UNIVERSITÄT KOBLENZ · LANDAU
Fachbereich 4: Informatik

Institut für Wirtschafts- und Verwaltungsinformatik

Methods to visualize ontology

Bachelor Thesis

To obtain the degree Bachelor of Science in Information Management

Submitted by

Vivek Srivastava

Matriculation number: 202120817

First supervisor: Prof. Dr. Maria A. Wimmer
Research Group eGovernment

Second supervisor: Christian Schneider
Research Group eGovernment

Koblenz, 14th July 2011

Erklärung

Ich versichere, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe und dass die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat und von dieser als Teil einer Prüfungsleistung angenommen wurde. Alle Ausführungen, die wörtlich oder sinngemäß übernommen wurden, sind als solche gekennzeichnet.

Die Richtlinien der Forschungsgruppe für Qualifikationsarbeiten habe ich gelesen und anerkannt, insbesondere die Regelung des Nutzungsrechts.

Mit der Einstellung dieser Arbeit in die Bibliothek bin ich *Ja* [x] *Nein* [ ] *einverstanden*

Der Veröffentlichung dieser Arbeit im Internet stimme ich zu. *Ja* [x] *Nein* [ ]

Koblenz, den 14.07.2011 *Vivek Srivastavat*

Dedicated to My

Master

H.H. Paramsant Dr. Chaturbhuj Sahaiji

Zusammenfassung

Wir leben in einer Welt, in der sich die Menge von Informationen nicht bloß vergrößert, sondern geradezu explodiert. Mehr und mehr Informationen sind verfügbar und nur einen Klick entfernt. Auf der einen Seite ergeben sich aus dieser Informationsexplosion neue Herausforderungen, die dann auf der anderen Seite die Forscher motivieren diese Herausforderungen anzunehmen und neue Lösungen zu finden. Eine Herausforderung die sich aus dieser Explosion ergibt ist das Verwalten, Organisieren und Strukturieren von Informationen. Um die Informationsverwaltung und -organisation zu optimieren, sollten Informationen durch einheitliche Begriffe charakterisiert werden. Das heißt, während dem Lesen, Hören oder Verweisen auf eine bestimmte Information sollte jeder dieselbe Bedeutung dieser Information verwenden und verstehen. Das gemeinsame Verständnis wird durch den Einsatz von Ontologien erreicht.

Zum einheitlichen Verständnis und zur einheitlichen Charakterisierung von Ontologien werden diese in maschinenlesbaren Sprachen wie RDF, OWL, usw. geschrieben. Wenn sie groß und komplex werden, ist es jedoch schwierig sie zu verstehen und zu analysieren. Um das Verständnis zu erleichtern wurden verschiedene Ansätze entwickelt Ontologien zu visualisieren. Da die Entwickler, Besitzer und Benutzer von Ontologien über die ganze Welt verteilt sind, ist eine Visualisierung von Ontologien über das World Wide Web (WWW) erstrebenswert. Während es bereits einige Evaluierungen der aktuell verfügbaren Visualisierungstools für Ontologien gibt, so wurde die Visualisierung von Ontologien im WWW bisher selten betrachtet. Bei einer Visualisierung im WWW spielen neben Anzeige und Navigation auch nicht-funktionale Aspekte wie Skalierbarkeit, Reaktionszeiten und Ladezeiten eine sehr wichtige Rolle. Weitere wichtige Bewertungskriterien sind die Erweiterbarkeit bzgl. zukünftiger Bedürfnisse und die Möglichkeit der individuellen Anpassung an bestimmte Anforderungen.

In dieser Arbeit werden vier der derzeit verfügbaren webbasierten Visualisierungstools evaluiert (FlexViz, Jambalaya applet, Experimental jOWL TouchGraph, Plone ontology). Im Rahmen dieser Arbeit werden weiterhin folgende Fragen erforscht:

1. Welche Anforderungen an die Visualisierung von Ontologien können definiert werden?

2. Wie kann eine Bewertung und Analyse durchgeführt werden?

3. Wie kann eine ausgewählte Methode getestet werden?

Basierend auf den Ergebnissen dieser Fragestellungen stellte sich die Visualisierungsmethode FlexViz im gegebenen Szenario als beste heraus. FlexViz wurde eingesetzt, indem es in das Content Management System Plone integriert wurde. Es wurde mit VCD Ontologien getestet und schließlich aus der Sicht verschiedener Interessengruppen analysiert. Es zeigte sich, dass FlexViz ein gutes Tool zum Visualisieren, Verstehen und Analysieren von bereits entwickelten Ontologien ist, da die Benutzeroberfläche benutzerfreundlich und interaktiv ist. Jedoch zeigt FlexViz deutliche Schwächen bzgl. dem Bearbeiten, Erweitern und Entwickeln von Ontologien.

Abstract

We are living in a world in which information is not just expanding, but exploding. More and more information is available with just a click. On one hand, this explosion of information raises new challenges, and on the other hand, it motivate scientists to move forward and discover new solutions to meet these challenges. One of the challenges which has resulted from the explosion of information is the management, organization and structuring of information. To optimize information management and organization, information should be characterized by a uniformity of terms. This means that while reading, hearing or referring to some piece of information, all recipients should use and understand the same meaning of the information. This need for common understanding of concepts is satisfied by the use of ontology.

To give uniformity of understanding and characterization, ontologies are written in languages like RDF, OWL and others which are machine readable. It becomes more challenging to understand and analyze them when they are very large and complex. To ease the understanding of ontologies, various visualization approaches have been developed. Ontology developers, owners and users are spread across the world, which creates new challenges regarding the visualization of ontologies over the World Wide Web (WWW). Even though there are a few evaluations available for currently available ontology visualization tools, not much work has been done with a focus on ontology visualization over the Web. For visualization over the web, apart from display and navigation functionalities, non-functional issues, such as scalability, reaction time and loading time, play much significant roles. Other important aspects are the possibility of extension to meet future needs and the customization of the visualization tool for specific domain requirements.

In this study, the author evaluates four visualization methods (FlexViz, the Jambalaya applet, Experimental jOWL TouchGraph, and Plone ontology) from the pool of currently available web based visualization methods. Further, this study attempts to answer of the following research questions:

1. What are the requirements for ontology visualization?

2. How is comparative analysis and evaluation carried out?

3. How can a chosen method be tested?

Based on the results of the evaluation, the visualization method FlexViz was found to be the most suitable for the given scenario. The deployment of FlexViz was carried out and integrated within Plone CMS and logical parts of the VCD ontology are tested. Finally, FlexViz was analyzed from different stakeholder's perspectives. Results showed that FleViz is a reliable tool for visualizing, understanding and analyzing a developed ontology because of its user-friendly and interactive interface, but performs poorly in regards to the editing, extension and development process

Acknowledgement

Man is a social animal – but life makes man learn the lessons of rising from dependence to independence and from independence to interdependence. The odyssey of life becomes easier when we move together. This thesis is a synergistic product of many minds. I am grateful for this opportunity to express my gratitude towards them all.

Foremost, I feel an exceptional deep sense of gratitude for my late father who nurtured my vision and taught me the good things that really matters in life. His happy memories will always remain fresh in my mind and his values and principles shall provide a constant inspiration throughout my life. His complex-free, simple, yet so powerful and exemplary life shall stay as a beacon not only as a guiding light, but as a driving force in all my work. His presence like fragrance is still everywhere around me. The need for that soft, gentle and divine embrace will perpetually remain throughout my life.

I owe a debt of gratitude to Prof. Dr. Maria Wimmer for allowing me to carry out this thesis work in Research Group eGovernment, University of Koblenz-Landau. I greatly appreciate her wisdom, objectivity, and generous assistance which are so uncommon. She was totally committed and with her considerable conceptual talent and insights contributed a lot in bring out this work. It is her effort which brought the best out of me, and for her ceaseless inspiration and unending support, without which I would not have ventured in this field.

I am grateful to my second supervisor Mr. Christian Schneider Research Group eGovernment, University of Koblenz-Landau who is calm and always ready to help with a smiling face. Also, my gratitude goes to him, for giving me immense support, guidance and assistance. I am also grateful to him for sincerely inquiring about the progress of my work from time to time.

I would like to acknowledge the continuing support of my brothers Sh. Sanjiv Kumar Ji and Amit Utkarsh. I thank them for their tender guidance, tough comments, accompanied with their gentle support like elder brothers.

The chain of gratitude would definitely be incomplete without my dear family members and lovely wife Manisha which provides an additional blissful dimension to my life.

Table of Contents

ZUSAMMENFASSUNG ..... **IV**
ABSTRACT ..... **V**
ACKNOWLEDGEMENT ..... **VI**
LIST OF FIGURES ..... **IX**
LIST OF TABLES ..... **X**
LIST OF ACRONYMS ..... **XI**
1 INTRODUCTION **1**
1.1 Research design 1
1.2 Structure of thesis 2
2 BASICS OF ONTOLOGY **3**
2.1 Ontology from philosophy to computer science 3
2.2 Importance of ontology 5
2.3 Types of ontologies 6
2.3.1 Top-level ontology 7
2.3.2 Domain ontology 7
2.3.3 Task ontology 7
2.3.4 Application ontology 8
2.4 Ontology representation languages 8
2.4.1 XML and XML Schema 10
2.4.2 RDF and RDF Schema 10
2.4.3 Web Ontology Language (OWL) 11
2.5 Conclusion 12
3 ONTOLOGY VISUALIZATION **13**
3.1 Significance of ontology visualization 13
3.2 Characteristics of good visualization tool 13
3.3 Analysis of existing ontology visualization methods 14
3.4 Web-based ontology visualization 15
3.5 Conclusion 16
4 EPROCUREMENT AND VIRTUAL COMPANY DOSSIER ONTOLOGY **17**
4.1 Virtual Company dossier (VCD) 17
4.2 VCD ontology 18
4.3 Stakeholders in VCD System 23
4.4 Conclusion 23
5 EVALUATION DESIGN **24**
5.1 Evaluation Criterion 25
5.2 Evaluation matrix 29
5.3 Conclusion 29

6 EVALUATION OF ONTOLOGY VISUALIZATION TOOLS 30
6.1 Selection of visualization tools 30
6.1.1 FlexViz 31
6.1.2 Jambalaya 32
6.1.3 jOWL TouchGraph visualization (Experimental) 34
6.1.4 Plone ontology 35
6.2 Evaluation of visualization methods 36
6.3 Testing 40
6.4 Conclusion 45
7 REVIEW, CRITICAL ANALYSIS & RECOMMENDATIONS 46
8 CONCLUSION 48
9 BIBLIOGRAPHY 50

List of Figures

Figure 1: Research steps 2

Figure 2 : Vehicle and other classes and relationship between them [7] 4

Figure 3: Types of ontologies [9] 7

Figure 4: History of ontology related technologies [11] 8

Figure 5: W3C Semantic Web Activity "layer cake" diagram [11] 9

Figure 6: Working of VCD system showing stakeholders and eProcurement process [30] 18

