschmidt and Van Harmelen methodology (see Section 3.1.2); this is called Catalyst shared ontology.
3. Applying COD for the ONF and SERTIT catalyst ontologies to reach to ONF and SERTIT collaborative ontologies
4. Developing a collaborative shared ontology upon the ONF and SERTIT collaborative ontologies

Eventually, these shared ontologies (collaborative and centralized) will be evaluated to appraise the capabilities of collaborative ontology engineering.

4.2. Centralized ontologies

In this Section, the left package illustrated in Figure 12 will be developed. In this package, two source ontologies (ONF and SERTIT catalyst ontologies) and a shared ontology upon them (catalyst shared ontology) will be developed (hybrid approach; see Figure 6).

4.2.1. Step one: Developing the ONF and SERTIT catalyst ontologies

The METHONTOLOGY phases for developing ONF and SERTIT catalyst ontologies are (see Appendix 4):

Pre-development

Environment and Feasibility Study: These activities were performed in the first meeting with the domain experts. In this meeting, the principles of semantic conflicts were described for them whilst citing the examples of forest definitions in French organizations. Also the selected tools such as Protégé and CMapTools COE (a tool for the knowledge acquisition phase) were introduced in a nutshell. At last, the existed knowledge resources in these organizations such as domain experts, databases, GIS layers were determined.

Development

Specification: as discussed in Section 2.2, the reason for the development of ontologies in these organizations is geo-information integration (aims at semantic GI integration among ONF and SERTIT communities). Also these communities are engaged in ForeStClim project in which other domains like Climate Change are discussed. So these ontologies can be integrated to the Climate Change ontologies for the similar scenarios in future. Regarding to the specification of expressiveness, the development of expressive ontologies (heavyweight) is determined for the sake of semantic inconsistency checking.

Conceptualization to structure the domain: Before starting the second round of meetings, the GI layers and databases of these organizations were investigated. The main structures of ONF and SERTIT catalyst ontologies were extracted by surveying their database schemas, GI layers, and Powerset³⁰ (enhanced Wikipedia). Then the Knowledge Acquisition (KA) phase was started through the face to face meetings

³⁰ <http://www.powerset.com/> ; Powerset is first applying its natural language processing to search, aiming to improve the way we find information by unlocking the meaning encoded in ordinary human language. Being acquired by Microsoft, it is now focusing on Wikipedia and more facilities to search on it.

with the domain experts. In this step, the use of an Intermediate Representation tool³¹ (COE) caused to prepare the domain experts for interacting with Protégé environment in the next phases. As both of SERTIT and ONF classifications are in hierarchical structure, the domain experts could easily build the hierarchies in COE. As an innovation, the ontological properties of each concept were imported in COE through the NL expressions. At the end, other knowledge resources related to each concept were also imported in COE such as GIS layers, Wikipedia pages and etc. Corresponding to Figure 14, the GIS layers of SERTIT classification hierarchy are attached to related concept as an acquired knowledge. The detailed steps of conceptualization phase are explained in Appendix 1.

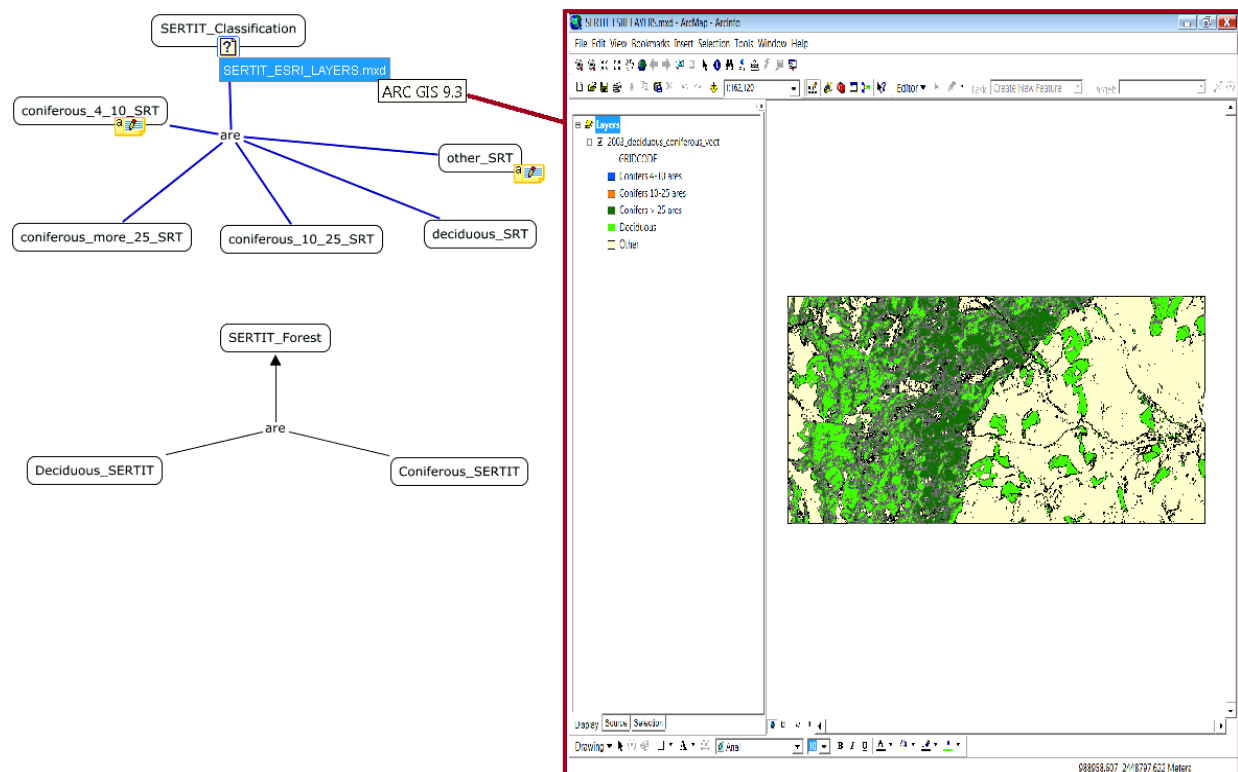


Figure 14. Linking the GI layers of ESRI ArcGIS to SERTIT hierarchy; “are” label represents the subclass/superclass relationship and was understandable for the domain experts as a kind of sub-category relationship.

Formalization and Implementation: These phases were performed in Protégé together. The following tasks were done in this phase: importing the OWL file of class hierarchy that was exported from COE, defining disjoint classes, importing properties from COE (in NL), considering inverse properties, functional properties (single valued property), inverse functional properties, transitive properties, symmetric properties, and defining the classes by using axioms and property restrictions.

Corresponding to the collaborative intention of this research, the translation of ontology components (like axioms) into their corresponding NL expressions helped the domain experts to understand the ontology (see Section 3.2.2). In this research there was no need to do additional task for translating them

³¹ In this research CMapTools COE was used as the intermediate representation tool.

into NL. Due to KA phase, all information about each concept was imported in NL expressions (in COE). For instance, the following conditions in the definition of `Deciduous_SERTIT` class are asserted:

\exists `loose_leaves_in` seasons, \exists `loose_leaves_in` winter, \neg `non_tree_land` as the sufficient and necessary conditions and `Deciduous_SERTIT` \sqsubseteq `SERTIT_classification` as the necessary condition. These conditions are not understandable for a person who is not ontologist. So, they can be represented by their corresponding meaningful NL expressions such as: It is a kind of `SERTIT_classification`, it is NOT `non_tree_land`, and it loses its leaves just in winter³².

Finally, the OWA³³ was checked before importing the individuals to the ontology.

Post-development

Evaluation and Maintenance: By using the Pellet reasoner [48], a semantic inconsistency related to `Other` class in ONF ontology was found. The reason for this inconsistency was that a superclass and its subclass were disjoint. This was not because of ontology engineer's mistake but the acquired knowledge was inconsistent inherently. After consultation with the domain expert, this inconsistency was resolved by removing the disjoint constraint. Also inconsistency checking was performed for SERTIT ontology and no inconsistency could be found.

4.2.2. Step two: Developing the catalyst shared ontology

As discussed in Section 3.1.2 and 3.1.1, the METHONTOLOGY cannot support the development of shared ontologies. So the explained methodology by Stuckenschmidt and Van Harmelen is applied to build the shared ontology. Also, they have proposed some resources to develop the shared ontology (like Upper Cyc ontology). In this research Upper Cyc ontology³⁴ was adopted to select the bridge concepts and properties. Not having sufficient fillers and bridge concepts in Cyc ontology, there was a need to use some scientific taxonomy to complete the shared ontology. So in the process of adapting vocabulary (phase three in Stuckenschmidt), three classes of tree species were added to the shared ontology. Also the subclasses of one of the classes (`Magnoliophyta_cyc`) were transferred to the set of `deciduous_plant_cyc` subclasses. This class was deleted after transferring its subclasses. In Figure 15, the developed Catalyst shared ontology is demonstrated as the output of centralized ontology engineering:

³² The reason for using \exists `loose_leaves_in` seasons condition is to guarantee the existence of

`loose_leaves_in` relationship. This is something that the current translators translate them to NL as they are not intelligent.

³³ Open World Assumption

³⁴ http://www.cyc.com/cyc/technology/technology/whatis_cyc_dir/whatsincyc



Figure 15. The Catalyst shared ontology upon the ONF and SERTIT catalyst ontologies; OWLViz plug-in of Protégé is used for this tree representation of shared ontology

In the last phase (refinement) an inconsistency was found through the use of reasoner. It was because of Larch class as it was the subclass of both `deciduous_plant_cyc` and `coniferous_tree_cyc` classes (two disjoint classes). It was supposed that each ONF class should be translated into only one of the SERTIT classes (in NL, it means that each tree species is coniferous or deciduous). And also for overlaying ONF and SERTIT layers, each feature of ONF cannot have two attributes (coniferous and deciduous)! By reviewing the other knowledge resources like Wikipedia (by using CMapTools COE as the knowledge repository and the attached URL of Larch tree), Larch tree has been considered as both of coniferous and deciduous³⁵. However, after referring to the ONF metadata (stored in COE), it was only considered as coniferous tree. After considering Larch as the subclass of `coniferous_tree_cyc`, each ONF class (refinement), firstly each ONF concept was translated to its equivalent concept in the shared ontology. For instance, `Mélèze_European_Larch_ONF` concept was mapped to `LarchTree_cyc` as the equivalent classes. Secondly, SERTIT classes (only coniferous and deciduous) are mapped to the shared ontology concepts. For instance, `Coniferous_SERTIT` was mapped to `coniferous_tree_cyc`. As `LarchTree_cyc` is the subclass of `coniferous_tree_cyc`,

³⁵ Larches are conifers in the genus *Larix*, in the family Pinaceae. They are native to much of the cooler temperate northern hemisphere, on lowlands in the far north, and high on mountains further south. Larches are among the dominant plants in the immense boreal forests of Russia and Canada. They are deciduous trees; <http://en.wikipedia.org/wiki/Larch>

Mélèze_European_Larch_ONF must be the subclass of Coniferous_SERTIT. The mentioned mapping was done by prompt ontology mapping plug-in.

4.3. Collaborative ontologies

The created ontology through the whole phases of centralized process until post-development can be considered as the Catalyst ontology in phase one. Before starting phase two, the catalyst ontologies were placed on the server. After the configuration of server and project policies, see Appendix 2, clients connected to the server for collaboration.

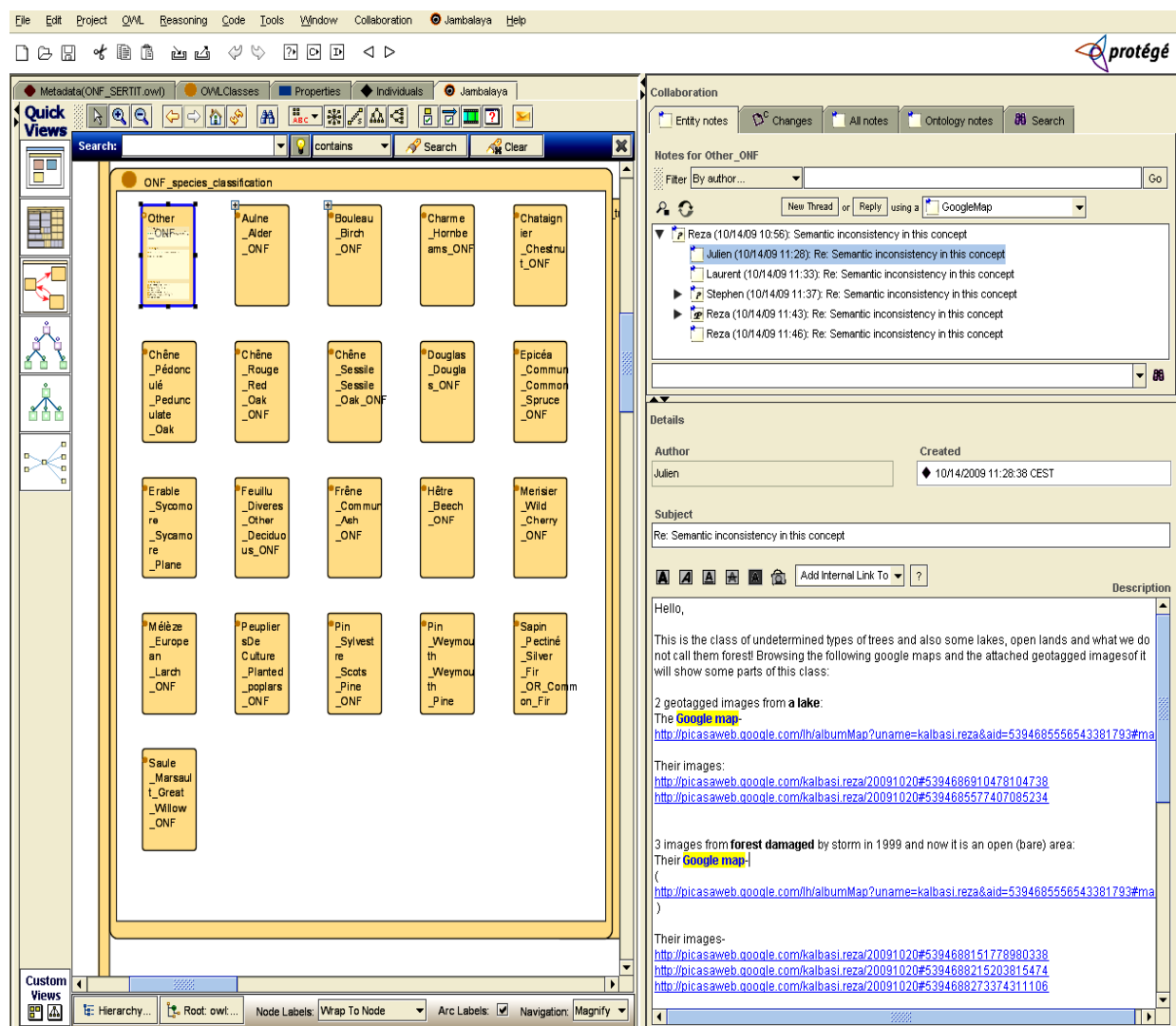
4.3.1. Step three: Developing the ONF and SERTIT collaborative ontologies

After logging in to the server, the collaborators started to discuss about the ontology components. Due to the proposed workflow for COD (Section 3.2.2), some discussion threads occurred. One discussion thread about the SERTIT ontology and seven discussions threads related to ONF ontology were recorded. The results of these collaborations are illustrated in Table 4.