Figure 7: Logical parts of VCD ontology [33] 18

Figure 8: Common-schema [34] 19

Figure 9: TendererSchema [34] 20

Figure 10: Tenderer-criterion-schema [34] 21

Figure 11: Collector-schema [34] 21

Figure 12: VCD ontology [34] 22

Figure 13: Criterion classification 25

Figure 14: FlexViz demo 32

Figure 15: Jambalaya plug-in applet demo 33

Figure 16: jOWL TouchGraph visualization (Experimental) demo 35

Figure 17: Plone ontology demo 36

Figure 18: Multiple inheritance in FlexViz 37

Figure 19: Testing the implementation 41

Figure 20: Adobe Flash Builder 4.5 overview 42

Figure 21: Graph editing for ontology mapping 43

Figure 22: Plone Site local test 44

Figure 23: Uploading SWF files in Plone 44

Figure 24: Visualization of CollectorSchema ontology locally in Plone 45

List of Tables

Table 1: Evaluation matrix 29

Table 2: Visualization tools grouping based on technology used 30

Table 3: Evaluation results 40

List of Acronyms

2D.....2 Dimensional
AI.....Artificial Intelligence
CA.....Contracting Authority
CMS.....Content Management System
DAML.....DARPA Agent Markup Language
DARPA..... Defense Advanced Research Projects Agency
DOM.....Document Object Model
DTD.....Document Type Definition
EDI.....Electronic Data Interchange
EIF.....European Interoperability Framework
EO.....Economic Operator
EU.....European Union
EVS.....European VCD System
HTML.....Hypertext Markup Language
ICT.....Information and Communication Technology
MXML.....Magic eXtensible Markup Language
NVS.....National VCD System
OIL.....Ontology Interchange Language
OMS.....Ontology Management System
OWL.....Web Ontology Language
OWL DL.....Web Ontology Language Description Logic
PEPPOL.....Pan-European Public Procurement Online
RDF.....Resource Description Framework
RDFS.....RDF Schema
RIF.....Rule Interchange Format
SDK.....Software Development Kit
SGML.....Standardized Generalized Markup Language
SHriMP.....Simple Hierarchical Multi-Perspective
SP.....Solution Provider
SVG.....Scalable Vector Graphic
SWF.....Shock Wave Flash (file format)
UNSPSC.....Universal Standard Products and Services Classification Code

UI.....User Interface
URI…….....……Uniform Resource Identifier
URL….....Uniform Resource Locator
VCD…….....……Virtual Company Dossier
VRML…….....…….Virtual Reality Modeling Language
W3C…….....……World Wide Web Consortium
XML…….....……eXtensible Markup Language
ZUI.....Zoomable User Interface

1 Introduction

In today's society, a huge amount of information is shared by many participants. This information must be characterized by a uniformity of terms. In a similar context, everyone should understand or and the same meaning of a word when they are reading, hearing or referring it. Among different computer science disciplines, this need for a common understating of concepts is satisfied by the creation of an ontology, which in turn led to the creation of the semantic web and personalized information management. [1]

Web Ontology Language (OWL) has been defined by the World Wide Web consortium (W3C). OWL is the proposed standard for web ontology. Practically, OWL is created for publishing, sharing and providing mechanisms for creating all the components of ontologies, which are concepts, instances, relations and axioms. It describes the semantics of knowledge in a machine-accessible and machine-readable way. At first glance, OWL ontology may seem as complex. So in order to enhance the understanding of ontologies, several visualization approaches have been developed, some of which are domain-independent and graph-based, while others are not. [2]

The World Wide Web (WWW) has provided a common platform for researchers, developers and users; they collaborate, share and develop applications together while in different parts of the world. The requirements and characteristics of visualization tools also change as they over the Web. Any visualization tool should display its contents and offer ways to navigate, but loading time, reaction time and scalability play a more vital role when the tool is web-based. Therefore, web-based ontology visualization tools must be analyzed and evaluated with a few additional criteria. Even though some evaluations are done for ontology visualization tools in general, not much work has been done with a focus on web-based visualization tools.

For any piece of software, there are different interest groups or stakeholders. The different stakeholder groups have different requirements and needs. The developer is interested in editing and testing, while the user looks for the interactivity as well as the user-friendliness of the software. Available evaluations of visualization tools lag in addressing the usefulness of the tools for various stakeholders.

1.1 Research design

The overall objective of this thesis is to analyze and evaluate four visualization methods and run a test case of logical parts of the Virtual Company Dossier (VCD) ontology locally for one of the highest scored visualization methods in the evaluation.

This thesis will attempt to answer the following research questions:

1. What will be the requirements for a special domain of ontology visualization?
2. How is the comparative analysis and evaluation carried out?
3. How can a chosen method be tested?

To answer the above questions, this study was carried out by initiating a literature review, analyzing the requirements of a special domain, evaluating four visualization methods, choosing the best-rated tool, deploying it with available logical parts of the VCD ontologies and embedding it in Plone CMS(Figure 1).

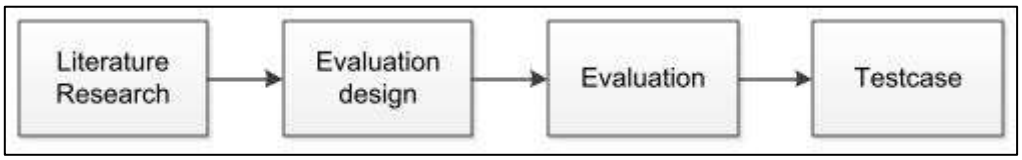

Figure 1: Research steps

1.2 Structure of thesis

The contents of this thesis are divided into three major groups.

Group 1: Theoretical Background

Group 2: Practical Work

Group 3: Results

Group 1 (Chapters 2-4) will describe the basic theoretical foundation needed to carry out this work by covering the topics relevant to this thesis. Chapter 2 will present previous work done on ontology, its importance, types, and representation languages. Chapter 3 will explain the importance of the visualization of ontologies. It also gives insight regarding the special requirements and challenges of specific domains. Further Chapter 4 describes the test case, the VCD ontology, and discusses various stakeholders associated with it.

In Group 2 (Chapters 5-6), we will describe the design of a matrix, which will form the basis of the evaluation of different ontology visualization packages. Specifically, we will discuss the criteria, how they are grouped and how the evaluation matrix is prepared. Chapter 6 will present four web-based visualization methods from a pool of available web-based visualized methods and discuss the rationale for short listing them. Thereafter, these methods will be evaluated and highest-scoring method will be chosen for deployment. We will also explain how the Plone Content Management System (CMS) is locally installed to embed the highest-scoring visualization method on the logical parts of the VCD ontology.

Group 3 (Chapters 7-8) will describe the outcome of the testing and analysis of the results. Recommendations regarding future work will also be discussed.

2 Basics of ontology

Ontology is a branch of philosophy that deals with nature and the organization of reality. Although in computer science the term “ontology” is derived from philosophy, it is used with a different meaning. [3] In this chapter, we will look closely at the concept of ontology. Section 2.1 will show how the term ontology has been adopted in the computer science discipline and what the basic building blocks of ontology are. We will further discuss the importance of ontology in Section 2.2. Section 2.3 will present the classification of ontology, and finally, in Section 2.4, ontology representation languages are discussed.

2.1 Ontology from philosophy to computer science

Since the beginning of human history it has been a profound endeavor for mankind to define “being” or to define what is meant by “essence” Aristotle, the father of metaphysics said that everything has got something (property) that makes it “a thing”. Metaphysics attempted to clarify the fundamental notions by which people understand the world, its existence, definitions of objects, property, space, time, and possibility. A branch of metaphysics which investigates the basic categories of being and how they relate to each other is ontology. [4]

In computer science, the term “ontology” is being adopted by the in Artificial intelligence, Software Engineering and Database communities. As all these areas of computer science deal with knowledge representation, they borrowed the term ”ontology” from philosophy. [5] Artificial intelligence aims to make intelligent machines that can make autonomous decisions which require the representation of knowledge. In object oriented programming objects are created to perform operations. This paradigm classifies the whole world into objects that represent a metaphor of the real world. Those elements have some basic characteristics (properties) and methods (functions). In designing a database, high-level conceptual models are needed to give an abstract representation of a domain of problems without considering the issues of implementation. In each of the above domains, the representation of knowledge is addressed in a different manner because each one is interested in a different problem, but all the three areas require the representation of knowledge or concepts. [5]

A comprehensive definition and explanation of ontology is given by Studer:

“*An ontology is a formal, explicit specification of a shared conceptualization*” [6] [7]

Conceptualization refers to an abstract model of some phenomenon in the world by having identified the relevant concepts of that phenomenon. Explicit means that the type of concepts used, and the constraints on their use, are explicitly defined. Formal refers to the fact that the ontology should be machine-readable. Share reflects the notion that an ontology captures consensual knowledge. That is, it is not the private knowledge of some individual, but accepted by a group. [6]

World Wide Web Consortium (W3C) gives another definition of ontology:
“An ontology defines the terms used to describe and represent an area of knowledge.

Ontologies are used by people, databases, and applications that need to share domain information. Ontologies include computer-usable definitions of basic concepts in the domain and the relationships among them. They encode knowledge in a domain and also knowledge that spans domains. In this way, they make that knowledge reusable. Ontologies are usually expressed in a logic-based language, so that detailed, accurate, consistent, sound, and meaningful distinctions can be made among the classes, properties, and relations" [6]

In summary, ontologies describe the basic concepts of a domain and also define the relationships among them. Regardless of the language in which ontologies are expressed, they share some structural similarities. These common structural similarities are the building blocks of ontology. Most of the ontologies contain classes, attributes, relationships and instances. We will explain each of these in detail [8]:

Classes: Classes are concepts that are referred to as type, sort, category or kind. They are abstract groups, sets or collections of objects. They can contain a collection of classes, objects or both.

Figure 2 shows an example of the vehicles class along with other classes and relationships between them. (Figure 2):

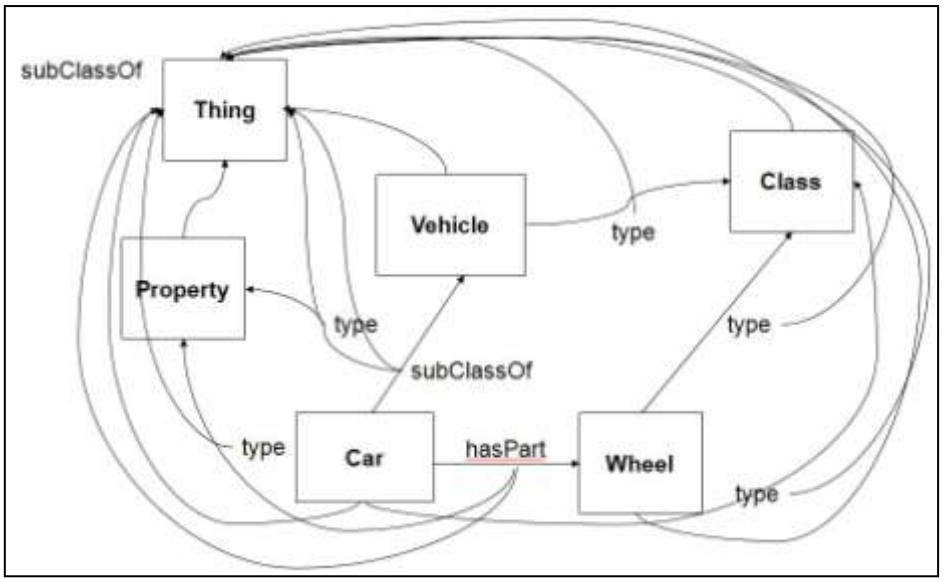

Figure 2 : Vehicle and other classes and relationship between them [9]

Attributes: In ontology, the properties of the object are attributes. Each attribute has a name and a value and is used to store values that are specific to the respective object.

For example, Ford Explorer object has attributes such as:

- Name: Ford Explorer
- Number of doors: 4

Relationship: How objects in ontology are related to other objects is specified by the relationships or relations. A relation is of a type or class that specifies in what sense the object is related to the other objects in the ontology.

For example, the ontology that contains the concepts Ford Explorer and Ford Bronco can have the following attribute:

- successorOf: Ford Explorer.