No	Concept	Topic	Decisions	Frequency of conversations	Rejecting any idea	Consensus
1	Aulne_Alder_ONF	Wrong name & Reduntant subclasses	Modify name & Delete subclasses	6	No	Achieved
2	Bouleau_Birch_ONF	Wrong name & Reduntant subclasses	Modify name & Delete subclasses	5	No	Achieved
3	Chêne_Rouge_Red_Oak_ONF	Incomplete name	Modify name	5	No	Achieved
4	Douglas_Douglas_ONF	Incomplete name	Modify name	5	No	Achieved
5	Erable_Sycomore_Sycamore_Plane_ONF	Wrong name	Modify name	5	No	Achieved
6	Other_ONF	Semantic Inconsistency	Removing one of the axioms	10	No	Achieved
7	Sapin_Pectiné_Silver_Fir_OR_Common_Fir_ONF	Confusing name	Modify name	9	Yes	Achieved
8	Coniferous_SERTIT	OWA about the subclasses	Delete subclasses	4	No	Achieved

Table 4. The ONF and SERTIT discussion threads summary: Collaborative development of ONF and SERTIT ontologies

Due to the COD results on ONF and SERTIT ontologies, some of the discussions were about the completeness of concept name. This incompleteness can result in wrong knowledge-based classification. This kind of error is related to the functional parameter, which is not possible to be distinguished through the reasoning services [12]. One of the discussion threads is illustrated in Figure 16:

Figure 16. A discussion thread about `Other_ONF` class

Due to Figure 16 and Figure 17, one of the collaborators has annotated `Other_ONF` class concept by GoogleMap annotation type existed in proposed COD. In the left side the classes are visualized in Jambalaya plug-in and in the right side the collaboration plug-in helped the participants to discuss about this concept. This concept has vague definition which makes a semantic inconsistency for both human and machine. The Pellet reasoner could spot and represent this inconsistency for the collaborators (see Figure 18). Through the collaboration, one of the areas related to this concept (an instance of `Other_ONF`) that is not considered as the forest is represented in Figure 17. The image was captured by a camera with a built-in GPS. The Coordinates show that the image is geotagged (images were uploaded by Google Picasa on Google maps).

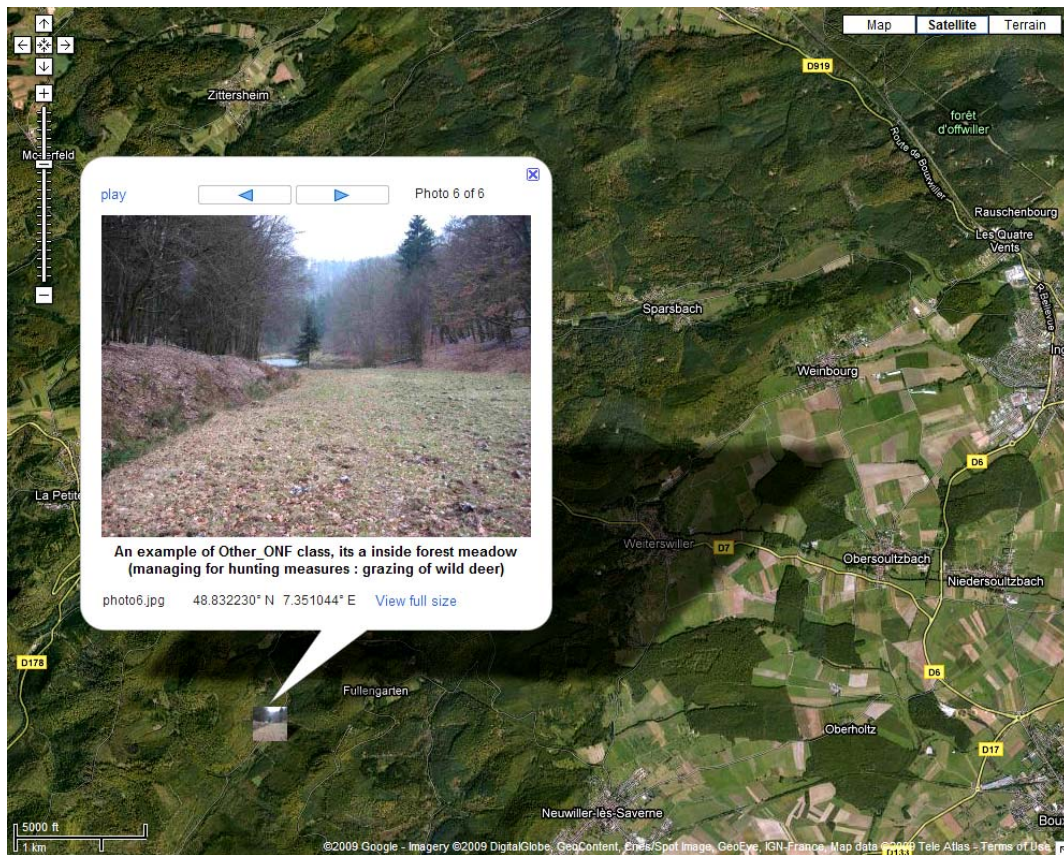


Figure 17. The use of Google maps and geotagged images in COD

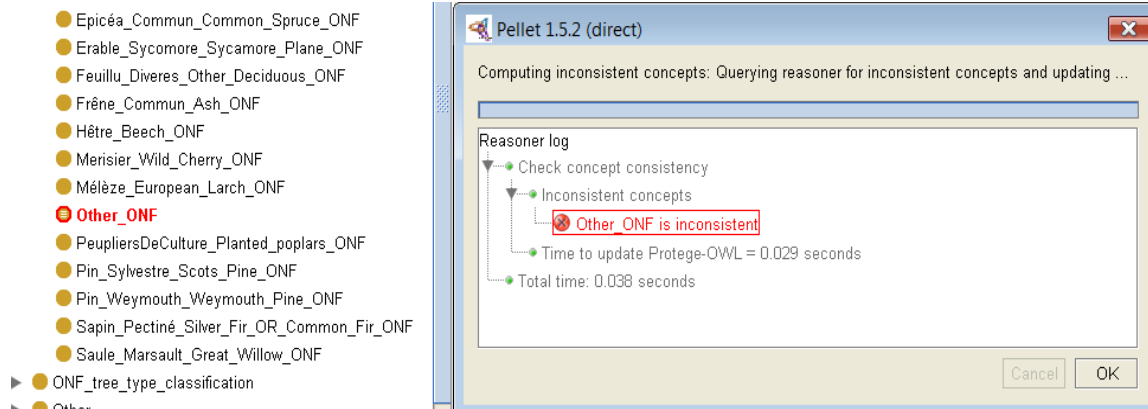


Figure 18. Inconsistency checking by the Pellet reasoner in Protégé; Here the `Other_ONF` class is semantically inconsistent

4.3.2. Step four: Developing the collaborative shared ontology

After developing the ontologies of ONF and SERTIT collaboratively, the shared ontology was developed collaboratively upon these two ontologies. The process was like the development of centralized shared ontology in Section 4.2.2. As the main structure of collaborative ONF and SERTIT ontologies were not different from the catalyst ONF and SERTIT ontologies, the Catalyst shared ontology was adopted for the initial point of collaboration (before refinement; see Figure 15). It was because of evaluating the results of centralized and collaborative refinements (maintenance). Through the collaboration of participants on this ontology one discussion thread occurred (see Table 5).

No	Concept	Topic	Decisions	Frequency of conversations	Rejecting any idea	Consensus
1	LarchTree_cyc	Semantic inconsistency (subclasses of 2 disjoint classes)	Changing the axiom and it becomes the subclass of one class & changing two ontology concepts' names to have broader meaning as the bridge concepts	7	No	Achieved

Table 5. The discussion threads summary: Collaborative development of shared ontology

With respect to Table 5, the inconsistency related to Larch tree was the topic of discussion. The result was different from the centralized methodology. The collaborative shared ontology is illustrated in Figure 19:

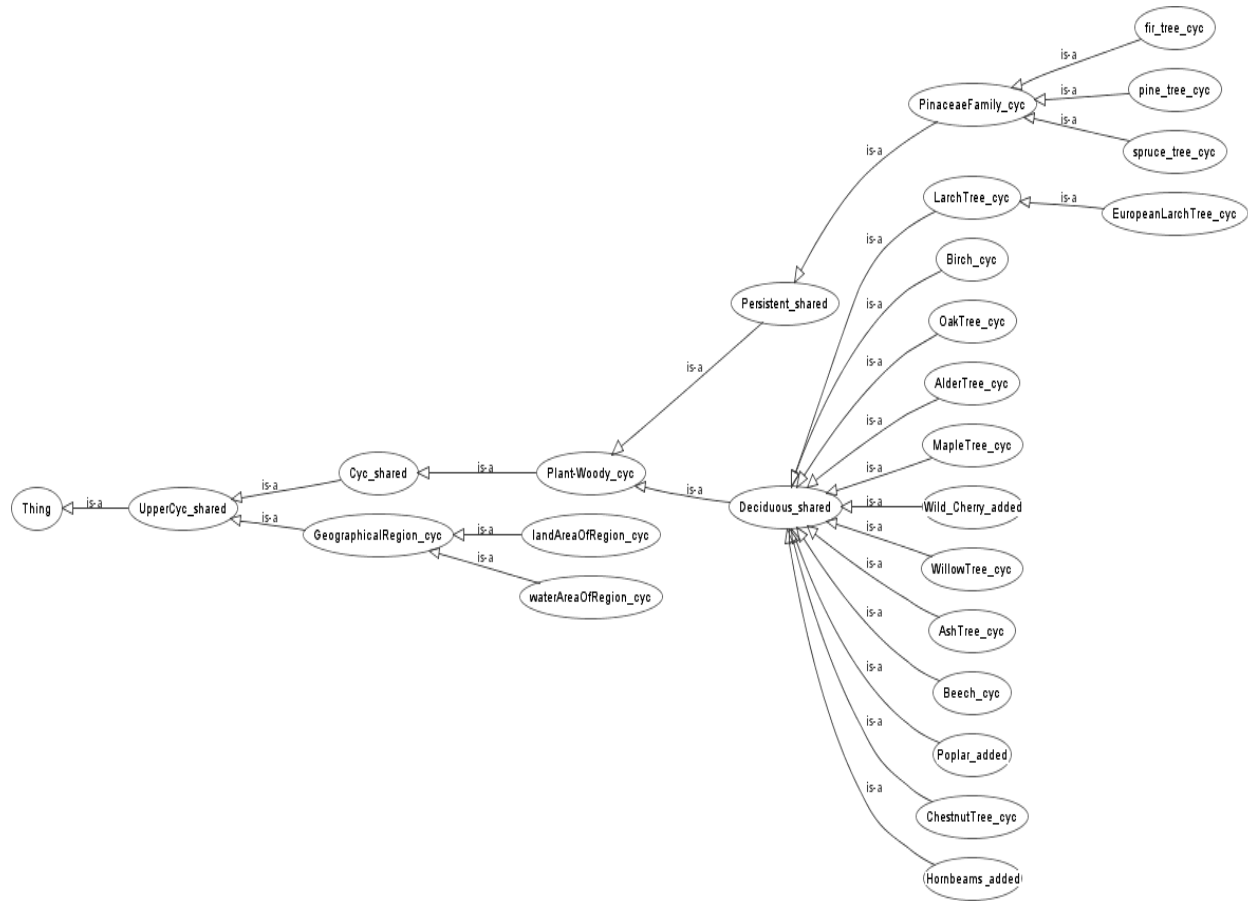


Figure 19. The collaborative shared ontology upon the ONF and SERTIT ontologies; OWLViz plug-in in protégé is used for this tree representation of shared ontology

4.4. The difference between collaborative and centralized methodologies

Corresponding to four described steps of ontology development in Sections 4.2.1, 4.2.2, 4.3.1, and 4.3.2, two collaborative and centralized methodologies are implemented in detail. At the end, the main difference was distinguished in *LarchTree_cyc* concept (the other differences were related to the completeness of concept names). Through the collaborative discussions, it was determined that Larch tree is both of coniferous and deciduous types. The problem was related to the denomination of two concepts: coniferous and deciduous. Corresponding to the discussion of domain experts, coniferous and deciduous were replaced by Persistent and Deciduous concepts. It was because of SERTIT definitions as the main factor to be coniferous and deciduous is the “leaf shedding”. A coniferous tree can be deciduous tree, but, the tree that has persistent leaves cannot be deciduous. As the Larch tree does not have persistent leaves, it becomes deciduous after the refinement phase while being coniferous in centralized shared ontology. In the centralized state, not being possible to engage the domain expert in the maintenance phase, ontology engineer couldn’t spot and resolve this problem.

4.5. The Support activities

Regards to METHONTOLOGY, a number of supporting activities should be done such as Knowledge Acquisition, Integration, Evaluation, Documentation, and Configuration management in parallel to the development phase. There is no need to discuss more about KA in this Chapter. Related to integration, no useful ontology was found to be reused in this research. The other activities are discussed in the following shortly:

Versioning (for configuration management)- one of the advantages of METHONTOLOGY is its versioning consideration which helps for periodic editing and archiving of ontology (one of the COD requirements). However, the different versions should only be accessible for the administrator while being accessible for the other roles via requesting from the administrator (if needed). In this research, two versions for the collaborative ontologies were developed (version.1 and version.2). After refining the ontology in each version, the list of changes and modifications was imported in the new version's metadata.

Evaluation- as the classification systems restricted this research's ontologies to have determined and fixed structures and also the use of Protégé (capable to find grammatical and OWL related errors) the technological, structural and conceptual parameters were not concerning. But the functional and usage related parameters were the targets of this research; as both of them are related to the user feedbacks. In functional assessment besides the user feedbacks, Janowicz, Maue et al [23] have proposed a methodology relying on Similarity Measurement. This methodology was not applicable for this research as in ONF ontology, the concepts are tree-species (all of them are trees, so they are completely similar and not possible to be ranked). The only parameters to evaluate the ontology in this research were inconsistency checking and user's feedbacks.

Documentation- After each phase of methodology, the outputs are archived and documented such as OWL files, KA achievements in COE and etc.

4.6. Summary

In this Chapter two different methodologies to integrate geo-information were evaluated. First, this integration was done through the development of a shared ontology which was built in a centralized methodology. Then, this shared ontology was created through the collaborative methodology. The difference between these ontologies illustrated that the collaborative methodology has better results rather than the centralized methodology.

Chapter Five

Discussions, Conclusions, and Recommendations

In this research a new methodology to develop ontologies has been proposed. The methodology was enhanced by Google maps and the geotagged images. And the application of this methodology is for geo-information sharing among GI communities. In this Chapter the results, the deficiencies, and future works of this research will be discussed. To have more efficiency, it has been tried to bring a specific recommendation after each discussion item. At the end, these recommendations will be generalized in Recommendation Section 5.3.

5.1. Discussion

5.1.1. Geo-information sharing, obstacles, and solutions

The problems for the cooperation of geospatial communities are: syntactic, structural, and semantic heterogeneity. This research has focused on the latter one as the syntactic and structural ones are resolved by OGC and ISO standards [8]. During this cooperation for geo-information sharing and integration, the context knowledge should be explicated at the first step. The simplest solution is the use of text documents (in Natural Language), tables, and also oral conversations among the communities' members. The actual example of this situation was existed in the case-study of this research. In this scenario, ONF and SERTIT communities had an agreement (through their communications) about the terminology of shared information. For instance, when they were talking about the “coniferous” term, there was no any vague point about the meaning of this term among them. As a result, the geo-information integration between these communities appeared a very simple task. So, it is sufficient that the data provider (ONF) adds a metadata about its layers and mentions that e.g. Larch tree is coniferous and etc. But, in reality this kind of tree is deciduous corresponding to SERTIT's terminology. Not explicating the interpretation of “coniferous” (better to say semantic of this term) resulted in this kind of conflict. The right solution was the use of ontology-based geo-information sharing approach [4]. Actually other solutions for data integration such as database and XML schema mappings exist which do not explicate the semantics of information. As the main scope of this research was ontology engineering, it was not possible to test these solutions (just referring to literature).