This relationship shows that the Ford Explorer is replaced by the Ford Bronco [8]

The most important type of relation is the ***is-a*** *relation, which* explains the hierarchical taxonomy.

For example, a Ford Explorer is-a 4-wheel drive vehicle, which **is-a** Car. [8]

Another common type of relation is the meronymy relation, written as ***part-of***. It represents how objects combine together to form composite objects. [8]

For example, since a steering wheel is a component of a car and a Ford Explorer is-a Car. It can be said that “steering wheel is-part-of Ford Explorer”. [8]

Instances: The ground level components of ontology are instances. They may include concrete as well abstract objects. Ontology might or might not contain instances but it provides a means of classifying instances or objects. [8]

In the next section we will see why ontology is important.

2.2 Importance of ontology

To share common information in a domain, ontology provides a common vocabulary for researchers. It provides machine readable definitions of basic concepts in the domain and the relations among them. Its importance can be highlighted as follows:

- Ontology shares a common understanding of the structure of information among people and software agents
- Ontology enables the reuse of domain knowledge
- Ontology makes domain assumptions explicit
- Ontology separates domain knowledge from operational knowledge
- Ontology helps to analyze domain knowledge

Sharing a common understanding of the structure of information among people and software agents: Ontology provides unique, unambiguous definitions of expressions, avoiding semantic conflicts. For example, there are various portals on the internet which provide medical information or medical e-commerce services. If all these portals share and publish the same underlying ontology of terms, then software agents can aggregate the extracted information from these sites. Also, agents can use this aggregated information to answer users queries or as input data for other applications [10] [6]

Enabling the reuse of domain knowledge: By allowing sharing, an ontology also contributes to the extension of existing work and preventing the duplication of effort. Any interested individual can search the existing knowledge base, use earlier work as a base and contribute further. In order to build a large ontology, several existing ontologies that describe the portions of the large domain can be integrted. One can also use a general ontology, such as Universal Standard Products and Services Classification Code (UNSPSC) ontology, and extend it further to describe the domain of interest. [10]

Explicitly make domain assumptions: In ontology all assumptions are stated clearly, so that there are no errors made due to false assumptions. For example there are several assumptions and various notations that are being used by software programmers across the world, and it becomes difficult for a person who has to maintain this software code if he/she is not aware of these notations and assumptions. Our only intent should be to have clear assumptions that are made explicit. Explicit assumption of domain knowledge are useful for new users, who can quickly learn what a particular term in a specific domain means [10]

Seperation of domain knowledge from operational knowledge: In ontology domain knowledge can be seperated from operational knowledge. McGuiness in 1998 emphasized the importance of having a clear demarkation between domain knowledge and its operational usage as creating a system is quite different than using it. For example, nucelar energy concepts in the domain knowledge area remain the same, but its application as power generator or as a weapon are completely different. Therefore one must have different sets of principles and a clear line drawn between research and its opreational activites. [10]

Analysis of domain knowledge: Ontology helps to analyse existing terminologies and check their relevance as it can be used either to build them further or to use the existing system. [10]

In the next section we will describe the classification of ontologies.

2.3 Types of ontologies

There are several classifications of ontologies in computer science. Van Heijst, Schreiber and Wieringa (1996) classified ontologies based on their use as terminilogical ontologies, information ontologies, knowledge modeling ontologies etc. Gomez-Perez, Fernandez-Lopez and Corcho (2003) classified ontologies based on the level of specification of their relationships as lightweight and heavyweight ontologies. Based on Guarino's (1998) [3] classification ontologies can be divided into following categories (Figure 3) [5], this classification seems most logical and descriptive Therefore, we describe it in brief.

- Top-level ontology
- Domain ontology
- Task ontology and
- Application ontology

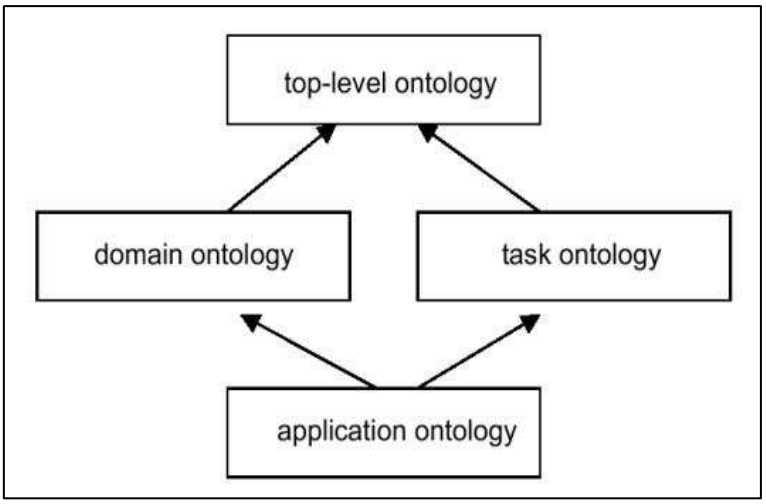

Figure 3: Types of ontologies [11]

2.3.1 Top-level ontology

Top level ontologies are models of common objects that are generally applicable across a wide range of ontologies. They contain a core glossary in which terms and objects in a set of domains can be described. It describes general concepts and events that are independent of a particular problem or domain. [11]

2.3.2 Domain ontology

Domain ontology describes the vocabulary related to a generic domain by specializing the concepts introduced in the top-level ontology. It models a specific domain, or part of the world. It represents the particular meanings of terms as they apply to that domain. For example, the word jaguar can have many different meanings. The ontology regarding the domain of automobile would model the meaning of “jaguar” as a car where as the ontology about the wild life domain would model the meaning of “jaguar” as an animal. [11]

2.3.3 Task ontology

The intention of task ontology is to gather building blocks and create human problem solving processes. Task ontology creates a step by step procedure for solving any problem; designing, manufacturing, scheduling etc. Task ontology is used to solve real-world, problems, by creating a knowledge base system. Task ontology should be able to provide theory (concepts/models) that is required for human problem solving processes. [8]

Task ontology should support following concepts:

- Task roles reflecting the roles played by the domain objects in the problem solving process
- Task actions representing unit activities appearing in the problem solving process
- States of the objects
- Other concepts specific to the task.

2.3.4 Application ontology

Application ontologies are ontologies that are made based on use cases. These ontologies are engineered for a certain activity or specific use. This ontology uses canonical engineering methods and is widely used to explain applications that are cross domains. Application ontology is primarily used in areas of Biotechnology and also data driven outcomes of certain processes. [3]

After the classification of ontologies, the next step is to examine the representation of ontologies. In the next section, the various ontology representation languages are discussed.

2.4 Ontology representation languages

Collaboration and cooperation among various agents as well as the interchange of ontologies across Web led to the development of a new ontology language based on standards such as eXtensible Markup Language (XML), for the formal specification of ontologies. It is aimed to allow the interchange of ontologies across the Web and the representation of the knowledge contained in the ontologies in a human readable form. [12]

The development of ontology representation languages in computer science begins with Hypertext Markup Language (HTML). Figure 4 shows the development path of ontology representation languages.

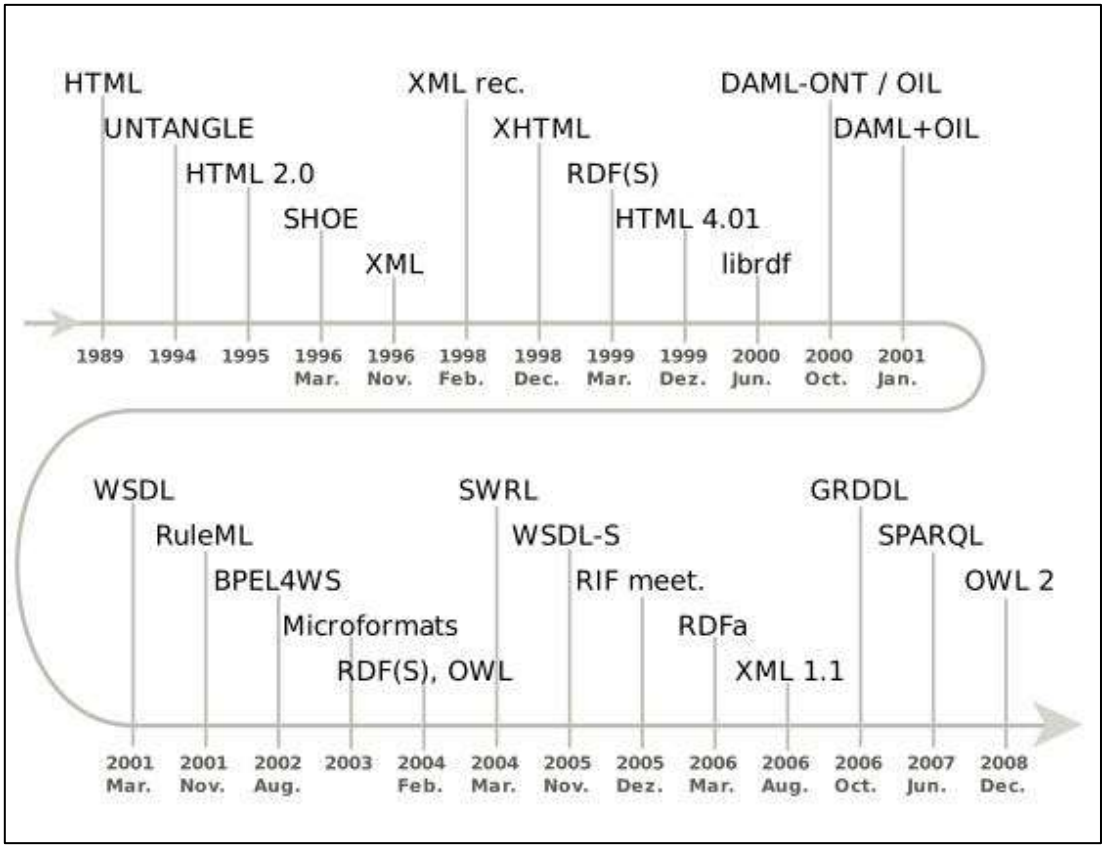

Figure 4: History of ontology related technologies [13]

Figure 5 shows various development languages that were used for the representation of Ontology; every new language in the sequence has some advantage over the preceding one. As the figure resembles layers of cake it was named as Layer Cake Diagram by Tim Berners-Lee. [14]

- XML is used as a base for syntax to allow interoperability on the web.
- XML Schema is used as a database for structuring capability for web objects comparable to database schema.
- RDF's Layer provides a simple language for expressing ontological concepts and relations in XML syntax.
- DAML + OIL or OWL defines more expressive ontologies which use an RDF level to represent instances of ontology constructs.

Although all of these layers are expressed in XML syntax, specific interpreters are still needed to understand each language. In general the higher language interpreters can correctly interpret every layer below their language level, which means that an OWL interpreter can use any embedded or referenced RDF or XML schema data type construct, in addition to OWL specific code. Finally the reasoning and proof methods and "web of trust" layer is near the top, which uses automated proof as well as security and identity features that remains relatively less understood and less mature. At the very top are the User Interface and domain applications that can utilize the entire semantic web to offer more intelligent services. [13]

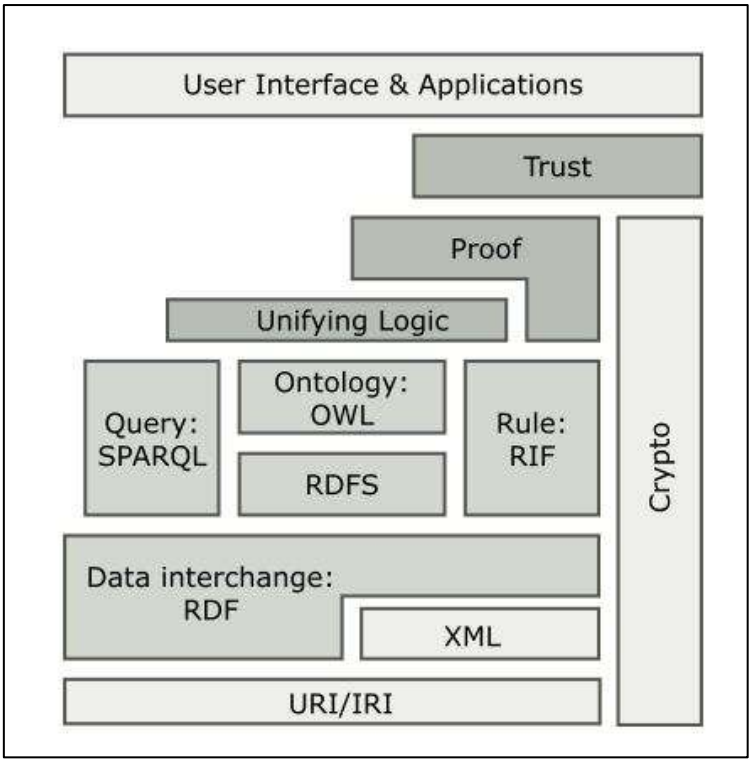

Figure 5: W3C Semantic Web Activity "layer cake" diagram [13]

As discussed above XML furnishes the base syntax for interoperability on the Web but its shortcomings regarding ontology representation language led to the development of RDS and later OWL is evolved. We will see in the following section

why a new language came into existence and how the former language provided a base for the development of the latter language. [12]

2.4.1 XML and XML Schema

XML is a very simple language and does not have predefined tags like HTML. XML was designed keeping in mind the challenge of large scale data publishing. It also plays a very vital role in information interchange on the World Wide Web. XML is application independent and is one of the best mediums for describing information. There are two schemas that XML can use. It can be XML schema or Document Type Definition (DTD). [15] [16]

The advantage of XML over HTML is that XML has a standard syntax for metadata and a standard structure for both documents and data. It also separates data from HTML and stores it outside HTML. [15]

XML was not created with ontology in mind and inferences cannot be made through XML. It has no special features for the specification of ontologies. Therefore, it simply becomes a medium of information interchange covering ontology exchange needs. [15]

XML Schema is based on the syntax of XML and is a relatively richer language than XML, as it offers the definition and structure of XML documents. XML schema is quite similar to database structure; it provides freedom to create tables and similar fashion. [16]

Although XML Schema is a step forward from XML, the nesting of tags still does not have a standard meaning. The semantics are accessible to human, but not machines. It does not provide a means of discussing the semantics of data. It is suited for close collaboration, where domain based vocabularies are used, but not suited for global communication because collaboration can only be supported if there is a shared understanding of vocabulary. This shortcoming led the development of the Resource Description Framework (RDF), which will be discussed in the following section. [16]

2.4.2 RDF and RDF Schema

RDF was developed and designed to offer a common way to describe information which can be read and understood by computer agents or applications. It is not designed to be displayed on the web. RDF is an approach to reference semantics in documents specified in external ontologies. RDF format is <Subject, Predicate, Object>. That means it has URIs along with anchor IDs which are optional For example Subject; Predicate would include properties, relationships, and characteristics, while all these statements would value to a specific resource i.e. object. An object itself can again be described by a further RDF triple, which is reification. RDF statements consist of subject, predicate and object but they have no way to describe the relationship between them or their meaning. [13]

RDF describes the named properties and their values of the resources, while Resource Description Framework Schema (RDFS) describes vocabularies used for

the description of RDF. These vocabularies describe properties, classes or resources and the relationships among them. RDFS has introduced basic ontological concepts or building blocks of ontology, such as the concept of class, subclass properties etc. It defined how concepts and resources can belong to a class or to more than one class, in the same way subclasses describe the properties of hierarchies. These can be used to define that subject and object of a property which belongs to the respective class. [17]

RDF and RDFS are much more useful than XML. Still they have a few drawbacks. For example, they do not support more expressive concepts like equivalence, inverse relations or cardinality constraints. The need to develop a more expressive ontology language gave birth to Web Ontology Language (OWL), which is an extension of RDFS and is more powerful as compared to its predecessors. [13]

2.4.3 Web Ontology Language (OWL)

As discussed in the previous section, to overcome the limited expressivity of RDF and RDFS, a more powerful language Web Ontology Language (OWL) was developed. OWL was also developed in order to create a standard and broadly accepted ontology language. [18]

Some of the advantages of OWL over previous languages are that it offers a vast vocabulary as compared to RDFS and provides greater and stronger interpretation to computer applications.

OWL empowers users to write more explicit and formal concepts in domain models. OWL has very strong syntax and a very detailed semantics. It also has a well organized reasoning support and adequate expressive power. OWL has three sublanguages which will be discussed in the following section.

2.4.3.1 Three sublanguages of OWL

OWL Lite: OWL Lite is the least complex of the OWL languages. It is used for basic classification of hierarchies and for creating simple constraints. It also enables easier and smoother migration of thesauri and taxonomies, as it does not have many complexities. OWL Lite is a subset of OWL DL. [18]

OWL DL: The DL in the name stands for Description Logics, a field of research regarding the logic supported development of OWL DL. Therefore, it supports computation and logic though being expressive, and all conclusions that can be drawn are completely revalidated mathematically or logically. It also ensures that all computation is finished within a stipulated period of time. DL consists of almost all constructs of OWL, an exception being the utilization of sub classes by other classes. [18]

The only restriction of OWL DL is that it loses full compatibility with RDF. RDF should be enhanced so that it can be synchronized with DL. Also, every Legal OWL DL is a Legal RDF but, every RDF is not a valid OWL DL unless enhanced. [18]

OWL Full: OWL Full provides users a great amount freedom of expression. It not only allows a class to be a collection of objects but also allows usages of class as an

object, which was not supported in earlier languages. It also supports the merger of ontology with predefined vocabularies such as (RDF and OWL). [18]

OWL LITE is subset of OWL DL which in turn is a subset of OWL Full. [18]

All processes and conclusions of LITE are available in DL and all processes and conclusions of DL are available in the universal set of OWL Full. [18]

It is recommended that, based on usage, ontology developer should decide which language suits him/her the most. If functions are simple and not very logic driven then LITE may be most optimum solution. However, if a system needs to be expressive and also driven by logic that must be clear and traceable, then the optimum solution would be to use OWL DL.

If a system needs to merge with a predefined vocabulary and needs classes to be used as object at times and for other classes there is no requirement to validate logic, then OWL FULL is the optimum solution.

2.5 Conclusion

We have seen that Owl has been designed to meet the requirements for a Web Ontology Language. It is definitely promising for the future of the Web, in which information is given explicit meaning, making it easier for machines to automatically process and integrate information available on the web. OWL adds a great vocabulary for describing properties and classes: among others, the relations between classes, cardinality, equality, richer typing of properties, characteristics of properties and enumerated classes. Therefore, OWL is also our choice for this work [19].

This chapter has explained the term ontology as well as its building blocks, importance and classification. Then, ontology representation languages were discussed. As stated earlier ontologies are machine readable and needs some way to visualize so that they can be understood better and make the analysis and development easier. The next chapter will examine visualization, its importance and the proposed methods of visualization over the Web.

3 Ontology Visualization

Ontology representation languages were designed to offer a common way to process the content of information, rather than to display it. Ontologies are written to be read and interpreted by computers and not human beings. Therefore, some means of visualization is needed for humans to understand them. For very large and complex ontology visualization, it is vital to help the user to comprehend and use it. Because of the progress of the Web more and more information must be processed, structured and interpreted. Many researchers and users are contributing to the development of various domains. These researchers, developers or users of ontology might not necessarily belong to the ontology development team or have an in depth knowledge of ontology and visualization of ontology will surely be a great aid to such stakeholders.

3.1 Significance of ontology visualization

In this section we will discuss the significance of ontology Later in chapter 4 we will also see that how the visualization of ontologies can aid different stakeholders groups within a specific domain.

Visualization is forming a picture of something in order to understand or remember it. It is a means by which our minds comprehend information much more easily than that information which is written in long paragraphs. Visualization techniques, in general, facilitate a better understanding of complex systems. Much research has been done within the field of visualization of ontologies. Bergh has argued in his master's thesis that ontology visualization techniques are the only tools currently available that facilitate the understanding of ontologies. [20] Ontology visualization tools not only aid the process of understanding, but also retain the expressiveness of ontology by hiding the formal terminology that is used behind it. [21] Those stakeholders who have an interest in understanding or using the ontology do not need to know ontology representation languages. Moreover, it fosters the development, extension and editing process.

We have seen that ontology visualization tools are very useful for the understanding and development of ontology. In the subsequent section, we will describe what characterizes a good ontology visualization tool.

3.2 Characteristics of good visualization tool

A good ontology visualization tool should be able to display what it is supposed to visualize. It should offer some way for the user to interact with it, and it should be reasonably scalable. [1]

Display

As discussed in Section 2.1, there are several components of ontology such as classes, relations, properties etc. A good visualization tool should support the presentation and visualize them graphically. The hierarchical structure must be contrasted or related to one another so that the user can differentiate the items easily by using color, size, shape or layout alignment. [22]

Interaction

Simply displaying the contents of ontology does not make ontology visualization tool good enough, it must have interactive possibility as well. This means it should contain good navigation techniques such as overview, zoom, filter, details-on-demand, history and extract. For example within a large ontology visualization the user should be able to experience where he is within the hierarchy, where he can go and from where he came from. [22] [1]

Scalability

Little is known regarding the scalability issues in visualizing large hierarchies. Katifori has stated in his survey that current systems face problems in visualizing more than 10000 visible nodes. A good ontology visualization tool should not have a loading problem within the expected/defined limits. [1]

These characteristics have formed the basis of our evaluation design and these points with some additional criteria will be revisited in more detail in Section 5.1

3.3 Analysis of existing ontology visualization methods

"Visualizing the semantic web, XML-based internet and information visualization" is a pioneering book which has presented the state of the art in the area of semantic web visualization. It focuses on several topics, including the visualization of semantic data and meta data, topic maps, ontology visualization, SVG/X3D for semantic web, etc. [23] Katifori [1] has provided a detailed overview of the existing visualization methods as well as their pros and cons. These methods are grouped under many categories: indented list, node-link and tree, zoomable, focus+context, space-filling, and 3D information landscapes. Catenazzi [24] suggested that among the above six categories the most frequently used for ontology visualization are the first four, which we describe briefly:

- **The indented list** methods represent the classification of the ontology following the file system explorer-tree view. One example is the Protégé Class browser. These indented list methods are instinctive and simple to implement. These findings aret confirmed as a consequence of user evaluation [25] when different visualization methods are compared. Their main disadvantage is that they represent a tree hierarchy and not a graph, and they also do not visualize role relations [24].
- **The node-link and tree** methods are another approach that is frequently used for ontology visualization. A set of interconnected nodes represents the ontology in this case. By this method the user gets a clear overview of the hierarchy and connections. However, it can produce a cluttered display when more than a hundred of nodes are used in this visualization. Some examples of these methods are OWLViz , OntoViz and RDF Gravity [24].
- **The zoomable** methods are used when the nodes in the lower levels of the hierarchy are located inside their parents and displayed in a smaller size. A user may zoom-in to a child node in order to enlarge it. One illustration of this approach is Jambalaya [26] [27]. Such methods appear successful for browsing in order to locate a specific node or nodes. However, these are not effective in

displaying the overview of the hierarchical structure, and are likely to produce disorientation after zooming-in several times. In order to avoid this loss of context problem, some orientation clues can be added. [24]

- **The focus + context** methods are used when the node being focused is in the centre, and the rest of the connected nodes are around it, although usually reduced in size. An example of this is the TGVizTab Protégé plug-in [28]. This technique is effective for providing a global overview, focusing on specific nodes, and quick browsing. However, it may result in a messy visualization and the user may find it difficult to create a mental model of the ontology user as the graph is continuously redrawn and node positions are re-arranged. [24]

Keeping in mind that there are various methods and approaches to visualizing ontologies, the question arises as to which method should be chosen. The specific user task that the visualization method should support is the main point to be considered, e.g. overview, zoom, filtering, editing, etc. Ontology scalability is also another factor that requires consideration. The end-user profile should also be considered. An ontology expert can easily understand the ontology "syntax" yet, in order to be able to use them to create a conceptualization of the domain it is the domain expert who will require the concept of the ontology constructs [24].