As discussed in Section 2.2.1, ontology-based geo-information sharing (integration) could be done in two ways: using ontology mapping techniques and using shared ontologies. Referring to literature [4], the second way was adopted for this research because of ontology mapping bottlenecks such as different

granularity and aggregation of ontology concepts. Also in this research, adopting shared ontology-based methodology had another advantage. In this way, the collaborative engineering of both source and shared ontologies could be tested. However, it is assumed that testing collaborative engineering can bring about interesting results in the case of ontology mapping solution. The objectives of this research did not let to test this solution too.

Although in this research no suitable ontology could be found for the sake of reusing (to build both source and shared ontologies), totally the solution of shared ontologies hinders the advantage of reusing other ontologies in this scenario.

Recommendations

Corresponding to the results of this research, it will be advantageous that the results of collaborative shared ontology-based information integration are compared with the results of the ontology mapping-based information integration [13]. In scenarios that ontology mapping can be used, it is a better choice (cost effective) rather than use of shared ontologies.

5.1.2. The development of ontologies: linking collaborative to centralized

Developing ontologies (source or shared) can be done through two methodologies: collaborative and centralized. The previous works were discriminating between these methodologies like [25], but this research has considered these methodologies as two complementary components (see Figure 8). Collaborative ontology engineering includes development of an initial ontology (catalyst) in a centralized methodology and the collaboration (of OE, DE, end-user) extension which is called COD (see Section 2.3.2). In this architecture, COD can be considered as a kind of plug-in for the current centralized methodologies. Actually none of these methodologies has priority over the other one. Although the risk of ontology engineer or domain expert's inaptitude, not considering end-user feedbacks, having static ontologies, and etc have been introduced as the drawbacks of centralized methodology in Section 2.3.1, collaborative methodology does not make sense when there is no need for the maintenance of ontology. Because of maintenance cost (economy and time) the collaborative ontology engineering can be useless. Consequently the evaluation of ontology should determine the need for COD to be added to centralized methodology. It is necessary to mention that there are a few related works that have different perspective about the collaborative ontology engineering. They believe that ontology should be developed just collaboratively such as [39] while claiming : “anybody can add a new element to the ontology, and refine or modify existing ones”. However these works try to build lightweight ontologies (simple taxonomies) in a wide range of collaborators (including all of the roles).

The main challenge is that how the domain experts and end-users can be engaged more through the COD. This collaboration needs two things: methodologies and tools. A methodology was defined in Section 3.2.2 and the tool supporting this methodology was adopted in Section 3.2.3. However the determination of methodology and tool should be done with respect to some touchstones. These touchstones are COD requirements which are determined in Section 3.2.1 through the literature review. Some of these requirements are exclusively for methodology such as consensus and jury and some for the tools such as robustness and reliability. Among these requirements the workflow of collaboration and the consensus achievement are the targets of related works such as [24, 25, 49].

Recommendation

In this research COD requirements are determined in a complete framework. Workflow and consensus and also visualization of ontology (expressivity) are key factors. The future works related to collaborative ontology engineering should consider these factors.

5.1.3. Catalyst ontologies

Two different methodologies were applied to develop the catalyst ontologies in this research: METHONTOLOGY for the source ontologies (ONF and SERTIT catalyst ontologies), Stuckenschmidt and Van Harmelen methodology to build the shared ontologies (see Figure 12). METHONTOLOGY was adopted among some other methodologies. Its maintenance and evaluation phases showed that this methodology has capability to be linked with COD. It could be possible to apply On-To-Knowledge instead of METHONTOLOGY as it supports both maintenance and evaluation phases. As these methodologies are similar in general perspective, it is assumed that On-To-Knowledge can be linked with COD properly. The nature of ontologies (fixed hierarchies and determined concepts) in this research did not emerge to apply different methodologies for evaluation. For example, it could not be possible to apply OntoClean [50] to evaluate the structure of ontology. Also [12], most the methodologies for evaluating the ontologies have their own difficulties and problems. For instance, there is no any specific threshold for a good or bad quality ontology related to some of these indicators. So, in this research only machine-based (reasoner) evaluation of ontologies (related to the functional parameter) and the user feedbacks are considered.

Although collaborative ontology engineering could enhance the quality of shared and source ontologies in this research, the relationship between collaborative ontology engineering and each of quality indicators is not determined.

Recommendation

As the METHONTOLOGY was adopted for the development of catalyst ontologies, other researches can apply On-To-Knowledge too. Also, the other methodologies which have mentioned the phases of evaluation and maintenance can be linked with COD.

Also the methodology of Stuckenschmidt and Van Harmelen was adopted due to the experience and literature review [4] of its developers. It is better that the other researches compare the capabilities of this methodology with other methodologies to build the shared ontologies.

It can be advantageous that the other researches focus more on existed evaluation techniques. The advantages of collaborative ontology engineering by comparing with these indicators (not only inconsistency checking) can be determined in future works.

5.1.4. COD methodology

Corresponding to the determined COD requirements and the deficiencies of previous works, a new methodology being proper for COD was proposed in Section 3.2.2. In below, this methodology is evaluated in the context of ForeStClim project corresponding to COD requirements (see Section 3.2.1):

R.1) Integration, representation, and storing of discussions and annotations and changes in ontology development:

Every discussion was started by an annotation. If the discussion was resulting to a change, the change was done in the next version of ontology. Representing and storing the discussions, annotations, and changes (in new versions) helped the other collaborators to become familiar with the discussion threads and the state of ontology.

Related to the annotation types, firstly some of these types were interesting for the collaborators. As there was no any restriction on the collaborators to use annotation types, they were using them with respect to their idea. The use of ArrivedVerdict (just by administrator), Example (though it was not used), Google Map, Agree Disagree Proposal, See Also, and Question types were understandable for them. But they were misusing Comment and Explanation and could not distinguish the difference between them. For instance, in one of the discussions they answered a Question by Comment annotation type (instead of Explanation). However the discussion continued well and this shows that not restricting the discussions and use of special annotation types do not make problem for the discussions. This is because of human interactivity and understanding the intention of other persons.

R.2) Expressiveness and ontology visualization:

The expressiveness and different visualization of ontology for the OE is not problematic. The main concerns are about the domain experts. As this research has proposed the use of NL expressions and tree view of ontology for the DE, the following results are considerable:

As the created catalyst ontologies of ONF, SERTIT, and shared were in OWL DL (to have reasoning), the translation of asserted conditions to NL sentences helped the non ontologist (like domain experts) to understand these conditions easily. However, it is better to have automatic translation while the current machine-based translators like ACE are not capable to translate complicated conditions [40]. Consequently, the use of more expressive ontology does not make the collaboration more difficult, if and only if the conditions can be translated into NL carefully and non-ambiguously.

R.3 & R.4) Continuous editing/Periodic editing:

This research did not adopt continuous editing to have versioning. Considering this policy, two versions of ontology were stored in the ontology repository (related to collaborative ontologies of ONF, SERTIT, and the shared ontology upon them). The problem of versioning is its metadata, which should be prepared carefully in the case of different changes in new version.

R.5) Jury:

The role of administrator in this methodology is not like Holsapple et al, Karapiperis et al, and DILIGENT methodologies. He could control discussions while not dictating some predefined tasks and personal decisions for the collaborators. This caused to control the process of ontology engineering while taking the optimum advantage of participants' collaboration. All of the collaborators claimed that the role of administrator from registration to the management of workflow was beneficial for this case-study. In one of the discussions, the administrator could steer the discussion when the collaborators was chatting together about non-related topics.

R.6) Evaluation and checking inconsistencies:

The results of this research are achieved through the use of inconsistency checking.

R.7) Provenance of information:

As each collaborator logs in with his username and password (related to his profile), for each annotation or discussion the name of the collaborator was attached. This helped the collaborators to become familiar with the source (a person who creates) of each event in collaboration and have more insights in their discussions. This is illustrated in Figure 16.

R.8) Scalability (size of users):

This requirement was not possible to test in this case-study as the number of collaborators wasn't huge. But, in the proposed methodology the new collaborators can enroll the group (via administrator's permission) while the time to reach to the consensus will be increased and the decision will be more precise.

R.11) Consensus:

In this case-study some ideas and discussions were exchanged among the collaborators. The final conclusions and decisions were consensual and nobody stopped the collaboration (even when his idea had been rejected). It is assumed that by increasing the number of collaborators, the challenge among the participants will be increased. In these situations, the administration should be more careful about the discussions and emotional aspects of collaborations.

After the process of COD, the participants are asked about the methodology to reach to the consensus. The questions were:

- Was the process of consensus achievement troublesome for you?
- Could you cope with the situations when your idea was disapproved?

As the collaborators were aware of the rationale behind the workflow for the consensus achievement, they did not have trouble with workflow and the disapproval of their ideas. However, this workflow should be tested in big communities to show its capabilities.

R.12) Access control:

Both ontology editors and the domain experts could only read the ontology components, but not change them (the delete, edit tabs were disabled when they were connecting). Also they could annotate and participate in discussions too. The administrator could do everything such as read, write, run or shut down the server, kill the other participant's Session (in this case study, it did not happen), start/stop remote projects (not happened). Through these policies for different roles, the control and security of ontology components were guaranteed in this research.

R.13) Workflow support:

Due to the proposed workflow, every participant was registered by the administrator. Two different scenarios happened while collaborating:

- The collaborators logged in the system and reviewed the history of changes and annotations (by using the collaboration tabs and its search facility). Once, one of the participants did not like to participate and logged out the system. In other cases, the participants were interested to enroll the discussions after reading the discussion threads.
- The participants were online and ready to do the collaboration. Most of our discussion threads took place in this state.

In both scenarios, the consensus was achieved by the collaborators. After reaching to the consensus, the administrator announced the final verdict as the end of discussion threads and making decision. Nobody left the group or object after discussions. However in the big communities, some objections may happen from some of the collaborators. In these situations they should follow the proposed workflow to respect to the community and consensus decision making (we have discussed this workflow before).

This workflow lacked the mechanism for the notification of participants. They were supposed to check the ontology or be connected to the server consecutively or schedule and organize the discussion time before it.

R.14 & R.15) Synchronicity and Asynchronicity of discussions:

Also in this case-study, every change or annotation by a client was represented to the other clients instantaneously. The collaborators did not have problem with synchronicity. When they were asked about the asynchronous case, it did not make difference for them.

Recommendations:

Although the use of GoogleMap annotation types and taking the advantage of Google maps and geotagged images is exclusive for Geo-Information sharing, other domains such as biomedical engineering can take the advantage of COD. For instance they can apply other kinds of images (not satellite images!!) related to their domain and studies.

The translation of ontology into NL expressions was an innovation in this research. In contrast to this research, future works should try to enhance this process automatically. From the literature ACE wiki [40] community are frontiers in this field (until the date of doing this research³⁶).

Also supporting COD with notification methods like sending email to the collaborators every day, week, and month is necessary in future work which this research lacked it.

5.1.5. Protégé

Also Protégé and some of its plug-ins and extensions such as Collaborative Protégé, Client Server Protégé, Jambalaya, and OWLViz supported COD are evaluated in the context of ForeStClim case-study.

R.1) Integration, representation, and storing of discussions and annotations and changes in ontology development:

Using flags for the representation and special ordering of discussion threads (by attaching the time of discussion) is user-friendly in Protégé. But the problem is related to the changes which should be represented in new versions in NL text. In the case of numerous changes, it becomes difficult for the

³⁶ November 2009

administrator to write all of them. Actually collaboration plug-in has *changes* tab which has not been used in this research as it is not understandable for the domain experts. Also, all of discussion threads, annotations will be removed if the concept is deleted or renamed in the new version. Related to the storage, all of the discussions, changes and annotations are stored in `annotation_ONF_SERTIT` ontology without any problem.

R.2) Expressiveness and ontology visualization:

The visualization of ontologies by trees (when they have hierarchical structure) was understandable for the participants. The use of Jambalaya and OWLViz plug-ins to represent the ontology in different ways were pleasant for the collaborators. One of these collaborators enumerated the following specifications of these plug-ins:

- Nested View- good for comparing different classes and their contents quickly. Its dynamic movement (animation) from more general classes to the specific classes is interesting. It is like the process of human thinking while thinking from general things and focusing on them in detail.
- Nested Treemap- not being user-friendly.
- Tree views- understandable visualization of ontology concepts, but being messy when the ontology has many classes (see Figure 20). To cope with this problem, the use of OWLViz plug-in was appreciating to visualize ontology in a tree view (see Figure 19).

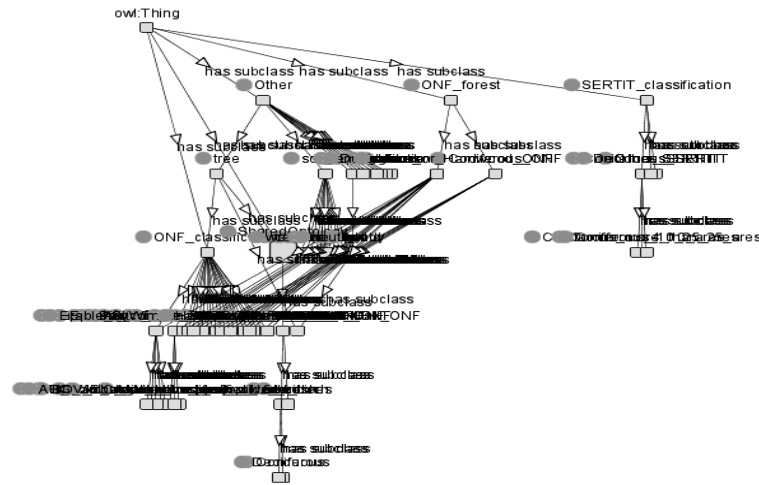


Figure 20. The problem of Jambalaya for tree visualization when the ontology has many classes

R.3 & R.4) Continuous editing/Periodic editing:

The important thing here was the support of metadata to explain the specific versions and their characteristics. This was supported by Metadata Tab in Protégé. Also exporting different versions of ontology by “Archive Current Version” in Protégé was its interesting capability for versioning.

R.6) Checking inconsistencies:

The power of Protégé to support inconsistency checking through Pellet reasoner was helpful to find the inconsistencies in this research. However, other reasoners could have different results in this research which are not tested.

R.7) Provenance of information:

The search capability of Collaboration tab via filtration of non-related information helps for finding the provenance of any change or annotation. In this case-study, the collaborators did not use this search (they were aware of this facility but preferred to use “All notes” tab in collaboration part, to see all of the annotations). It can be because of the number of discussion threads which were not numerous.

R.8) Scalability (size of ontology):

This requirement was not possible to test in this case-study as the ontology was not a big one. However, it was tested for other big ontology such as CORINE ontology without any problem.

R.9) Reliability (not to lose data):

Protégé was reliable and no data was lost in the process of ontology engineering.

R.10) Robustness:

It means that the Protégé should be robust enough as well as other tools applied by the user. The collaborators did not mention anything about this issue.

R.12) Access control:

The collaboration plug-in enforced the collaborators to login to the server. Not any bug and problem happened for the collaborators when interacting with ontology. Also, the considered policies such as not editing facility for DE were completely supported by Protégé and collaboration plug-in. The problem of Protégé is that the other clients can change the contents of other clients' comments (unsecure). In this case-study the collaborators did not make this fault as there was no any wicked manner.

R.13) Workflow support:

The most significant deficiency in Protégé is workflow support. This platform does not support any workflow. It also lacks the support for email services which is essential for notifying the collaborators. However, the proposed workflow in this platform could be tested manually by the collaborators.

R.14 & R.15) Synchronicity of discussions:

The client-server architecture and also the collaboration APIs of Protégé helps for having synchronous editing and discussions.

Recommendations

- The representation of changes in this platform should be more user-friendly. It should be enhanced via visualization techniques like using PromptViz plug-in.
- For visualizing ontology, the combination of Nested View in Jambalaya and Tree view in OWLViz can enhance the user's understanding of ontology.
- Related to the access control, the other users could change the contents of discussion thread. This is a deficiency in collaborative Protégé that should be resolved for the next versions.
- Not possibility to define specific workflow in Protégé can be a motivation for future works.

5.1.6. The role of Google maps and geotagged images in COD

As an innovation in this research, the use of Google maps and some geotagged images related to one concept helped to resolve the semantic vagueness and inconsistency. The semantics of `Other` class was not clear for some of the collaborators. As one of the collaborators had the experience of working on that area, he could bring some examples (individuals) of this area on the Google map (see Figure 17). The use of this technique was called “Spatial Image Based learning” by one of the collaborators.

Recommendation:

Developing Google map plug-in for the Protégé can be useful for geo-information communities.

5.1.7. User feedbacks

As the period of time for doing this research did not let to wait for more user feedbacks, not any deficiency was observed from end-user. It is anticipated that some deficiencies related to the ontology exist in this research which could not be recognized. This issue needs more time for studying.

Recommendation:

The future works (specially the big applications) should study the nature of user feedbacks and how the administrator can report them to the collaborators.

5.2. Conclusion

Geo-Information sharing (GI integration) among distributed geospatial communities will not be possible if the semantic heterogeneity in these communities is not removed. To resolve this problem, ontologies are used for explicating the communities' knowledge formally. This research proposes an enhanced collaborative ontology development methodology for sharing geo-information.

During the development of this methodology, it is determined that collaborative ontology engineering is composed of two phases: 1. development of a centralized and initial ontology (this is called catalyst ontology in this research), and 2. COD (collaboration extension). This point of view shows that this research does not discriminate between centralized ontology engineering and collaborative ontology engineering. In a nutshell, collaborative ontology engineering is the centralized methodology while adding collaborative maintenance to it.

To define the proper methodologies to be applicable in these phases, the previous works of centralized and collaborative ontology engineering were evaluated. METHONTOLOGY for developing the catalyst source ontology (phase one), Stuckenschmidt' method for developing the catalyst shared ontology (phase one), and COD (phase two) for collaboration are introduced in this research. COD needed a proper platform to be implemented for which collaborative Protégé was adopted. Proposing a new collaborative methodology (COD) and adopting collaborative Protégé were upon the determination of some requirements for collaboration.

Also for the first time, geo-information on Google Maps and geotagged images were used to enhance the understanding of the expressivity of geospatial concepts in the ontology. Because of this breakthrough, the applied methodology is called "an enhanced methodology". This innovation is flexible for other domains such as biomedical engineering where they can apply their own images to enhance the expressivity of ontology concepts.

The process of ontology-based geo-information integration can be done through ontology mapping or creation of shared ontologies. Because of ontology mapping's bottlenecks, the latter one was adopted for this research. In this methodology, first the source ontologies of each community are developed. Then a shared ontology containing bridge concepts among source ontology concepts is developed. In the case-study of this research, two communities were sharing Geo-Information. To make this task possible for them, this research applied a collaborative methodology through the development of source and shared ontologies. The collaborative methodology had better results when the results of this methodology were compared with a centralized methodology (to build source and shared ontologies). The made modifications on ontology (in collaborative state) were related to the incompleteness of concept names and semantic inconsistency.

The results of this research showed that COD can be also advantageous to develop the shared ontologies as well as source (single) ontologies.

5.3. General Recommendations:

Based on the specific recommendations made in Section 5.1 after each discussion item, the general recommendations for the future works are:

It is recommended that the semantic aspects of geo-information sharing are considered by geospatial communities. As discussed in Section 2.2, the communities' ontologies and a shared ontology based on them should be developed. First, the catalyst ontologies of these communities should be built. To develop these ontologies two methodologies have capability to be linked to the collaboration extension: METHONTOLOGY and On-To-Knowledge. During the development of catalyst ontologies, the use of CMapTools COE can speed-up the process of knowledge acquisition (see Appendix 1). At the end of phase one, evaluation indicators should be considered (see Section 2.3) in which inconsistency checking can be the simplest but advantageous one ("If the results of evaluation tests are satisfactory, there is no need for maintenance and doing collaboration. Nevertheless, other researches have shown that some of ontology deficiencies can only be found through the collaboration [12]"). Second, COD extension should be performed during the maintenance phase of previous step (see Section 3.2.2). Then a shared ontology upon these ontologies will be developed through the Stuckenschmidt and Van-Harmelen methodology collaboratively (see Section 4.3.2). The use of Upper ontology such as Cyc can be useful as it helped to find the most of bridge concepts in this research.

In scenarios which implementing the ontology mapping is possible, it is more cost effective rather than developing a new ontology (shared). Also in these cases, the reuses of existing ontologies will speed-up the ontology development process.

Corresponding to the use of Protégé and ESRI ArcGIS separately in this research, the need to integrate these platforms was considered. The semantic layer in GIS & RS softwares can be added upon the stored GIS layers or the spatial databases integrated to these softwares. For instance, each layer can be considered as a class if the layers are the representations of classes in a classification system.

As proposed in this research (Section 3.2.1), COD requirements are the main touchstones for collaborative ontology engineering which should be considered for the related methodologies and platforms. In this research, workflow of collaboration has been noticed as the most important factor which should be considered as the first factor for the future works. After that the visualization of ontology for the domain experts through the natural language expressions and the tree view are important in this process. Other researches can focus on ACE [40] to translate the ontology into natural language expressions. The next factor considered by previous works is consensus [24, 25, 49] which other consensus-based methodologies can be proposed by future works. However, consensus decision making (used in this research) has shown its capabilities in many projects such as Wikipedia.

Through the collaborative development of ontologies in other domains such as biomedical engineering, "Image based learning" can enhance the expressivity of ontology concepts. In these domains, it should be possible to take the image of ontology components.

The ontology editors need to enhance their interface to become more user-friendly for the collaborative ontology engineering. As proposed in this research (see Section 3.2.3), the visualization of ontology by tree

views and natural language translation of ontology components can enhance these tools. Also, supporting workflow implementation is a big deficiency in these platforms which should be done manually.

Appendix 1: Conceptualization phase

In this appendix, the whole process of conceptualization in ForeStClim project will be described. The tool used in this phase was COE which helped to speed-up the process of conceptualization.

Totally the important issues in Knowledge Acquisition are:

1. Before meetings: evaluating the existing knowledge resources such as wikis, DB Schema, Folders, Concept Maps, possibility to do data mining via OntoGen (pdf, text formats). As mentioned before (Section 4.2.1), some of these activities were done before the 2nd meetings.
2. Language: Although OWL full supports the most expressiveness among all of the OWL extensions, OWL DL was adopted as a decidable fragment of FOL to have enough expressiveness supporting the reasoning service to find the inconsistencies.
3. During the meetings: face to face meetings with the domain experts were arranged in this part. In this step, the concepts are imported to the CMapTools COE³⁷ [51] by the domain experts as some nodes (see Figure 21).

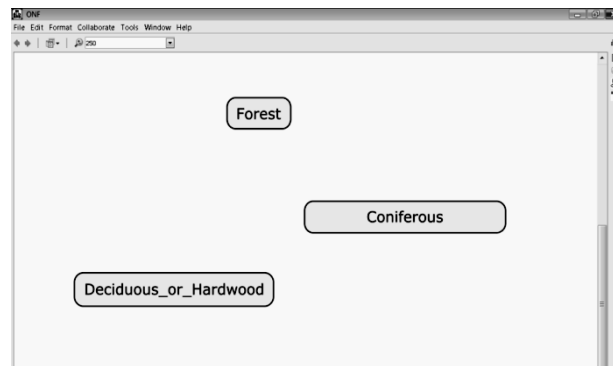


Figure 21. Importing main concepts by the domain experts; COE interface

The concepts are categorized in different categories corresponding to their commonalities. This should be done by the domain experts. The use of different colors for each category is useful (see Figure 22). For these categorization, many methodologies can be used such as card sorting and etc [12] which they were not used in this research (the structure of classification hierarchy was determined and stable).

³⁷ In SWING CMapTools was applied, but in this research we have applied CMapTools COE which support ontologies and OWL import/export.

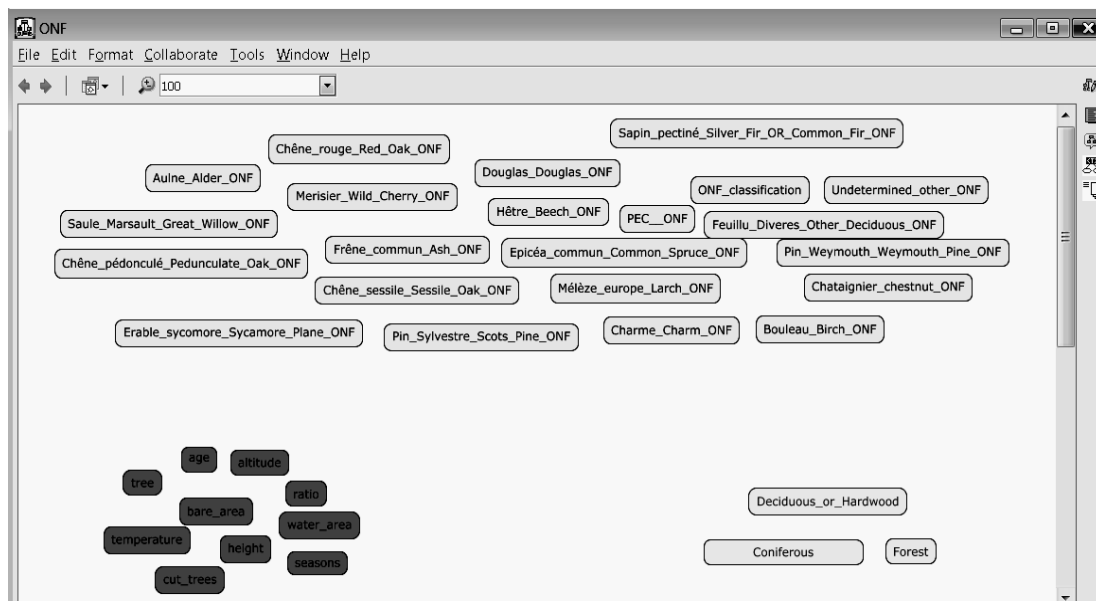


Figure 22. Grouping the concepts due to their similarity; this was easy in this research as the main hierarchy of ontology was determined by the classifications.

Then the class hierarchy was created by defining the superclass and subclass relations. The arc label should be “are” as the notion of subclass and superclass relations in COE. In the case of many domain experts in distributed locations, they can do these tasks in a distributed setting. They can connect to the Cmap Server and cooperate to do knowledge acquisition by annotating the concepts (right click on concept name/Annotate and a tag (see Figure 23) will be attached to the concept showing its annotation and the name of author).

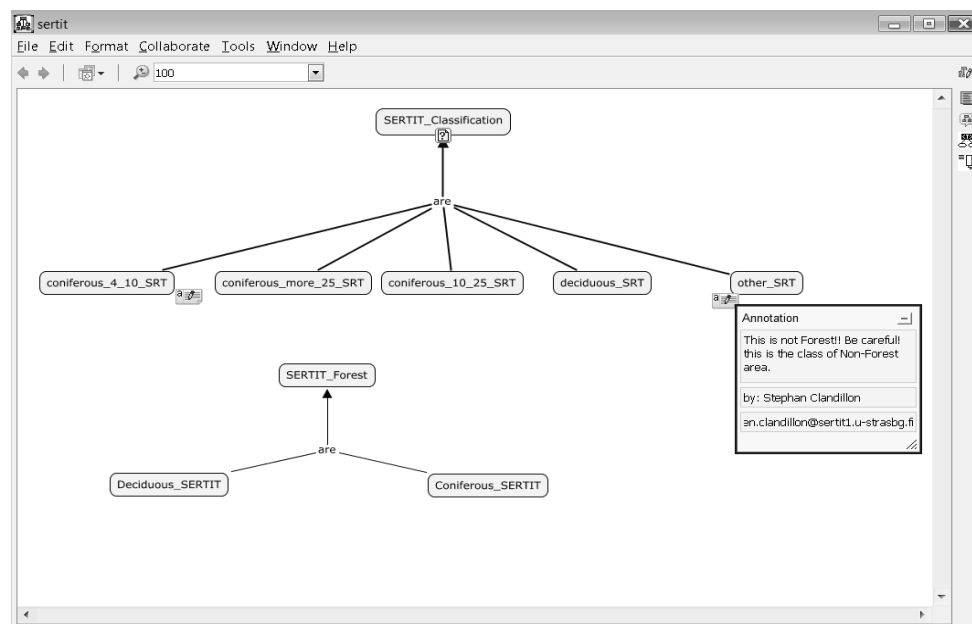


Figure 23. KA in COE; the domain expert can do it in a distributed setting. In this figure one of the domain experts has annotated the concept

After reaching to the consensus by the domain experts about the domain knowledge, the hierarchies in COE environment can be exported to the OWL file (the interoperability between the tools of this research; Protégé and COE). The COE also supports the other features of OWL but it was not intended to engage the domain experts to learn ontology languages such as OWL. However the subclass/superclass relationship is intuitive for them. Due to this fact, other ontology components such as object properties were not created in the Cmap. In SWING project besides the hierarchical properties, the non hierarchical properties were represented in Cmap environment too. The result of adding these properties made the Cmap in this style (see Figure 24):

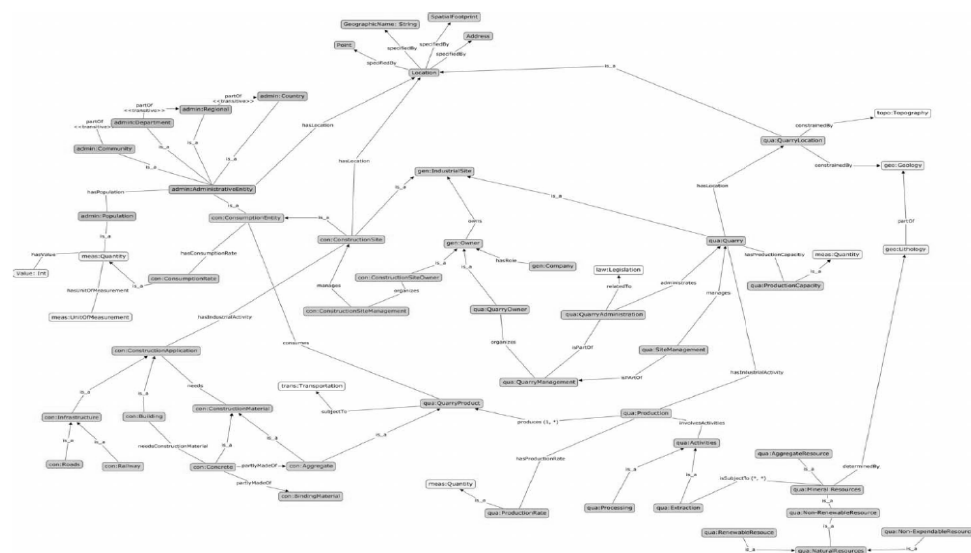


Figure 24. Representation of concepts and the properties among them in SWING [51]

But this research hypothesized that this may make it messy as the domain experts are not familiar with properties (in fact talking about triples and so on). In this research it is claimed that the non ontology engineer persons can conceive the ontology characteristics up to the subclass/superclass relationship. This is because of two reasons: first human familiarity with categories and classifications where a superclass covers the subclass, second human has nodding acquaintance with mathematics and set/subset concepts. Not only the use of subclass/superclass terminology was ignored in this research, but also they are category/sub-category, which are more tangible for the naïve persons. This claim was tested in the KA phase of this research. The domain experts were requested to express their understandings of two different style of representation (See Figure 25). The lower one was considered deceptive and messy in their outlooks.