In conclusion there is no one method that is always the best. Katifori [5] suggests giving the user a choice to choose among several visualization methods. Despite the differences in the techniques given above, the following features should always be available [24]:

- A query mechanism which identifies nodes and relations of interest
- Filtering to facilitate hide or display a part of the ontology
- Reduction of information overload
- The incorporation of reasoning mechanisms, and the visualization of their effects

3.4 Web-based ontology visualization

This thesis focuses on web-based visualization methods. Therefore it becomes more relevant for us in reference to eProcurement ontology in which different stakeholders are scattered EU-wide. In this section, we will discuss the need for and usefulness of web-based visualization methods. Thereafter, the discussion will move to the current technologies used in web-based visualization.

The scientific community has been quick to exploit the potential of the Web for marketing and promotion, and research groups present their work as multimedia documents, with images and animations of their results. However this is just the beginning. These images and video clips are passive views of the research, in which the visualization is created by the publisher of the data and the viewer merely looks at the pictures as though leafing through a book. The medium of the Web offers far greater opportunities. It also encourages the active participation of the viewer in the way data is visualized. Indeed, the potential of Web can be exploited by using it as an analysis tool, rather than just as a tool for the presentation of results [29]. From the perspective of ontology visualization, it offer great possibilities for sharing and collaboration between users and developers are in different parts of the world. A few advantages of web-based visualization methods are be listed below:

- No special software is needed as the visualization runs in a standard web-browser
- It offers platform independence, providing the same experience across different operating systems
- It is easy for the developers to push new updates and fix bugs without requiring any action from the user
- Fosters collaboration and development

The Java applet is one of the technologies used for web-based ontology visualization. It has a number of advantages such as, portability and independence of the environment/browser. Java is safe and the applet environment is secure. It can be loaded when needed and there is no need to install or manage them. It began as a way to integrate desktop applications within the web browser environment. Despite its initial success, its success slowed down due to slow download and start up time unpredictable behavior on different operating systems and lack of standard security model [21]:

Flash/Flex based technologies have provided several advantages over java applets. They can also be embedded in web pages, but are compatible with different operating systems and have much faster load time than java applets. Adobe[1] claims that, as of December 2010 more than 98.8 % of internet users have the flash plug in installed on their system.

Recent trends on the web show that, by the employment of Ajax technology or jowl many of Java-applets issues can be resolved [30]. This approach encourages user interaction by way of utilizing asynchronous Hypertext Transfer Protocol (http) requests to the server. This avoids reloading the whole page after a user action and only fetches the relevant parts of the page which need to be updated, thus providing a more responsive experience. Ajax applications rely a lot on Java Script that is well supported by major web browser leading to consistent visualization [21].

3.5 Conclusion

In this chapter we have seen the significance of ontologies, characteristics which make ontology visualization tool successful, and the related work done. Finally the issues related to web-based visualization tools were discussed. The next chapter begins with the Virtual Company Dossier ontology in eProcurement domain which will move this work further from general theoretical background to a domain specific.

[1] http://www.adobe.com/products/player_census/flashplayer/version_penetration.html (Accessed 14th July 2011)

4 eProcurement and Virtual Company Dossier ontology

One of the objectives of this thesis is to investigate and test the deployment of a web-based visualization method for the eProcurement domain. In this chapter Pan European Public eProcurement On-Line (PEPPOL) and the Virtual Company Dossier (VCD) service are introduced. PEPPOL is a unique project of its kind because of its heterogeneity and complexity. The role of stakeholder's changes in each scenario, legal specifications are different in different countries and many more. For our thesis VCD system's framework, VCD's ontology and associated stakeholders are relevant and we will address these within the framework of our thesis.

European Union (EU) spent annually 1,500 billion Euros every year, which accounts for 16 percent of Europe's GDP. PEPPOL an EU project cofounded by European Commission was set up to leverage and facilitate EU-wide interoperability in public eProcurement without borders, which conjointly exist with national infrastructures. This consortium comprises of 18 leading public eProcurement agencies in 12 countries. PEPPOL envisions a future where any company in the EU (in particular small and medium-sized enterprises) can communicate electronically with any European governmental institution for the entire procurement process. Project has several building blocks which enables seamless use of EU-wide eProcurement solutions ranging from eSignatures to electronic invoices. One of the key building blocks out of all building blocks of the PEPPOL project is VCD and will be discussed in the following section. [31] [32] [33]

4.1 Virtual Company dossier (VCD)

VCD has been developed to address the demand for better interoperability in eTendering and provide simplification, transparency and electronic monitoring of supplier qualifications in public procurements. It is an electronic cross-border document container that carries attestations and candidate statements required in public tendering procedures.

Figure 6 shows the VCD system that provides a comprehensive set of tools supporting public buyers and suppliers in the tendering phase, for both national and cross-border procurement in the EU. The VCD tools can be used by suppliers to identify and collect suitable evidences and to submit their qualifications as part of tenders to public buyers. They can in turn use the tools to receive legally accurate qualifications in a structured way and to prove the suitability of tenderers. The Figure 6 also shows the transnational procurement procedures, which has several difficulties for the participating parties involved. The set of legal rules that apply to each party differs. Therefore, the qualitative selection criteria do not point towards any specific evidences as some evidences may not exist in a country or may exist in a different form. Besides, competent issuing authorities are not known across borders as a result the documents have to be legalized and may require to be translated by the Economic Operator. Some of these barriers may be reduced or overcome by the Virtual Company Dossier (VCD) that aims to provide an interoperable electronic document solution that supports the exchange of evidences. Contracting Authorities can use it for monitoring eligibility and suitability of national and foreign candidates. [31] [33] [32]

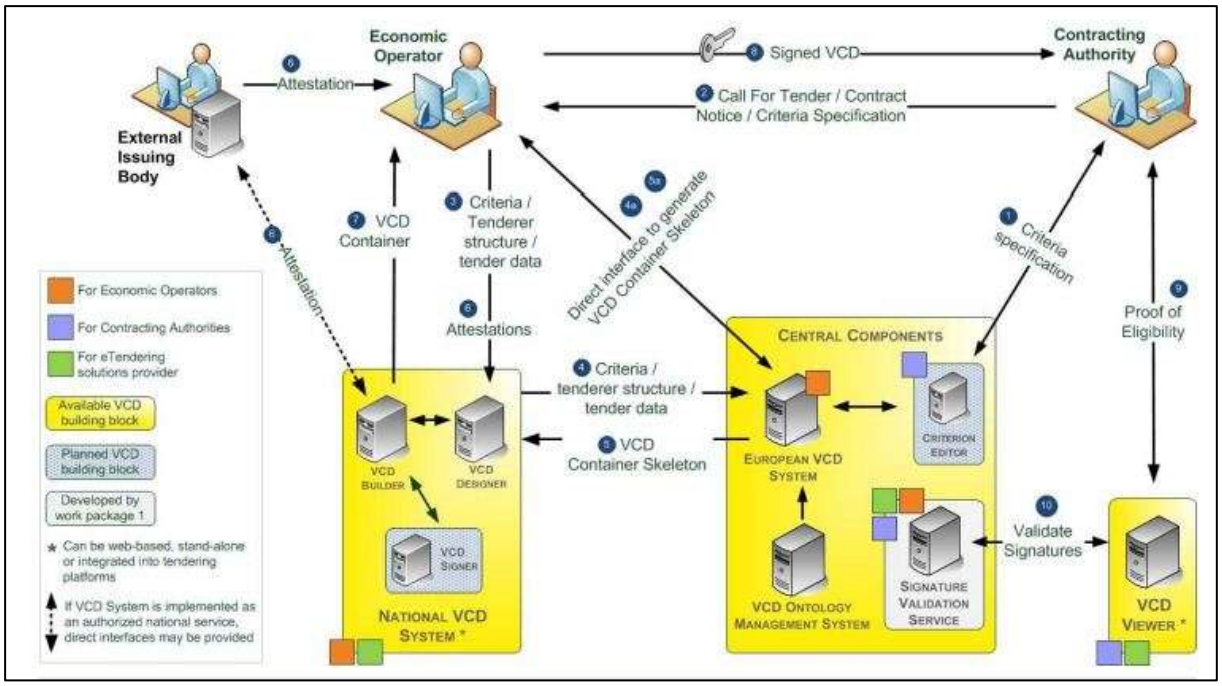

Figure 6: Working of VCD system showing stakeholders and eProcurement process [31]

Figure 6 shows the three main components of the VCD system viz. Central Components, National VCD System (NVS) and VCD Viewer. The rules and criterions are represented as machine interpretable ontologies. The editing and management functionality for the different ontologies are provided by the Ontology Management System (OMS). The Ontology Management System is capable of being used simultaneously by the different ontology editing teams in order to keep up to date the legal rule sets. In the next section we will detail the logical parts of VCD ontology. [31]

4.2 VCD ontology

Figure 12 shows the complete VCD ontology, which is split into five logical parts. These five logical parts can be further grouped into 3 groups (Figure 7):

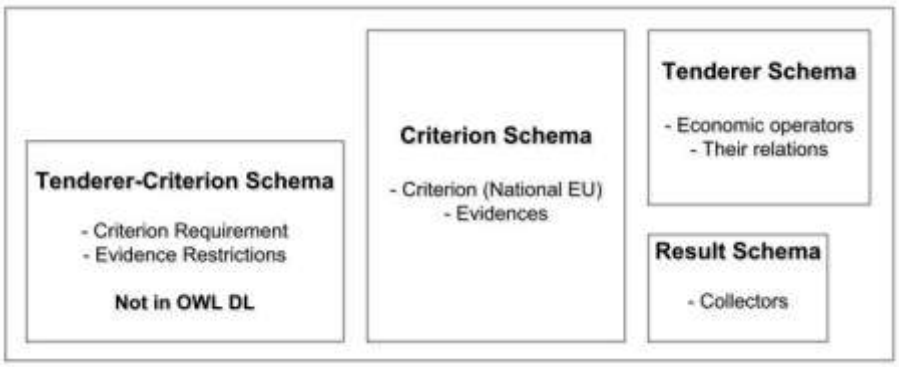

Figure 7: Logical parts of VCD ontology [34]

1. **Group 1:**The first group contains the common-schema which has some common classes and properties used by the other two groups

2. **Group 2:** Second group comprises of criterion-schema, tenderer-schema, tenderer-criterion-schma and contains the schemas for specifying criteria, evidences, tenderer structure and the relationship between them.
3. **Group 3:** Third group has collector-schema and contains the schema for specifying input and output for the reasoning steps.