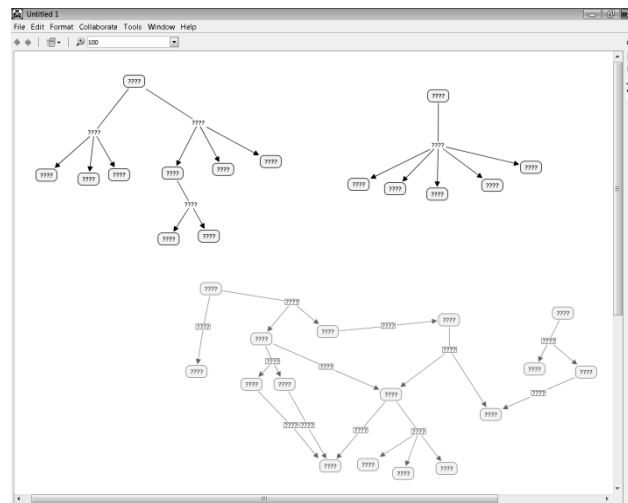


Figure 25. Two different styles of representation; due to users' point of view, the lower one is somehow messy

By the way these properties have been extracted from the sentences concluded in KA phase. For example after the meetings with domain experts, the following definition for coniferous tree was determined: "Coniferous trees do not shed their leaves". "Shed" can be considered as an object property in the ontology. But instead of representing these properties in Cmap, they were stored in their related concepts. By right click on each concept and the use of "Add Info ..." and "Hidden Info:", these properties were described in NL. These NL sentences will be used for the definition of concepts in ontology editor too while importing the corresponding axioms. By these policies, the time of ontology engineering could be saved (see Figure 26).

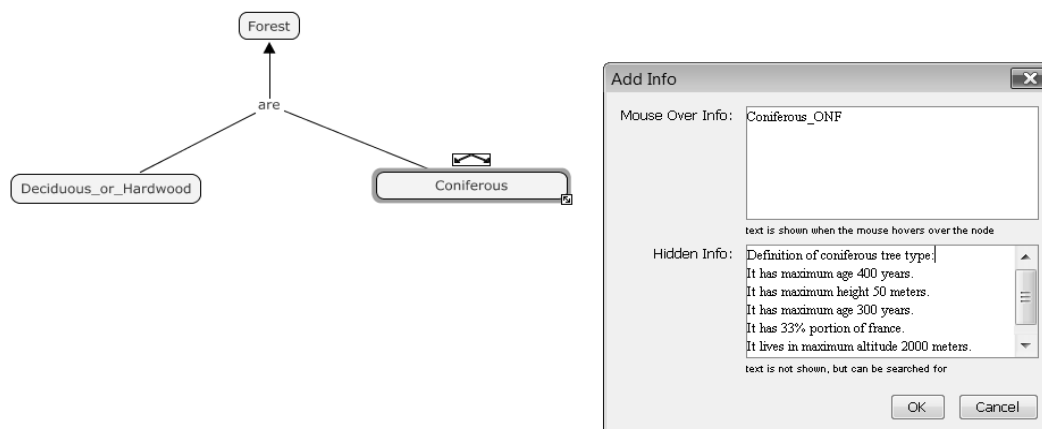


Figure 26. Defining each concept by its properties in NL; although the domain experts were not aware of any property, they understood these definitions well

In SWING project (see Figure 27), the matrices have been introduced for storing and organizing concepts, properties, axioms, and individuals. This will decrease the overlooking of some important ontology components. But this research did not store them in these tables as the ontology was not as big as SWING's one. In this case, the time of KA could be saved without making any mistake in the ontology components.

A	B	C	D	E	F	G
Relation Name	French Name	Domain Concept	Range Concept	Inverse Relation	Domain Cardinality	Range Cardinality
owns	appartient	QuarryOwner	Quarry	isOwnedBy	1	n


```

concept QuarryOwner
  owns inverseOf (isOwnedBy) impliesType (1 *) Quarry
  nonFunctionalProperties
    frenchTranslation hasValue "appartient"
  endNonFunctionalProperties

concept Quarry
  isOwnedBy inverseOf (owns) impliesType (1 1) QuarryOwner

```

Figure 27. Matrices for storing the information about ontology relations [51]

And for storing the individuals in COE, “Add Info” and the “Mouse Over Info” form was used (see Figure 27):

Figure 28. Importing and storing the individuals in COE

Adding additional knowledge resources to COE:

Corresponding concepts in Upper Cyc Ontology, the Wikipedia pages, and the GIS layers related to each concept were added to the concept maps. They were done by using “Add Web Address...” for web pages and “Add & Edit Link to Resources...” for linking to the other softwares or resources (see Figure 29, Figure 30, and Figure 14).

Totally this research could import and store all important and collected knowledge in COE. Also its capability to interlink to the other softwares and platforms and its capability to export to OWL format increased the speed of conceptualization phase.

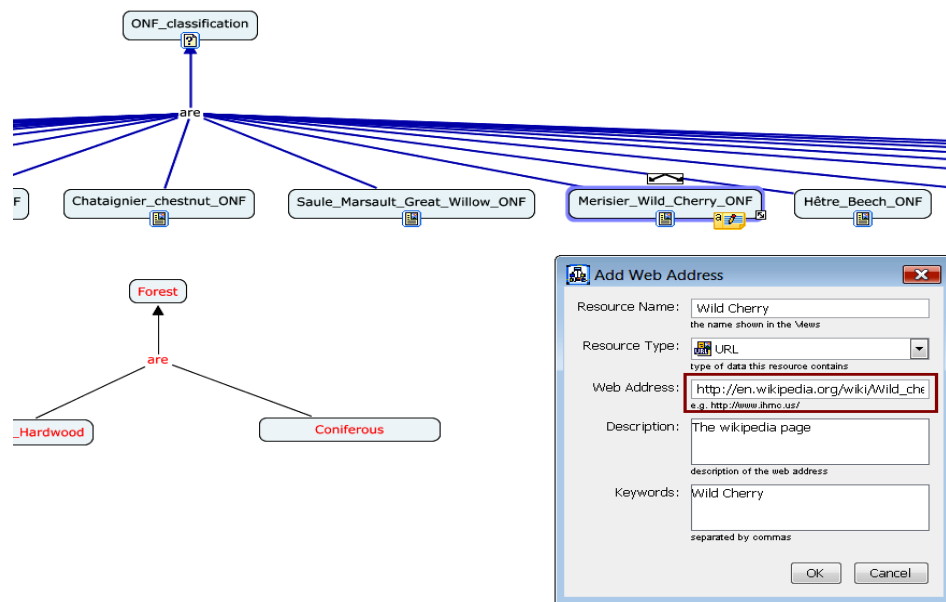


Figure 29. Adding web URL of related Wikipedia pages in COE increased the speed of conceptualization phase

The figure shows the COE interface with a Wikipedia page for 'Wild Cherry' and an OpenCyc ontology entry for 'wild cherry'. The Wikipedia page is on the left, showing the title 'Wild Cherry', a description, and a photo. The OpenCyc entry is on the right, showing the title 'OpenCyc Collection: wild cherry', a unique ID, English ID, English aliases, and a description. The OpenCyc entry is highlighted with a red box. The COE interface is visible in the background, showing the 'ONF_classification' and 'are' nodes, and the 'Merisier_Wild_Cherry_ONF' box.

Figure 30. Adding web URL of related Wikipedia pages and Cyc ontology in COE increased the speed of conceptualization phase

Appendix 2: Client Server Protégé

Being connected to the Protégé Server simultaneously, multiple clients can access to ontology with different levels of accessibility.

Setting up the Server- To start the Protégé server two ways are proposed such as:

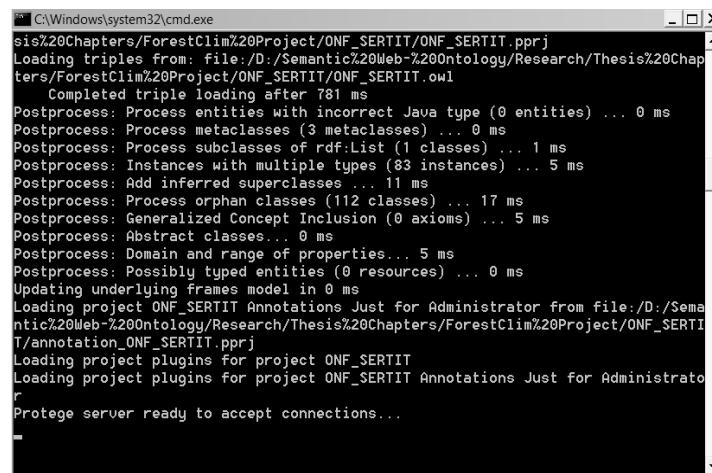
- Running a batch file in the directory of installation; the name of this script is: run_protege_server.bat on windows OS and run_protege_server.sh on Linux and MAC OS.
- Or the advanced running: firstly starting a background process ("rmiregistry") as a part of JRE³⁸ and secondly invoking a java program to start the server:

1. In console write: start /min jre\bin\rmiregistry
(The directory should be the directory of Protégé installation)
2. Starting the Protégé server by the following command in console:

```
"jre\bin\java -Xmx200M -cp protege.jar;looks2.1.3.jar;unicode_panel.jar -
Djava.rmi.server.codebase=file:/c:/program%20files/protege_3.4.1/protege.jar
edu.stanford.smi.protege.server.Server examples\server\metaproject.pprj"
```

It should be mentioned that this command consists of some parameters for starting the server which will be discussed in server configuration part.

A message like Figure 31 should be appeared in console:



```
C:\Windows\system32\cmd.exe
sis%20Chapters\ForestClim%20Project\ONF_SERTIT\ONF_SERTIT.pprj
Loading triples from: file:/D:/Semantic%20Web-%20Ontology/Research/Thesis%20Chap
ters\ForestClim%20Project\ONF_SERTIT\ONF_SERTIT.owl
Completed triple loading after 781 ms
Postprocess: Process entities with incorrect Java type (0 entities) ... 0 ms
Postprocess: Process meta-classes (3 meta-classes) ... 0 ms
Postprocess: Process subclasses of rdf:List (1 classes) ... 1 ms
Postprocess: Instances with multiple types (83 instances) ... 5 ms
Postprocess: Add inferred superclasses ... 11 ms
Postprocess: Process orphan classes (112 classes) ... 17 ms
Postprocess: Generalized Concept Inclusion (0 axioms) ... 5 ms
Postprocess: Abstract classes... 0 ms
Postprocess: Domain and range of properties... 5 ms
Postprocess: Possibly typed entities (0 resources) ... 0 ms
Updating underlying frames model in 0 ms
Loading project ONF_SERTIT Annotations Just for Administrator from file:/D:/Sema
ntic%20Web-%20Ontology/Research/Thesis%20Chapters\ForestClim%20Project\ONF_SERTI
T\annotation_ONF_SERTIT.pprj
Loading project plugins for project ONF_SERTIT
Loading project plugins for project ONF_SERTIT Annotations Just for Administrator
Protege server ready to accept connections...
```

Figure 31. The Protégé server is started

Configuring the Server: Users and Policies-The Metaproject contains the related policies to the projects and server which the clients are connected to them. These policies are essential in the environments where multiple roles collaborate on developing ontologies. The project policies define the permissions for the

³⁸ Java Runtime Environment

clients to access to a project and the server policies regard to the administrative policies related to the server and its control.

Project Policies in this research- in this part some restrictions and policies can be dedicated to some of the clients (roles).

- **Users-** the actual clients who engage in COD (see Table 6):

ID	Name	Surname	User name	Password	Group
1	Rob	Lemmens	Rob	Rob	Admin Group
2	Julien	Prinet	Julien	Julien	Domain Expert
3	Stephen	Clandillon	Stephen	Stephen	Domain Expert
4	Laurent	Gautier	Laurent	Laurent	Domain Expert
5	Reza	Kalbasi	Reza	Reza	Ontology Engineer Admin Group
6	Stephen	Clandillon	Stephen	Stephen	End User

Table 6. COD collaborators profile

- **Groups-** the groups are (due to our proposed methodology in Section 3.2.2):

Admin Group, Domain Expert, Ontology Engineer, and End User

- **Operations-** the following operation can be performed in COD (see Table 7):

No	Name	Operation
1	AdministerServer	A super-right: a user who has this right can kill other users' sections, stop and start server projects without having these rights set for each project individually.
2	DisplayInProjectList	The operation of displaying the project in the project list. This is used to control the projects displayed when a user connects to a server.
3	KillOtherUserSection	The operation of killing another user's section.
4	Read	The operation of reading an ontology project. This controls who can open a remote project.
5	ShutdownServer	The operation of shutting down the server.
6	StartRemoteProject	The operation of starting a (previously stopped) server project.
7	StopRemoteProject	The operation of stopping/closing a running server project.
8	Write	The operation of writing to an ontology project.

Table 7. Collaborative Protégé operations for COD

- **GroupOperation-** Operations related to the groups:

- ◆ AdminGroup can {AdministerServer, KillOtherUserSession, StartRemoteProject, StopRemoteProject, ShutdownServer, Read, Write, DisplayInProjectList}
- ◆ DomainExpert can DisplayInProjectList
- ◆ DomainExpert can Read
- ◆ DomainExpert can Write
- ◆ OntologyEngineer can {Read, Write, DisplayInProjectList}

Figure 32. The defined operations for each group of collaborators in COD; the snapshot has been captured from the Protégé ontology editor

- **PolicyControlledObject-** the superclass of classes which have some policies.

Allowed Group Operations (Policies)	
allowedGroup	allowedOperation
AdminGroup	AdministerServer, KillOtherUserSession, StartRemoteProject, StopRemoteProject, ShutdownServer, Read, Write, DisplayInProjectList
DomainExpert	Read
DomainExpert	DisplayInProjectList
OntologyEngineer	Read, Write, DisplayInProjectList

Figure 33. The COD policies for the interaction of different roles; this shows that domain experts do not have permission to edit the ontology components; the snapshot has been captured from the Protégé ontology editor

- **Project-** the subclass of PolicyControlledObject and includes all the projects which can be collaborated (see Figure 34).

name
◆ ONF_collab
◆ ONF_collab Annotations Just for Administrator
◆ SERTIT_collab
◆ SERTIT_collab Annotations Just for Administrator
◆ Shared Ontology
◆ Shared ontology Annotations Just for Administrator

Figure 34. The available projects for the collaborators; the snapshot has been captured from the Protégé ontology editor

Login to the server- After running the server, the clients can connect to it via their username and password (for security). This is shown in Figure 35:

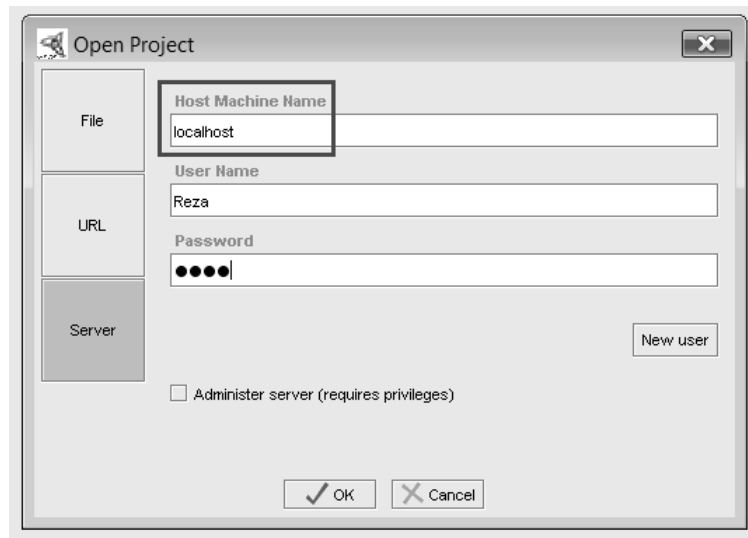


Figure 35. Login to the server by defining the Host Machine Name (here we have considered the local host), username and password

Appendix 3: OWL and RDFS – the example of ForeStClim Source Code

Being the recommendation of W3C, Web Ontology Language (OWL) is designed for web contents which can be processed (instead of being only human readable). The OWL uses other previous layers of semantic web such as URI, XML/S (XML Schema is not an ontology but is a message format) and RDF(S) while more machine interoperability will be achieved through it³⁹. The OWL has three sublanguages which are used corresponding to the amount of needed expressiveness: OWL Lite, OWL DL, and OWL Full. Although RDF(S) is not the sublanguage of OWL, it can be considered as an ontology language with the least expressiveness.

Different requirements that different users have for a Web ontology⁴⁰ [4]:

RDF(S): being useful for generating classification hierarchies as the least expressive ontology. The main components and features of RDF(S) are:

1. Class
2. Subsumption relation (subClassOf)
3. In both of RDF and RDFS, instances are related to other instances through properties
4. SubPropertyOf

OWL Lite: It adds some simple constraints on RDF(S) such as cardinality values of 1 or 0. Its simplicity supports a quick migration path for thesauri and other taxonomies. The main components and features of OWL Lite which RDF(S) does not support them are:

1. Making possible to define equal individuals by **SameIndividual (a, b)**
2. Making possible to define equal classes and properties by **EquivalentClasses (a b)** , **EquivalentProperties (a b)**
3. **DifferentIndividuals (a b)** to determine inequalities for different individuals. Because OWL does not assume UNA (unique name assumption) which means that two different individuals are not considered different until mentioning explicitly.
4. Inverse, Transitive and symmetric properties are supported: **ObjectProperty(a inverseOf(b))** , **ObjectProperty(a : b Transitive)** , **ObjectProperty(a : b Symmetric)**
5. It supports whether a property is optional or required and whether it is single- or multi-valued
6. Functional Property: to define that a property must have maximum one value for an individual
7. More refined version of domain and range restrictions: e.g. **minCardinality** and **maxCardinality**
8. In contrast to RDF(S) that you cannot enforce the range of a property to have “at Least” one value, in OWL Lite it is feasible. For instance in RDF(S) you can use **ObjectProperty(has domain(Forest))**

³⁹ <http://www.w3.org/TR/owl-features/>

⁴⁰ Also a W3C recommendation; <http://www.w3.org/TR/2004/REC-owl-guide-20040210/#Introduction>

range(Tree)) does not mandate Forest to have at least one tree, while in OWL Lite by using following statement the Forest class must have at least one tree:
Class(Forestrestriction(hassomeValuesFrom(Tree)))

OWL DL⁴¹: its intention is to support maximum expressiveness and possible computational completeness (all entailments are guaranteed to be computed) and decidability (all computations will finish in finite time) of reasoning systems. OWL DL includes all OWL language constructs with restrictions such as type separation. e.g. a concept cannot be both a class type and an individual at a same time. A number of additional language's constructs and characteristics of OWL DL are:

1. **disjointClasses(a b)**: to declare the disjointness of two classes
2. **unionOf** and **interSectionOf**: for defining Union and InterSection
3. It provides a construct for negation concept: **complementOf: complementOf(a b)**
4. **oneOf** construct for enumeration: **EquivalentClasses(c oneOf(a1 ... an))**
5. Like enumeration, we can define a property to have a specific value by stating the value: **Class(Ci restriction(a hasValue b))**

OWL Full is meant for users who want maximum expressiveness and syntactic freedom with no computational guarantees and most of the reasoners do not support OWL Full. A number of additional language's constructs and characteristics of OWL Full are:

1. In contrast to OWL Full, OWL Lite and DL are based on a strict segmentation of the vocabulary: no term can be both an instance and a class, or a class and a property, etc. For instance *"Is "747" an instance of the class of all airplane-types made by Boeing or is "747" a subclass of the class of all airplanes made by Boeing? And are particular jet planes instances of this subclass?"* [4]
2. Full RDF(S) is much more liberal: a class C1 can have both a type and a subClassOf relation to a class C2, and a class can even be an instance of itself. In fact, the class **Class** is a member of itself. OWL Full inherits from RDF(S) this liberal approach.
3. OWL Full facilitates the process of ontology integration.



Figure 36. Increasing the amount expressiveness and computational complexity. From left to right, the right one include the left one both semantically and syntactically

Finally, it should be mentioned that other languages such as XOL, SHOE, OML, OIL, DAML+OIL can be used as ontology languages too. Among them, DAML+OIL as the ancestor of OWL is one the most interested languages among web community (because of its mastery to define high expressive ontologies). As an example the OWL DL code related to one of the developed ontologies is this research is represented in the following:

⁴¹ Description Logic

OWL Code example of ForeStClim Ontology

(due to project's policies, a complete source code cannot be represented):

```
...
<owl:Class rdf:about="http://www.owl-ontologies.com/ForeStClim.owl#Coniferous_ONF">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:hasValue>
        <age rdf:about="http://www.owl-ontologies.com/ForeStClim.owl#age_300">
          <has_age rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
            >300</has_age>
        </age>
      </owl:hasValue>
    </owl:Restriction>
    <owl:onProperty>
      <owl:FunctionalProperty rdf:about="http://www.owl-
ontologies.com/ForeStClim.owl#has_min_age"/>
    </owl:onProperty>
  </owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:hasValue>
      <ratio rdf:about="http://www.owl-
ontologies.com/ForeStClim.owl#ratio_33_percent">
        <has_ratio rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
          >0.33</has_ratio>
      </ratio>
    </owl:hasValue>
    <owl:onProperty>
      <owl:FunctionalProperty rdf:about="http://www.owl-
ontologies.com/ForeStClim.owl#has_portion_of_france"/>
    </owl:onProperty>
  </owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:hasValue>
      <age rdf:about="http://www.owl-ontologies.com/ForeStClim.owl#age_400">
        <has_age rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
          >400</has_age>
      </age>
    </owl:hasValue>
    <owl:onProperty>
      <owl:FunctionalProperty rdf:about="http://www.owl-
ontologies.com/ForeStClim.owl#has_max_age"/>
    </owl:onProperty>
  </owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:someValuesFrom rdf:resource="http://www.owl-
ontologies.com/ForeStClim.owl#temperature"/>
    <owl:onProperty>
      <owl:FunctionalProperty rdf:about="http://www.owl-
ontologies.com/ForeStClim.owl#live_in_min_temperature"/>
    </owl:onProperty>
  </owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:onProperty>
```

```
<owl:FunctionalProperty rdf:about="http://www.owl-
ontologies.com/ForeStClim.owl#live_in_max_altitude_meter"/>
</owl:onProperty>
<owl:someValuesFrom rdf:resource="http://www.owl-
ontologies.com/ForeStClim.owl#altitude"/>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Restriction>
<owl:someValuesFrom rdf:resource="http://www.owl-
ontologies.com/ForeStClim.owl#age"/>
<owl:onProperty>
<owl:FunctionalProperty rdf:about="http://www.owl-
ontologies.com/ForeStClim.owl#has_max_age"/>
</owl:onProperty>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty>
<owl:FunctionalProperty rdf:about="http://www.owl-
ontologies.com/ForeStClim.owl#has_min_age"/>
</owl:onProperty>
<owl:someValuesFrom rdf:resource="http://www.owl-
ontologies.com/ForeStClim.owl#age"/>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Restriction>
<owl:someValuesFrom rdf:resource="http://www.owl-
ontologies.com/ForeStClim.owl#height"/>
<owl:onProperty>
<owl:FunctionalProperty rdf:about="http://www.owl-
ontologies.com/ForeStClim.owl#has_max_height"/>
</owl:onProperty>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Restriction>
<owl:hasValue>
<height rdf:about="http://www.owl-ontologies.com/ForeStClim.owl#height_50">
<has_height rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
>50</has_height>
</height>
</owl:hasValue>
<owl:onProperty>
<owl:FunctionalProperty rdf:about="http://www.owl-
ontologies.com/ForeStClim.owl#has_max_height"/>
</owl:onProperty>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty>
<owl:FunctionalProperty rdf:about="http://www.owl-
ontologies.com/ForeStClim.owl#has_portion_of_france"/>
</owl:onProperty>
<owl:someValuesFrom rdf:resource="http://www.owl-
ontologies.com/ForeStClim.owl#ratio"/>
</owl:Restriction>
</rdfs:subClassOf>
...
```

Appendix 4: Centralized ontology engineering methodologies

In this appendix the centralized ontology engineering methodologies are described in brief (By using diagrams).

Uschold and King's methodology [52]:

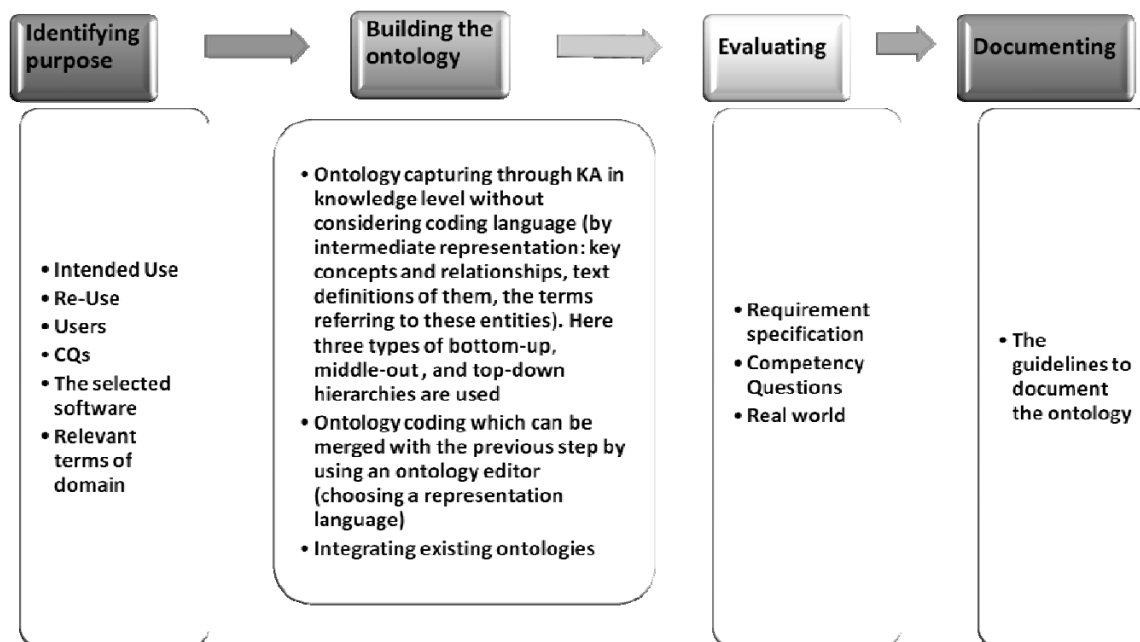


Figure 37.Uschold and King's methodology

Grüninger and Fox's methodology [18]:

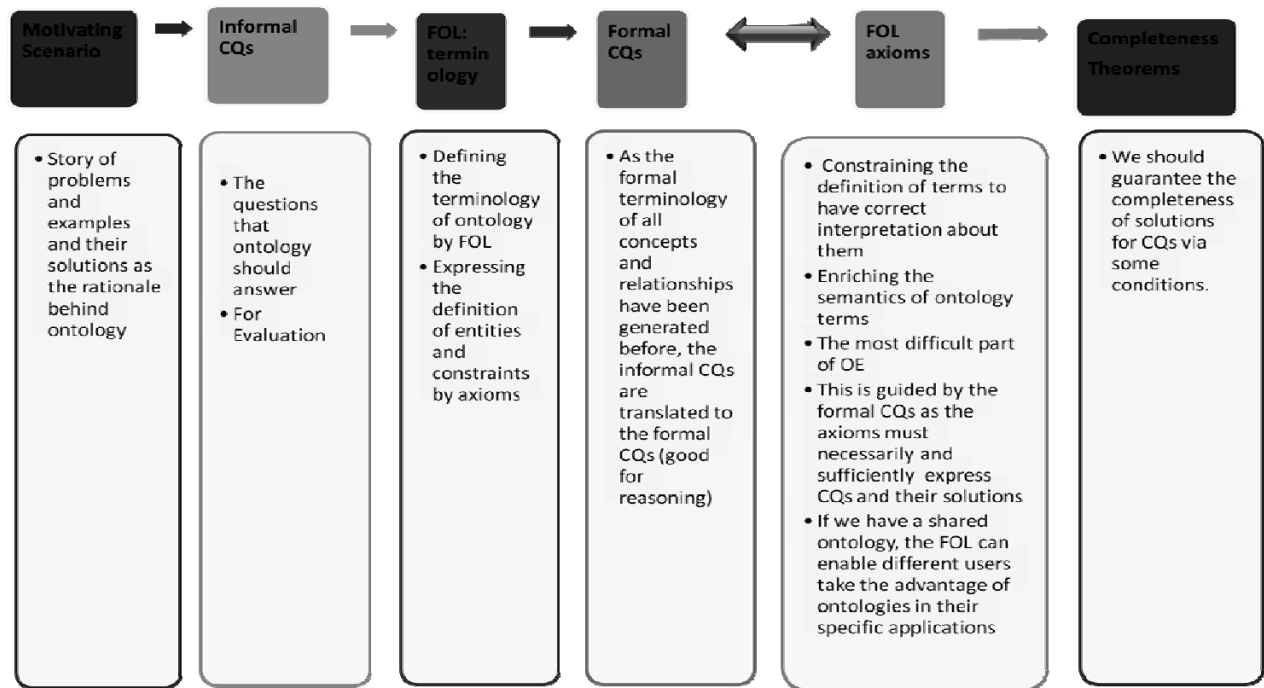


Figure 38. Grüninger and Fox's methodology

On-To-Knowledge [53]