These logical parts of the VCD ontology are described below:

Common-schema

Figure 8 shows the visualization of ontology of common-schema. It contains classes and properties that will be used throughout the VCD ontology, such as owlx:AbstractClass and owlx:InstantiableClass. Both are subclass of owl:Class but distinguishes by instance existence possibility or not. [34]

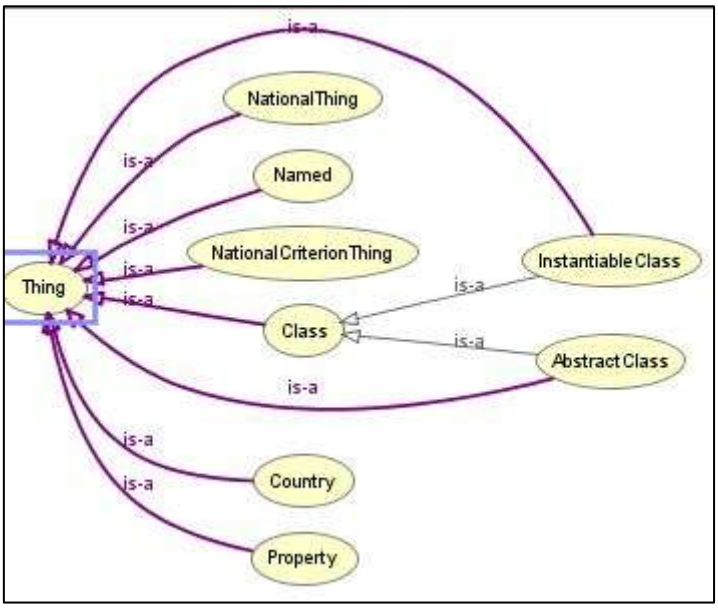

Figure 8: Common-schema [34]

Other classes which are defined in this logical part are Classname, Propertyname, Named class. It also contains classes of National Objects annotating open tasks. [34]

Criterion-schema

Criterion-schema (Figure 12) defines the schema for criteria and evidences. They can be grouped hierarchically using the class CriterionGroup to model the structure of legal documents. In this schema there are five instantiable classes viz. EUCriterion, NationalAtomicCriterion, NationalCombinedCriterion, VirtualEUCriterion and NationalPseudoCriterion. [34]

VirtualEUCriteria are not reflected in EU directive, but are only included to allow mapping of national criteria that could not be mapped otherwise. NationalPseudoCriteria are used where there is no correspondence in the national law, still EU criterion might need to be proven. At the national level, combinations of criteria are possible. Tenderer has to prove all sub-criterions to prove combined criteria. [34]

Evidences can be issued by a competent issuing body or by a tenderer and can be assigned a priority. Evidences can be distinguished as primary or secondary evidence. Primary evidence is “backed by law” whereas secondary evidence is not backed by law but still often accepted by contracting authorities. [34]

Tenderer-schema

Tenderer-schema (Figure 9) specifies how a tenderer structure is represented in the ontology. Different aspects of a tenderer are modeled in distinct classes. [34]

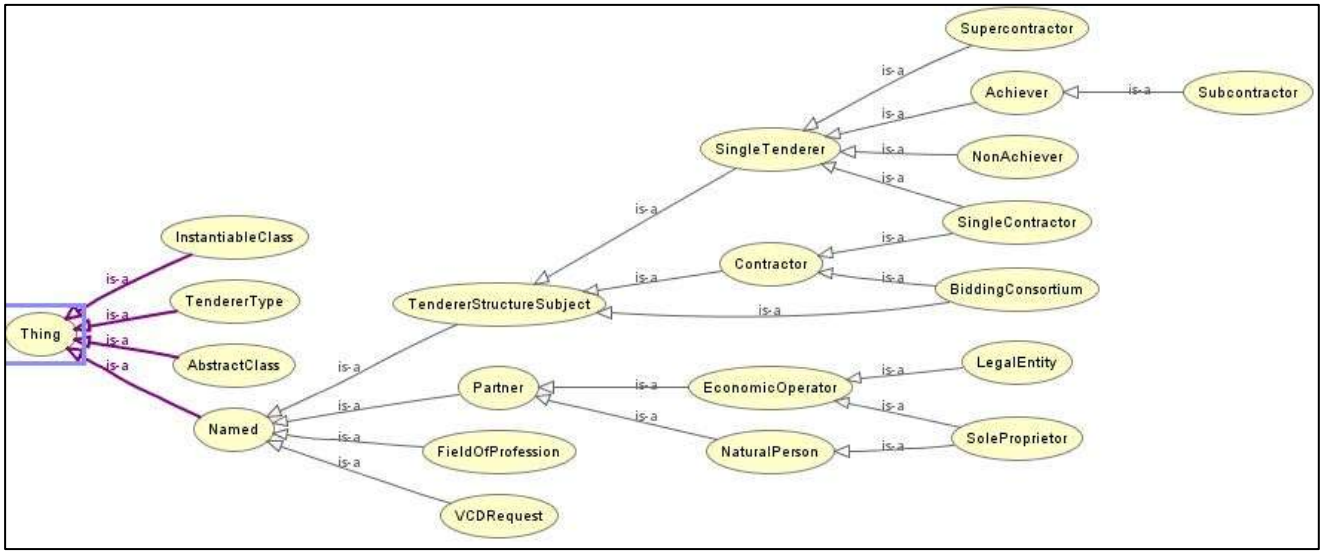

Figure 9: TendererSchema [34]

This Schema models two different dimensions of a company in eTendering process. On one hand, they have characteristics that describe them as a company and on the other hand their characteristics describe their role in the tendering process. VCDRequest is also defined here to create a VCD request. Various subclasses of TenderStructureSubject describe how different companies cooperate for the tender at hand. Partner and its subclasses show the structure of companies. [34]

All the classes in this schema are instances of the class TendererType specified in tenderer-criterion-schema which is the next section. [34]

Tenderer-criterion-schema

Tenderer-criterion-schema (Figure 10) is defined for two distinct kinds of rules.

First, types of rules are ones that govern which criteria have to be proven by whom. Those are called the criterion requirements and are represented by the class CriterionRequirement. These are only required if we want to provide a suggestion of probable criteria based on tenderer data; they are not needed if we expect the tenderer to provide the criteria he wants to prove by himself. [34]

The second kind of rule specified by this schema governs limitations of availability of evidences. For example, criminal records might only be available for natural persons,

but not for legal entities. These rules are called evidence restrictions and are represented by the class EvidenceRestriction. [34]

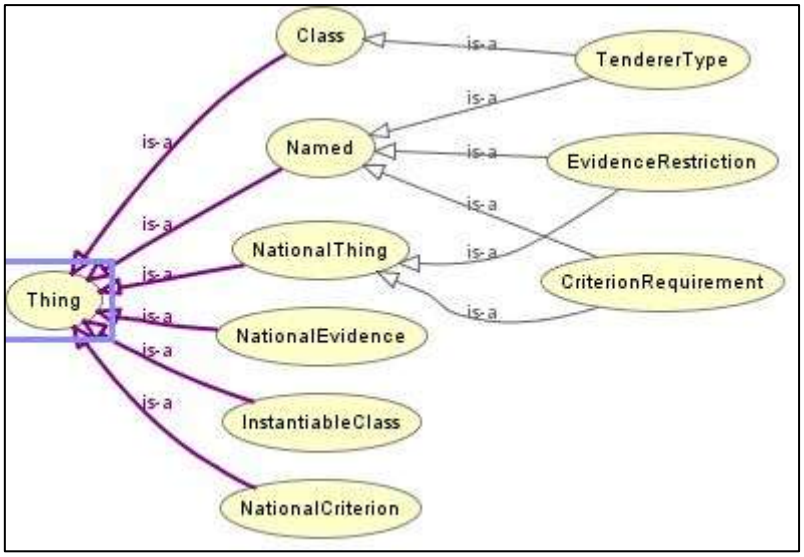

Figure 10: Tenderer-criterion-schema [34]

Collector-schema

Collector-schema (Figure 11) provides the classes and properties used as input and output for the reasoning deriving evidences from required criteria. [34]

The input for the reasoning is specified by linking the Tenderer Structure Subjects to the criteria we want to prove for them with the wantsToProveCANC property. [34]

The output of the reasoning will be grouped into so-called collectors. The collectors are structured so that they show the actual path from ca-national criteria via EU criteria to the national criteria and finally evidences. [34]

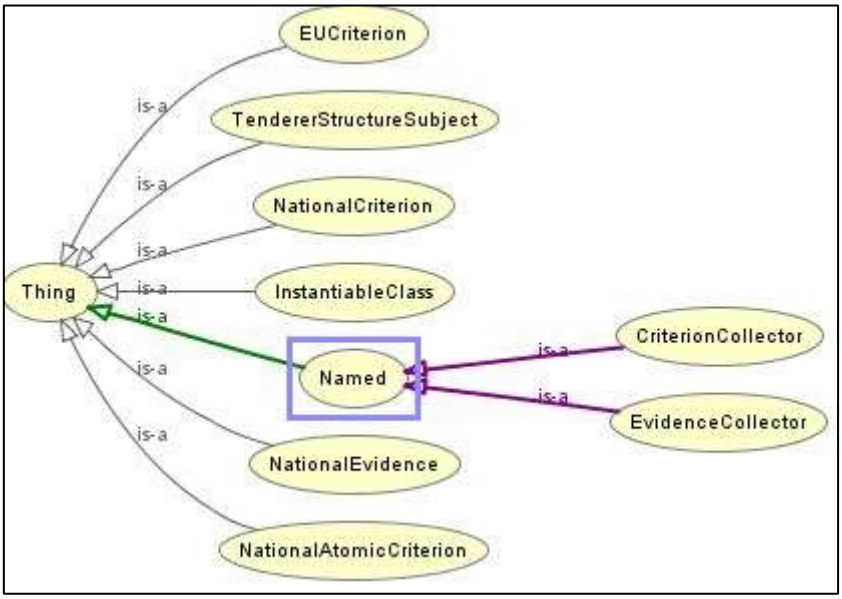

Figure 11: Collector-schema [34]