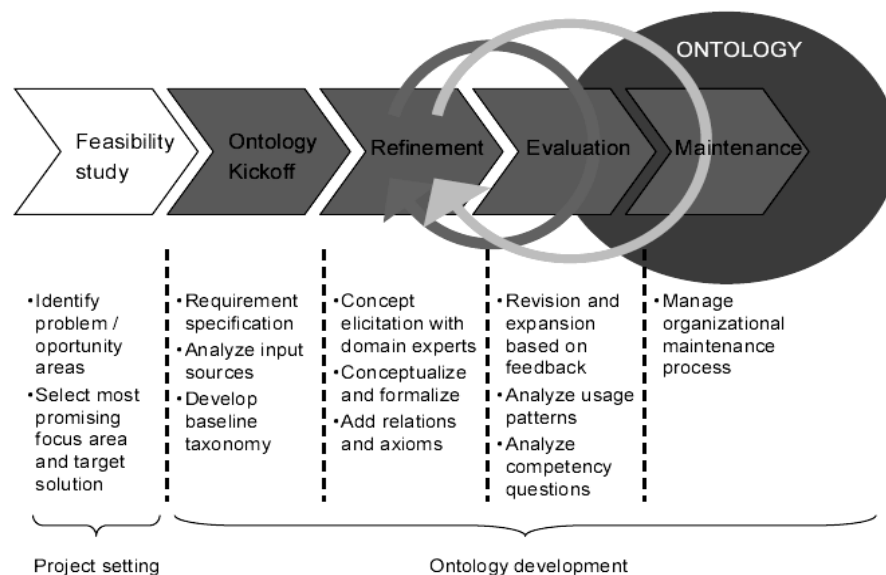


Figure 39. On-To-knowledge methodology [53]

METHONTOLOGY [16]:

Originating from ontology development process, the total steps of METHONTOLOGY are ontology management activities, development oriented activities, and ontology support activities. In METHONTOLOGY, the amount of effort for each activity has been determined as shown in Figure 40:

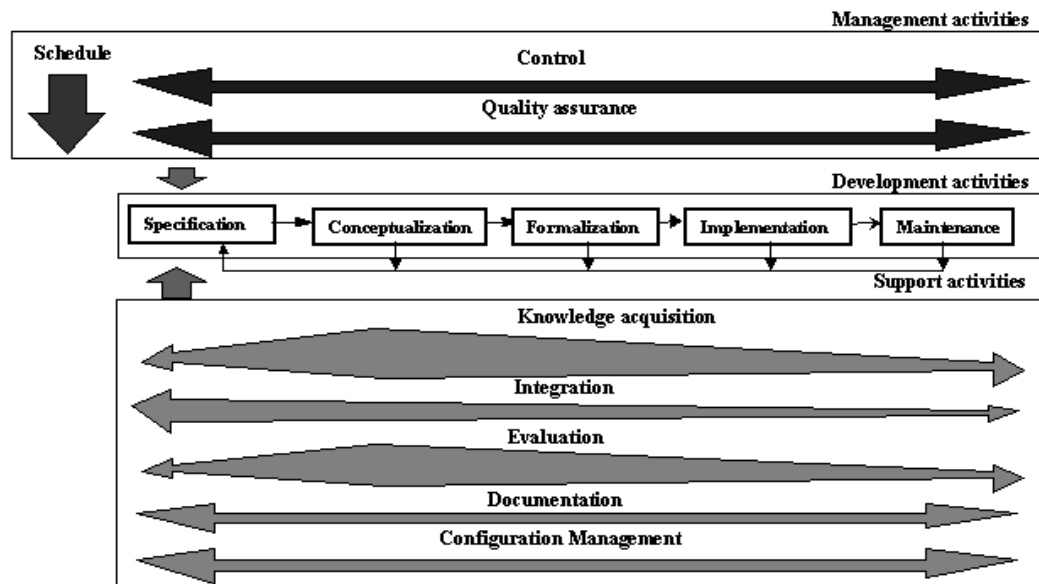


Figure 40. The METHONTOLOGY development process; Due to this figure, the amount of KA in conceptualization phase is higher rather than other phases [16]

Appendix 5: Collaborative ontology engineering methodologies

In this appendix, the evaluated collaboratively ontology engineering methodologies are described in brief:

Holsapple and Joshi [24]

As a Delphi-oriented⁴² structuring of collaboration, the ontology will be engineered through four steps:

1. **Preparation:** defining the design criteria, determining the boundary conditions, determining the evaluation standards.
2. **Anchoring:** Specifying the initial ontology which has some iteration of modifications on the initial ontology and some evaluations of ontology through the design criteria and standards of preparation phase.
3. **Iterative Improvement:** After sending the immature ontology to the collaborators, the leader of the ontology development will review the feedbacks (gathered formally by Delphi technique) of collaborators about the ontology components and applies to modify the next version of ontology. These tasks will be performed iteratively and the comments will be classified into different levels of importance.
4. **Application:** demonstrating the uses of ontology will be the last phase of this methodology.

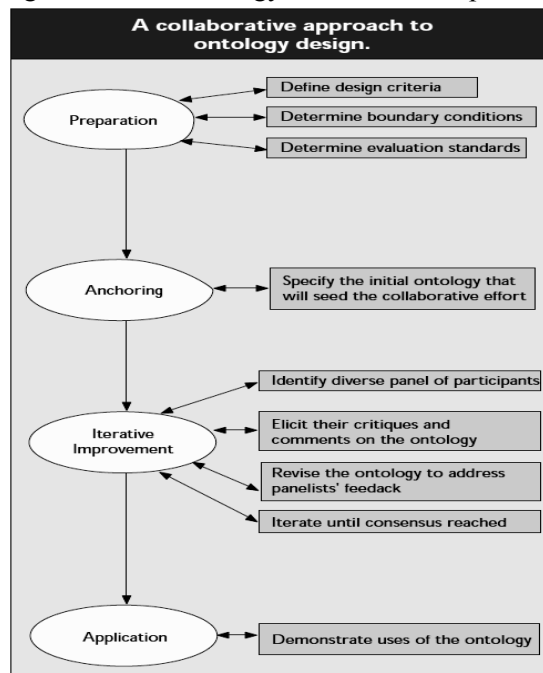


Figure 41. The proposed collaborative ontology engineering methodology by Holsapple and Joshi [24]

⁴² Collecting and integrating miscellaneous points of view about a domain formally.

In Delphi workflow, a mediator creates a panel of anonymous experts (as collaborators) to engage in making the comments. Via email, the mediator sends a questionnaire to the collaborators. After replying the questionnaire, the mediator investigates the feedbacks and comments. For the disagreements, the mediator prepares another questionnaire and repeats the second step.

Karapiperis and Apostolou [25]

This methodology has used Holsapple's methodology for collaborative ontology engineering but has focused more on the consensus building techniques. The Nominal Group Technique (NGT) has been adopted instead of Delphi technique for this methodology. The reasons for ignoring Delphi are: the results could be influenced by the coordinator's skill and influenced by the level of expertise and wording of questionnaire. NGT technique generates a large number of new ideas or prioritizes alternative solutions to let the collaborators make comments about the ontology components freely. The identified steps of NGT workflow are:

1. Presentation of the issue and giving alternatives to the collaborators without proposing any solution by the facilitator
2. Participation of collaborators to comment and prioritize the alternatives
3. Listing all the ideas of collaborators in a round robin (one after the other) format
4. Associating each idea to a number and writing them down on the flip chart to explain all of them for everybody to understand them carefully
5. Ranking the ideas by collaborators
6. Higher the total for each idea, higher the rank and each idea is designated accordingly
7. Ranking again the designated ideas to reach to the consensus

Also the phases of collaborative ontology engineering in this methodology are:

Phase1. Design criteria:

Clarity of every ontology component, Coherence and consistency among the ontology components, Extensibility of the ontology (in future you can change any ontology component without any conflict for the other components), and Minimal ontological commitment (the designer should consider only the related classes and properties and axioms of intended domain minimally to have extensibility possibility in future).

Phase2. Initial ontology development:

The development of initial ontology, which only the common phases of mentioned methodologies have been enumerated. This is comparable with Lemmens [5] as both of these methodologies are based on [54]: Domain of ontology, purpose of ontology, users, competency questions as a useful method to specify the ontology details, reusing the existed ontologies, enumeration of ontology terms, class definition and Class hierarchy, data type and the object properties of classes determination, determination of restrictions and constraints, and filling the ontology with the individuals.

Phase3. Iterative evaluation to evolve the ontology collaboratively:

In this phase, instead of validating the syntax of the language, the inconsistencies and redundancies of the ontology components are evaluated. To engage the collaborators in this phase, they have prepared an Evaluation Sheet (ES) which some actions should be done via this sheet (can be seen in Figure 42): Part A: to find the overlapping of classes through the common individuals, part B: adding the missed properties of current classes, part C: adding the missed relations of the current classes, part D: to broaden the vocabulary through the synonyms, and part E: to substitute a semantic relation when it is vague for the collaborator (not being a direct relation). In the USEFULNESS and AMBIGUITY columns some ranking methods have been applied to vote for the concepts, properties and relations (for this just USEFULNESS)

INITIAL ONTOLOGY EVALUATION SHEET (1st cycle)									
Page 1 / 9		USEFULNESS		AMBIGUITY					
		SYNONYMS		OVERLAPPING CONCEPTS					
		(complete all synonyms)		(complete all concepts that may have common individuals)					
CONCEPT :	EMPLOYEE	<div style="border: 2px solid black; padding: 5px; text-align: center;">D</div>		<div style="border: 2px solid black; padding: 5px; text-align: center;">A</div>					
PROPERTIES :	First Name Last Name Identification Number Sex Date of Birth Marital Status Home Address Telephone No Date of Hire					Name Surname ID, Employee No Gender Birthdate Address Phone no			
Complete missing properties :									
<div style="border: 2px solid black; padding: 5px; text-align: center;">B</div>						RELATION USEFULNESS Grade from 1 to 5		<div style="border: 2px solid black; padding: 5px; text-align: center;">E</div>	
RELATIONS (between concepts)						DIRECT RELATION (yes / no)		<div style="border: 2px solid black; padding: 5px; text-align: center;">C</div>	
Complete missing relations :									

Figure 42. Evaluation Sheet [25]

Phase4. Application

Described in previous methodology

DILIGENT [49, 55, 56]

DILIGENT is one of the most discussed distributed ontology engineering methodologies, which has been applied in many projects and researches (e.g. SEKT-project⁴³, SWAP). The considered roles in this methodology are: domain experts, end-users, knowledge engineers, and ontology engineers. To have any action in this methodology, every role should have proofs related to his/her action.

⁴³ <http://www.sekt-project.com>

The phases of DILIGENT are:

1. **Build:** all of the roles start to build the initial ontology, but only a few persons should engage in this process to reach to a small part of a shared ontology. This is the first action toward the consensus as the experts have argumentation to build the initial ontology.
2. **Local adaptation:** after building the initial ontology, the users adapt their needs to some parts of the shared ontology. They may retrieve some portions of knowledge from the shared ontology. They can locally change and interact with their local ontologies but the changes are possible through the requests from the control board of this methodology. The prerequisite to do these tasks: firstly the users should understand the shared ontology, secondly identify the similarities between the own and shared conceptualization (via mapping methods to map the correspondence concepts), thirdly identify the missed conceptualizations, fourthly change conceptualization and organize the local knowledge according to the conceptualization (ontology instantiation).
3. **Analysis:** the board should decide about the requests of users and investigate the similarities between users' ontologies. The decisions have to make balance among the users' requirements.
4. **Revision:** the board revises the shared ontology regularly to avoid the divergence of local ontologies from the shared ontology.
5. **Local Update:** the users update their local ontologies with respect to the new shared ontology version. By the way in this phase the users adapt their local ontologies to the new version like phase two.

To restrict the discussions and collaborations, the DILIGENT developers have proposed an ontology to formalize the argumentation of ontology engineering while spotting inconsistencies in arguments⁴⁴ is possible by a reasoner [55].

⁴⁴ For example, the arguer should not contradict his/her previous ideas, favors, and votes.

Appendix 6: The architecture of Collaborative Protégé

Remote Method Invocation (RMI) mechanism of Java in client-server Protégé facilitates the clients to access the ontologies stored on a central Protégé server and browse or change the ontology remotely. The users can be divided into desktop or web Protégé clients who can change and annotate the ontology components. Annotations are typed comments about the ontology comments or ontology changes. Corresponding to the Figure 43, the clients can access to the ontology repository in server where Changes and Annotations ontology (ChAO) knowledge bases exist. ChAO knowledge bases are the instances of ChAO ontology classes and these instances are about changes and annotations about the ontology:

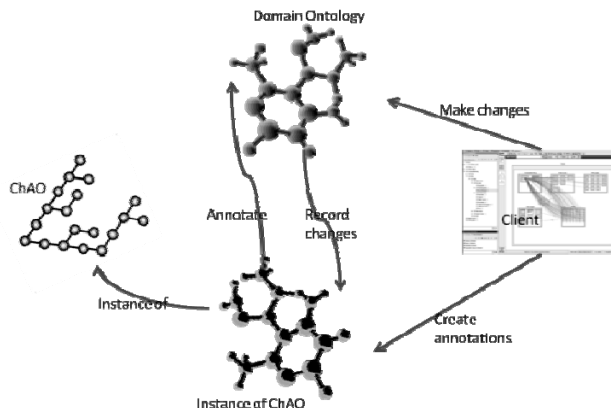


Figure 43. Different components of Collaborative Protégé and their relationships together

Four layered Java APIs (Changes API, Annotations API, Ontology Components API, and Ontology API) support collaborative ontology development in Protégé (in Protégé Server). The provided methods by these APIs are illustrated in Figure 44:

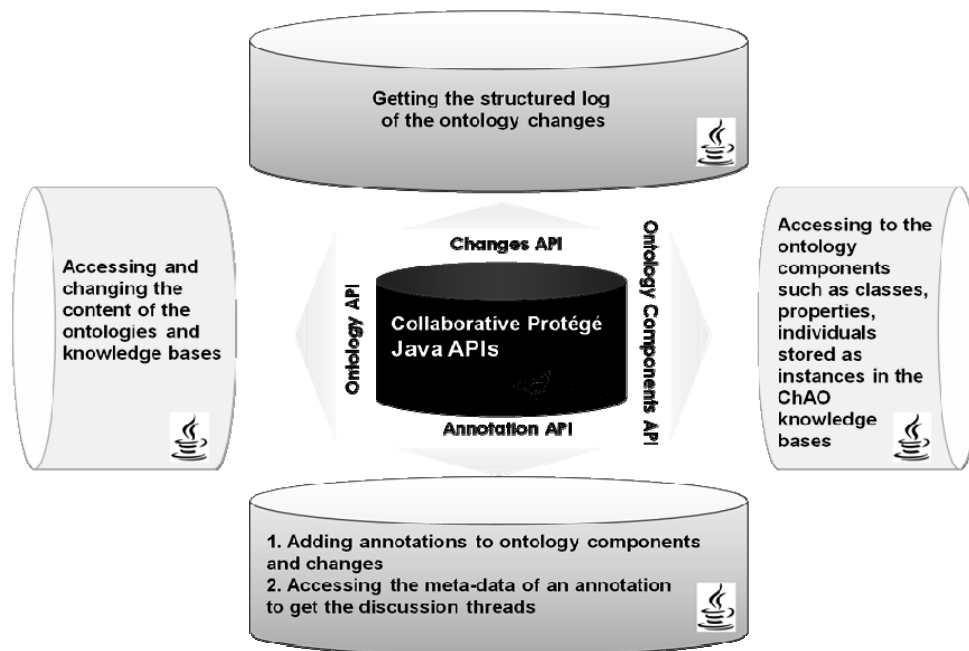


Figure 44. Provided methods by Java APIs in Protégé

Collaborative Protégé User Interface:

Due to Figure 16, the right panel is for collaboration which can be hidden when you don't need collaboration. In the collaboration tab, some tabs support collaboration in the process of ontology engineering.