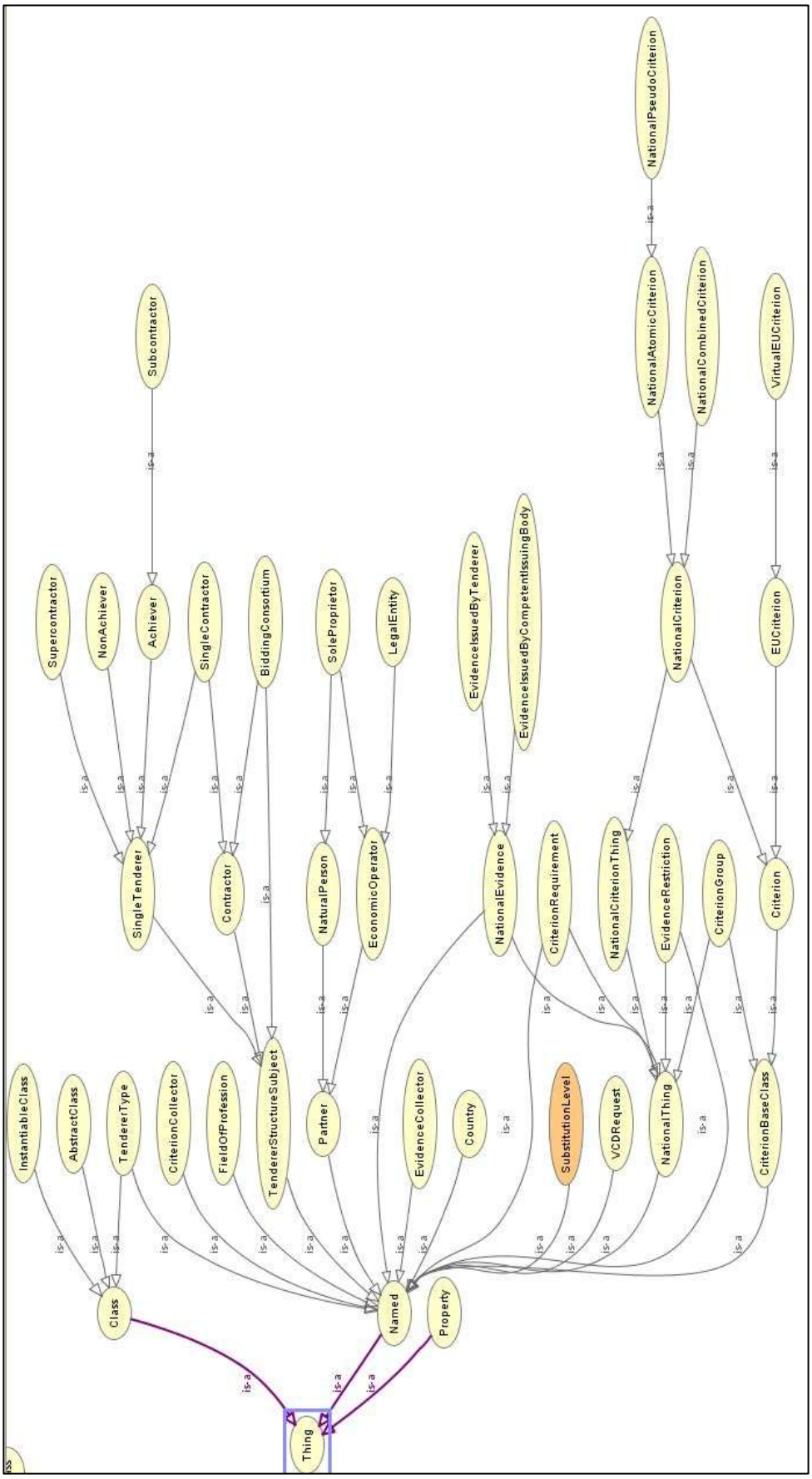

Figure 12: VCD ontology [34]

4.3 Stakeholders in VCD System

Stakeholders in PEPPOL play an important role as the project establishes interoperable solutions supplementing national frameworks, rather than replacing them. Stakeholder (groups) can be differentiated according to their geographical coverage and field of interest with regard to VCD System. Within a complex system the roles of stakeholders changes in different frames. From Figure 6 we can see three major stakeholder groups' viz. Economic Operators (EO), Contracting Authorities (CA) and eTendering Solution Providers (SP), but CA, EO, or SP can be influenced by many other stakeholders, which make it more important and complex in OMS. We will briefly discuss their interests in VCD ontology visualization. [32]

Solution Providers

eTendering solution providers are the system builders of the OMS. They develop, edit and extend the VCD ontology. They are interested in understanding and editing the ontology. For them visualization fosters the development process, moreover they prefer a visualization tool blended with ontology editor tool.

Contracting Authorities

Contracting Authorities are interested in ontology visualization because they specify various criterions for tendering. They work on criterion editor to edit the criterion specification for EU criterion and National criterion. Other stakeholders like European Commission, Legal departments and other beneficiaries also influence CA. CA is interested in editing, and understanding VCD ontologies where as other stakeholders who influence CA might be only interested in viewing and understanding of ontologies.

Economic Operators

A Economic operator has interest in understanding the VCD ontologies so that they he/she can understand the criterion and evidences.

Within EU the role of stakeholders also changes for example EO can also become a CA in a given scenario or vice versa. All together SP, CA and EO are all interested in ontologies either to understand various logical parts of VCD ontology or to edit the VCD ontology to define criterion/evidences.

4.4 Conclusion

In this chapter we have seen the VCD and the logical parts of VCD ontology within the domain of eProcurement. We have also seen the perspective of various stakeholders who are interested in VCD ontology. With this the background of thesis work is concluded. In the next chapter we will begin with the design of evaluation strategy to determine the most suitable design of a evaluation strategy to determine the most suitable ontology visualization tool for the VCD ontology.

5 Evaluation design

In order to evaluate web-based ontology visualization methods and get fair results, a good evaluation design is necessary. While the previous sections discussed the importance of ontology, visualization of ontology and ontology-schema in VCD, in this chapter we develop our strategy for evaluation of the ontologies.

In most of prior approaches to group the evaluation criterion, either unipolar or bipolar scales are used to evaluate the criterion. A unipolar or bipolar evaluation result show the presence or absence of a particular criterion, but it does not gives the magnitude of the criterion being measured. For this thesis we have chosen not to use uni-polar or bipolar scales but a numerical scale. A score from 1 to 5 is assigned to every criterion. Numerical scale gives the magnitude of the criterion being measured and tells us not just its presence or absence rather throws some light on its magnitude also. [1]

5 = Excellent
4 = Good
3 = Fair
2 = Poor
1 = Bad

After choosing the numerical scale, the next step in evaluation design is listing and grouping of relevant criteria (Figure 13). Since, visualization method is a piece of software therefore it is appropriate to choose the criterion based on the recommendations of Sommerville (Sommerville, 2004) as functional and non functional. Accordingly, we have grouped all criterions in functional and non-functional criterion. An ontology visualization tool shall display its elements and provide a means to navigate within the displaying ontology therefore; functional criterions are further divided into two sub groups viz. elements display and navigation. For a web-based tool apart from its functionally other issues like availability of software, support, performance, integration with a CMS are the other issues which shall be addressed too. Hence, non-functional criterions are also divided into two sub groups viz. software and performance. [35]

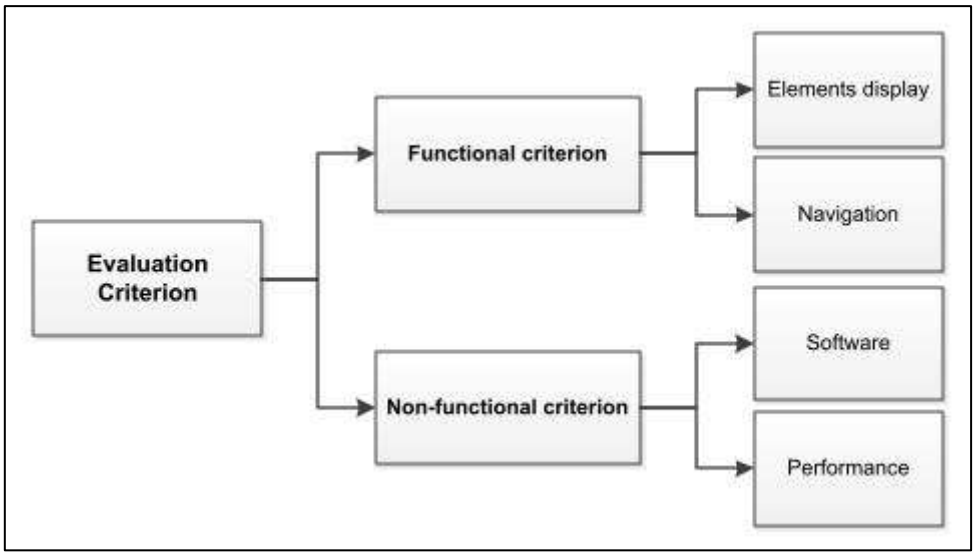

Figure 13: Criterion classification

Functional criterion:

1. **Elements display:** As explained in the ontology section, ontology is composed of several elements and those elements should be displayed in a way that user can easily retrieve the information. Criterions chosen in this group are based on the suggestions of Katifori in his visualization tool survey. [1]

2. **Navigation:** Navigation techniques help the user to interact dynamically with the ontology. This group is characterized by the categorization of tasks based on the task analysis proposed by Shneiderman [36] who presented seven high-level tasks that an information visualization application should support. Apart from seven high-level tasks editing and query customization possibilities will also be explored.

Non-functional criterion:

1. **Software:** This group contains the criterion which focuses on type of software and general software related issues.

2. **Performance:** Performance and scalability related tasks will be evaluated in this group.

5.1 Evaluation Criterion

Functional criterion

Elements Display

1. **Classes:** The visualization method should display all the ontology classes, either directly or upon the request of the user, providing at least their name, in an intelligible manner. A score of 5 indicates that the respective method displays all the classes and 1 does not display classes at all. [1]

2. **Instances:** The actual data that is associated with the ontology are the instances. In many cases user is interested in instances and the information associated with it. However, representing them as nodes connected to a class is not always effective because of their great number and other alternatives should be used, e.g., presenting the instances of a selected class as a list within a separate window. [1] A score of 5 indicates that the respective method displays instances and 1 does not display instances at all.

3. **Taxonomy (Isa relations):** To understand the inheritance relations between classes, it is important to display the taxonomy on which the respective ontology is based. The system should display, in a hierarchical representation. Partial views, allowing the user to focus on a portion of the taxonomy, are also a desirable feature. Score 5 means the respective method displays taxonomy and 1 does not display taxonomy at all.

4. **Multiple inheritance:** In case where a class has more than one parent along with representation of the effective taxonomy is not always easy to display. The cases where a class has more than one parents are not easy to represent in combination with an effective representation of the taxonomy. It is desirable for the visualization to indicate nodes with multiple parents and provide efficient means to view all nodes which are direct ancestors. It should be noted here that many of the presented ontology visualizations support multiple inheritance by replicating child nodes under all their parents. [1] Score 5 means the respective method displays multiple inheritance and 1 fails to do so.

5. **Role relations:** Role relations and multiple inheritances are two types of links that transform ontology from a hierarchal structure to a graph which is more difficult to represent than graph. Apart from the link that is visible, a label with the link name shall also be displayed with an option of display/hide. [1] Score 5 means the respective method displays role relations and 1 fails display.

6. **Properties:** The properties associated with an entity are also very important and a visualization method should show them, either on the main ontology visualization or within separate window. [1] Score 5 means the respective method displays properties and 1 does not display properties at all.

Navigation

1. **Overview:** Visualization method shall an overview of the entire collection, view total number of classes, total number of instances and depth of hierarchy either directly or on demand of user. [36] Score 5 means the respective method gives an overview and 1 means no overview display at all.

2. **Zoom:** Visualization method should have an option to zoom in on items of interest. When zooming, it is important that global context can be retained and views the sub hierarchy. [36] It is important to provide user an option of focusing on a specific node. Score 5 means the respective method offers zooming option and 1 does not offer zoom option.

3. **Filter:** Visualization method should have an option to filter out uninteresting items. [36] Score 5 tells that visualization method has a filter option while 1 means no filter option.

4. **Details-on-demand:** Visualization methods should have a possibility of selecting an item or group and get details when needed like class/instance properties, class siblings, number of subclass etc. [36] Score 5 offers details-on-demand while 1 offers no details-on-demand.

5. **Search:** Keyword search is very essential when ontology is large and complex. Although it is not directly relevant to ontology visualization, but visualization method which contains keyword search option definitely makes it navigation friendly for user. [1] Score 5 will be assigned if respective visualization method offers keyword search and 1 if it has no keyword search.

6. **History:** Keep a history of actions to support undo, replay, and progressive refinement is very useful which makes it an indispensible option. It will allow user to retrace his/her steps during browsing. [36] Score 5 means the respective method keeps history and 1 means no history saving option at all.