The Entity notes Tab (or Annotation Tab): this tab shows the annotations (here the notes remind the annotations) attached to the selected class, property and individual. A small sign is used to show that there is an annotation about the class or property or individual (next to the ontology component). The lower part of the Entity notes Tab is devoted to represent the annotation detail. In fact the annotations are the instances of Annotation class. In Annotation ontology, the subclasses of Annotation class are some types of annotations which the annotation instances should be one of these types. It is very appropriate that the developers can add their own types to these types (Advice, Comment, Question, Example, Explanation, SeeAlso etc). As shown in Figure 16, the users can reply to an annotation or create a new discussion thread. If a user intends to see some specific annotations, she can filter the annotations via filtering the author, the date, the annotation text, and annotation type. One of the useful facilities in this tab is the usage of internal links (the links referring to one of the ontology components such as a class, property, and instance).

The Changes tab: this tab is used to represent a chronological and continuously list of all the changes related to the selected entity. In this tab the users can also reply to a change and make comments (annotating) about them.

All notes Tab: corresponding to the domain of discourse, all of the comments and notes created about the entities are represented in this tab. This is very good to get general idea about the discussions and comments of ontology components.

Chat Tab: all of the users connected to the Protégé server can have communication together via chat environment. The other users connected to the server can observe the chats and the people who are on-line.

Ontology notes Tab: if the user intends to annotate and attach a comment about the ontology itself rather than its components, she can perform the mentioned operations in the Entity notes tab. This tab can be used to describe the whole policies of ontology creation and intention such as design guidelines for the ontology.

Search Tab: this tab (Figure 45) facilitates the process of searching among the annotations for the user. Search about the authors, annotation text, annotation type, and date of annotations via usage of logical operators (AND, OR) can be done by the user. Also in other tabs, the user can search about related contents.

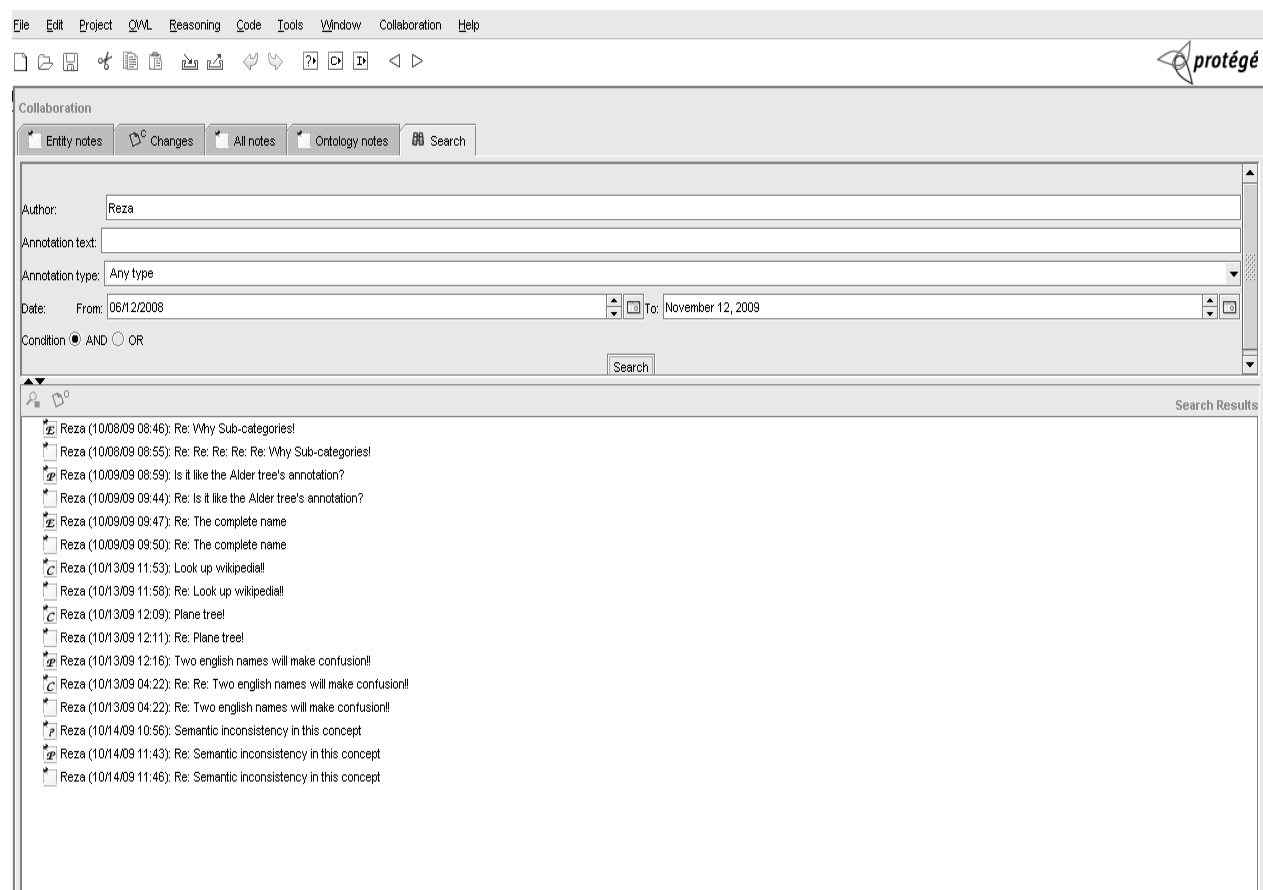


Figure 45. Search Tab in Collaborative Protégé

Bibliography

1. Shekhar, s. and S. chawla, *Spatial Databases: A Tour* 2003, New Jersey: Prentice Hall
2. Zhong-Ren Peng , M.-H.T., *Internet GIS: Distributed Geographic Information Services for the Internet and Wireless Networks* 2003: John Wiley and Sons.
3. IEEE-Computer-Society-Standards-Coordinating-Committee, *IEEE standard computer dictionary : a compilation of IEEE standard computer glossaries*, 610. 1990, New York, NY, USA: Institute of Electrical and Electronics Engineers.
4. Stuckenschmidt, H. and F. Van-Harmelen, *Information Sharing on the Semantic Web*. 2004: SpringerVerlag.
5. Lemmens, R., *Semantic Interoperability of Distributed Geo-Services*. 2006, TU Delft: Delft, The Netherlands.
6. Lund, H.G., *Definitions of Forest, Deforestation, Afforestation, and Reforestation*. [Online] Gainesville, VA: Forest Information Services, 2009. Available from the World Wide Web: <http://home.comcast.net/~gyde/DEFpaper.htm>.
7. Comber, A., P. Fisher, and R. Wadsworth, *What is land cover?* Environment and Planning B: Planning and Design, 2005. 32: p. 199 - 209.
8. Lutz, M., et al., *Overcoming semantic heterogeneity in spatial data infrastructures*. Computers & Geosciences, 2009. 35(4): p. 739-752.
9. Berners-Lee, T., J. Hendler, and O. Lassila, *The semantic Web*. Scientific American, 2001.
10. *OED- Oxford English Dictionary* Oxford-University-Press, Editor. accessed on 22 June 2009 by ITC Digital library.
11. Gruber, T.R., *A translation approach to portable ontology specifications*. Knowledge Acquisition, 1993. 5(2): p. 199-220.
12. Schade, S., et al., *Ontologies in the SWING project: Report on Modelling Approach and Guideline*. Deliverable 3.2, 2008.
13. Euzenat, J. and P. Shvaiko, *Ontology Matching*. 2007: Springer.
14. Gomez-Perez, A., M. Fernandez-Lopez, and O. Corcho, *Ontological Engineering: with examples from the areas of Knowledge Management, e-Commerce and the Semantic Web. (Advanced Information and Knowledge Processing)*. 2007: Springer-Verlag New York, Inc.
15. Lassila, O. and D.L. McGuinness, *The Role of Frame-Based Representation on the Semantic Web*, in *Technical Report KSL-01-02*. . 2001, Knowledge Systems Laboratory: Stanford University. Stanford, California.
16. Gomez-Perez, A., M. Fernandez-Lopez, and O. Corcho, *Ontological Engineering: with examples from the areas of Knowledge Management, e-Commerce and the Semantic Web. (Advanced Information and Knowledge Processing)*. 2004: Springer-Verlag New York, Inc.
17. Fernandez-Lopez, M., A. Gomez-Perez, and N. Juristo. *METHONTOLOGY: From Ontological Art Towards Ontological Engineering*. in *Spring Symposium on Ontological Engineering of AAAI*. 1997.
18. Gruninger, M. and M. Fox. *Methodology for the Design and Evaluation of Ontologies*. in *IJCAI'95, Workshop on Basic Ontological Issues in Knowledge Sharing, April 13, 1995*. 1995.
19. Reitsma, F., et al., *Semantics, ontologies and eScience for the geosciences*. Computers & Geosciences, 2009. 35(4): p. 706-709.

20. McGuinness, D. and F. van Harmelen, <http://www.w3.org/TR/owl-features/> Owl web ontology language overview. Proposed recommendation, W3C., 2003.
21. Van-Harmelen, F. and G. Antoniou, *A Semantic Web Primer*. 2004: MIT Press.
22. Fernández-López, M., et al., *Building a Chemical Ontology Using Methontology and the Ontology Design Environment*. IEEE Intelligent Systems, 1999. 14(1): p. 37-46.
23. Janowicz, K., et al. *Similarity as a Quality Indicator in Ontology Engineering*. in *5th International Conference on Formal Ontology in Information Systems*. 2008.
24. Holsapple, C.W. and K.D. Joshi, *A collaborative approach to ontology design*. Communications of the ACM, 2002. 45(2): p. 42-47.
25. Karapiperis, S. and D. Apostolou, *Consensus Building in Collaborative Ontology Engineering Processes*. Journal of Universal Knowledge Management, 2006. 1, no.3: p. 199-216.
26. Noy, N.F., A. Chugh, and H. Alani, *The CKC Challenge: Exploring Tools for Collaborative Knowledge Construction*. IEEE Intelligent Systems, 2007. 23(1): p. 64-68.
27. Noy, N.F., et al. *A framework for ontology evolution in collaborative environments*. in *5th Int. Semantic Web Conf., ISWC*. 2006. Athens.
28. Schaffert, S., R. Westenthaler, and A. Gruber. *IkeWiki: A UserFriendly Semantic Wiki*. in *3rd European Semantic Web Conference (ESWC 2006)*. 2006. Budva, Montenegro.
29. Siorpaes, K. and M. Hepp. *myOntology: The Marriage of Ontology Engineering and Collective Intelligence*. in *Bridging the Gap between Semantic Web and Web 2.0 (SemNet 2007)* 2007.
30. Tudorache, T., et al., *Supporting Collaborative Ontology Development in Protégé*, in *The Semantic Web - ISWC 2008*. 2008. p. 17-32.
31. W3C. *Annotea Project*. 2003 , Accessed at 20 May 2009; <http://www.w3.org/2001/Annotea/#overview>
32. Avery, M., et al., *Building united judgment: A handbook for consensus decision making*. 1981, Fellowship for Intentional
33. Ellis, D.G. and B.A. Fisher, *Small Group Decision Making: Communication and the Group Process* 1993: McGraw-Hill Humanities.
34. Kahan, J. and M.-R. Koivunen, *Annotea: an open RDF infrastructure for shared Web annotations*, in *Proceedings of the 10th international conference on World Wide Web*. 2001, ACM: Hong Kong, Hong Kong.
35. Campanini, S.E. and R.T. Paolo Castagna. *Platypus Wiki: a Semantic Wiki Wiki Web* in *1st Italian Semantic Web Workshop Semantic Web Applications and Perspectives (SWAP)* 2004. Ancona, Italy
36. Dello, K., Elena, and R. Tolksdorf. *Creating and using Semantic Web information with Makna*. in *SemWiki*. 2006: CEUR-WS.org.
37. Krötzsch, M., S. Schaffert, and D. Vrandecic, *Reasoning in Semantic Wikis*, in *Reasoning Web*. 2007. p. 310-329.
38. Auer, S., S. Dietzold, and T. Riechert. *OntoWiki - A Tool for Social, Semantic Collaboration*. in *The 5th International Semantic Web Conference (ISWC 2006)*. 2006: Springer.
39. Hepp, M., D. Bachlechner, and K. Siorpaes, *OntoWiki: community-driven ontology engineering and ontology usage based on Wikis*, in *Proceedings of the 2006 international symposium on Wikis*. 2006, ACM: Odense, Denmark.
40. Kuhn, T. *AceWiki: Collaborative Ontology Management in Controlled Natural Language*. in *SemWiki*. 2008: CEUR-WS.org.
41. Domingue, J. *Tadzebao And Webonto: Discussing, Browsing, Editing Ontologies On The Web*. in *11th Knowledge Acquisition for Knowledge-Based Systems Workshop*. 1998.
42. Sure, Y., et al., *OntoEdit: Collaborative Ontology Development for the Semantic Web*. The Semantic Web - ISWC 2002: First International Semantic Web Conference, Sardinia, Italy, June 9-12, 2002. Proceedings. 2002. 221.
43. Xexéo, G., et al., *COE: A collaborative ontology editor based on a peer-to-peer framework*. Advanced Engineering Informatics, 2005. 19(2): p. 113-121.

44. Farquhar, A., R. Fikes, and J. Rice, *The Ontolingua Server: a tool for collaborative ontology construction*. International Journal of Human-Computer Studies, 1997. 46(6).
45. Malzahn, N., et al. *Collaborative Ontology Development - Distributed Architecture and Visualization*. in *German e-Science Conference 2007*. 2007. Baden-Baden , Germany: CC-BY-NC-ND Creative Commons Attribution.
46. Arpirez, J.C., et al., *WebODE: a scalable workbench for ontological engineering*, in *Proceedings of the 1st international conference on Knowledge capture*. 2001, ACM: Victoria, British Columbia, Canada.
47. Tudorache, T. and N. Noy. *Collaborative Protege*. in *16th International World Wide Web Conference (WWW2007)*. 2007. Banff, Alberta, Canada.
48. Sirin, B. and E. Sirin, *Pellet: An owl dl reasoner*. International Workshop on Description Logics, 2004.
49. Pinto, H., et al., *Distributed Engineering of Ontologies (DILIGENT)*, in *Semantic Web and Peer-to-Peer*. 2006. p. 303-322.
50. Guarino, N. and C.A. Welty, *An Overview of OntoClean*, in *Handbook on Ontologies*. 2004, Springer. p. 151-172.
51. Klien, E., et al., *Ontologies in the SWING Application: Requirement Specification (Deliverable 3.1)*. 2007.
52. Uschold, M. and M. King, *Towards a methodology for building ontologies*, in *Workshop on Basic Ontological Issues in Knowledge Sharing*. 1995: IJCAI-95, Montreal, Canada.
53. Staab, S., et al., *Knowledge Processes and Ontologies*. IEEE Intelligent Systems, 2001. 16(1): p. 26-34.
54. Noy, N. and D.L. McGuinness, *Ontology Development 101: A Guide to Creating Your First Ontology*, in *Tech. Rep. KSL-01-05, Knowledge Systems Laboratory*. 2001.
55. Tempich, C., et al., *An Argumentation Ontology for DIstributed, Loosely-controlled and evolvinG Engineering processes of oNTologies (DILIGENT)*, in *The Semantic Web: Research and Applications*. 2005. p. 241-256.
56. Vrandečić, D., et al., *The DILIGENT knowledge processes*. Journal of Knowledge Management, 2005. 9(5).