7. **Extract:** Visualization method shall allow extraction of sub collections and query parameters. [36] Score 5 will be given if the visualization method allows extraction while 1 is its absence.

8. **Editing:** It is probably not useful for a normal user but for a developer it is important to know if visualization method offers adding, editing and deleting of ontologies. Score 5 means the respective method allow user/admin to edit the ontology and 1 means no editing option.

9. **Query customization:** User can run, customize query and search through the results of the query. Score 5 means the respective method offers query customization and 1 shows no query customization.

Non-functional criterion

Software

1. **Web-based:** This criteria check if the visualization method is Web-based or not. Web-based methods includes methods which can run directly on a web browser (Firefox) or through a Firefox plug-in. Score 5 signifies that respective visualization method is 100 % web based and 1 shows that it is not a web-based method rather a desktop-based

2. **Software availability:** This criterion evaluates if the software is open source, free, shareware or commercial. Score of 5 shows that the evaluation method is an open source method and is freely available and 1 signifies purely commercial.

3. **Future Support/Development:** It is also important to investigate the possibility of future support and development so that the respective method can be updated, extended and developed if required. Score 5 means the respective method has an active support and development whereas score 1 shows that so significant development has been done in the last 1 years.

4. **Plone Integration:** eProcurement project is Plone based hence it would be interesting to know the possibilities of plone integration. Score 5 shows that the respective method can be completely integrated with Plone based website whereas rating 1 signifies that it cannot be at all integrated with Plone.

5. **Collaboration:** Collaboration will throw a light on the possibilities of working and collaboration over a network. Score 5 on a scale of 5 shows that the respective method can be collaborated over a network and 1 shows no possibility of collaboration.

6. **User group management:** Various stakeholder groups are interested in various aspects of ontology and this criterion will find out if various user groups can be managed and administered. Score 5 means all the user groups can be managed and 1 means no possibility of user group management.

Performance

1. **Scalability:** Little is known about scalability of visualizing large ontologies. Ernst and Storey [1] in a user survey categorized them in five categories: 1 for less than 100 nodes, 2 for nodes between 101 and 1000, 3 for nodes between 1001 and 10,000, 4 for nodes between 10,000 and 100,000 and 5 for more than 100,001 nodes. [1]

2. **Loading time:** Loading time is very important when visualization method is web-based and ontology is very large. Score 5 means the respective method takes less than 10 seconds in loading and rating 1 means loading time is more than 5 minutes.

3. **Reaction time:** Checks the reaction time after mouse click. Score 5 means the respective method has a reaction time less than 5 second and rating 1 means reaction time is more than 1 minute.

5.2 Evaluation matrix

Based on the grouping criterion described in the Section 5.1, we have prepared a matrix for evaluation of visualization methods (Table 1).

Criteria			Visualization method
Functional	Elements display	Classes	
		Instances	
		Taxonomy (Isa relations)	
		Multiple inheritance	
		Role relations	
		Properties	
		Total group weight	
	Navigation	Overview	
		Zoom	
		Filter	
		Details-on-demand	
		Search	
		History	
		Extract	
		Editing	
		Query customization	
		Total group weight	
Non-functional	Software	Web based	
		Software availability	
		Future Support/Development	
		Plone Integration	
		Collaboration	
		User group management	
		Total group weight	
	Performance	Scalability	
		Loading time	
		Reaction time	
		Total group weight	
Total weight			

Table 1: Evaluation matrix

5.3 Conclusion

In this chapter we have designed an evaluation strategy. We have also defined various criterion and scale based on score to evaluate visualization method. Finally we have prepared an evaluation matrix which we will use in our evaluation in the subsequent chapters.

6 Evaluation of ontology visualization tools

Among the few available comparisons of comparisons of ontology visualization tools, - Katifori has done the most extensive and detailed survey in his work. [1] A focused comparison based on a specific requirement is seldom done. In this work we have focused on web-based visualization tools to meet the needs of future challenges. In the past, groups have focused on analysis of single methods. For example Katifori [25] has done a comparative study of four visualization methods for information retrieval task by choosing a single tool i.e. Protégé and has evaluated plug ins used in Protégé class browser. Since then, existing ontology visualization tools have become better and new technologies have emerged to meet the requirements of the user. Therefore, instead of trying variation of the same technology, we evaluated four different technologies to get a broad view of the current possibilities of visualization tools. There are not many web-based visualization tools available currently. In our research we have found the following web-based visualization tools: FlexViz[2], Jambalaya plugin[3], GoBar [1], GrOWL [37], jOWL TouchGraph visualization (Experimental)[4] and Plone ontology[5]. These tools can be grouped, based on the technology they are using (Table 2):

Technology used	Tools
Adobe Flex	FlexViz
Java applet	Jambalaya, GrOWL
TouchGraph	jOWL TouchGraph visualization (Experimental)
GraphViz	Plone ontology, GoBar

Table 2: Visualization tools grouping based on technology used

6.1 Selection of visualization tools

As discussed above we have analyzed the following technologies and chosen four visualization tools which are web-based:

Adobe Flex: FlexViz is a graph based visualization tool with a very simple and user friendly interface. It generates a shockwave-flash (SWF) file which is portable and can be embedded in a web page. It is very interactive tool. [38] There is only one tool available which used Adobe Flex hence a clear choice.

Java applet: Jambalaya is a plug-in created for Protégé and uses Shrimp to visualize the knowledge bases created by user. Jambalaya can also be used as a java applet which can be embedded in a web page. Running as a standalone java

[2] http://www.thechiselgroup.org/flexviz (Accessed 14th July 2011)

[3] http://www.thechiselgroup.org/jambalaya (Accessed 14th July 2011)

[4] http://jowl.ontologyonline.org/TouchGraph.html (Accessed 14th July 2011)

[5] http://plone.org/products/ploneontology (Accessed 14th July 2011)

applet gave us a strong reason to choose Jambalaya among other plug-ins created for Protégé. [26] GrOWL can also be implemented as a Java applet or a standalone Java application. [37] We have chosen Jambalaya over GrOWL for three reasons. Firstly, A demo of Jambalaya applet is available online and no demo is available for GrOWL. Secondly, Sourcecode for Jambalaya is available on sourceforce and link indicated for sourcecode of GrOWL is dead[6]. Thirdly Jambalaya can be used as a Protégé plug in as well as applet.

Touch Graph: David Decraene in his website has experimented jOWL a javascript library for traversing OWL RDFS documents based TouchGraph like visualization. Although it is still in the development phase but because of its light score interactivity it can offer a promising future for ontology visualization over WWW which has motivated us to evaluate and analyze it. [30]

GraphViz: There are two ontology visualization methods which use GraphViz[7]. Plone ontology can easily be integrated in Plone CMS Therefore, Plone ontology is a complete solution where one does not have to worry about its integration with CMS. Sourcecode for Plone ontology is available but sourcecode of GoBar could not be found. Therefore, we have decided to evaluate Plone ontology instead of GoBar as our fourth choice.

6.1.1 FlexViz

Graphs are the basic concepts in discrete mathematics and data structure. The applications of graphs are very extensive and vary from common events to complex mathematical or computer science problems. The building blocks of a graph are vertices (nodes) and edges. Ontologies can be represented as a graph by using the basic properties of graphs, such as directed graphs.

FlexViz is a graph based visualization tool written is Adobe Flex. It supports single ontology browsing. Nodes of graph are mapped as concepts and relationships (edges) between nodes (eg. “is_a”, “depends_on”) are represented as arcs. It is designed to provide a light-score, interactive, and visually accessible ontologies on Web. [38]

Figure below (Figure 14) shows a demo of FlexViz. As discussed in Section 3.4, web-based visualization primarily render static images or text representations. However, FlexViz greatly enhance user tasks and promotes new technologies by making it very interactive and light-weight.

[6] http://www.uvm.edu/~skrivov/growl/ (Accessed 4th July 2011)

[7] http://www.graphviz.org/ (Accessed 4th July 2011)

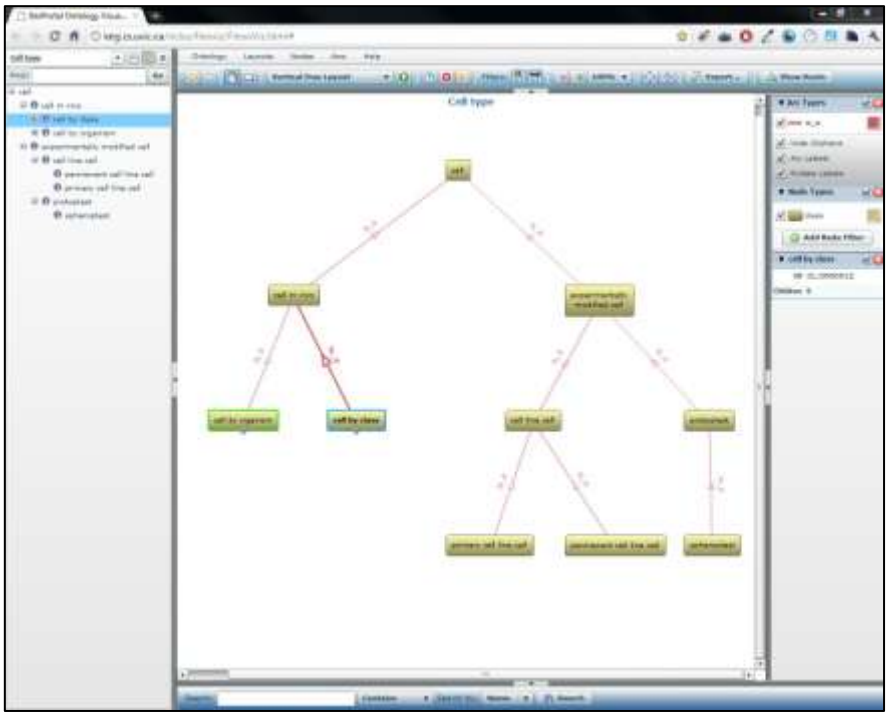

Figure 14: FlexViz demo[8]

FlexViz has many interesting features and can be summarized as follows:

- Filtering based on nodes (concepts) and edges (relationships)
- searching
- Results may be viewed in various graphical layout
- customization of node and edge (arc) labels is possible
- customizing node and arc tooltips
- zooming
- forcibly staying node in the screen (can cause nodes to overlap)
- back and forward button to navigate history
- Nodes can be expanded or collapsed to show or hide children
- Colors of nodes, edges are customizable
- Visualization can be displayed as a widget on a web page with a fixed ontology
- Graph can be exported as a image data or a xml file
- Source code is released in SourceForge[9] which offers customization and extension based on requirement

6.1.2 Jambalaya

Protégé[10] is a very popular open source knowledge-base framework and ontology editor. It is java based, extensible and provides a plug-and-play environment which

[8] http://keg.cs.uvic.ca/ncbo/flexviz/FlexoViz.html# (Accessed 14th July, 2011)

[9] http://sourceforge.net/projects/flexviz/ (Accessed 14th July, 2011)

[10] http://protege.stanford.edu/ (Accessed 14th July, 2011)

makes it very flexible for application development. Jambalaya is one of the many Protégé plug ins for visualization.

Simple Hierarchical Multi-Perspective (SHriMP) 2 Dimensional (2D) visualization techniques written in java is a domain-independent visualization technique designed to enhance how people browse, explore and interact with complex information.

Jambalaya is developed by Chisel group at University of Victoria [11] as an extension of the SHriMP graph visualization toolkit reconfigured as a plug-in for Protégé with additional features to increase understanding of RDF/OWL projects (Figure 15). [39]

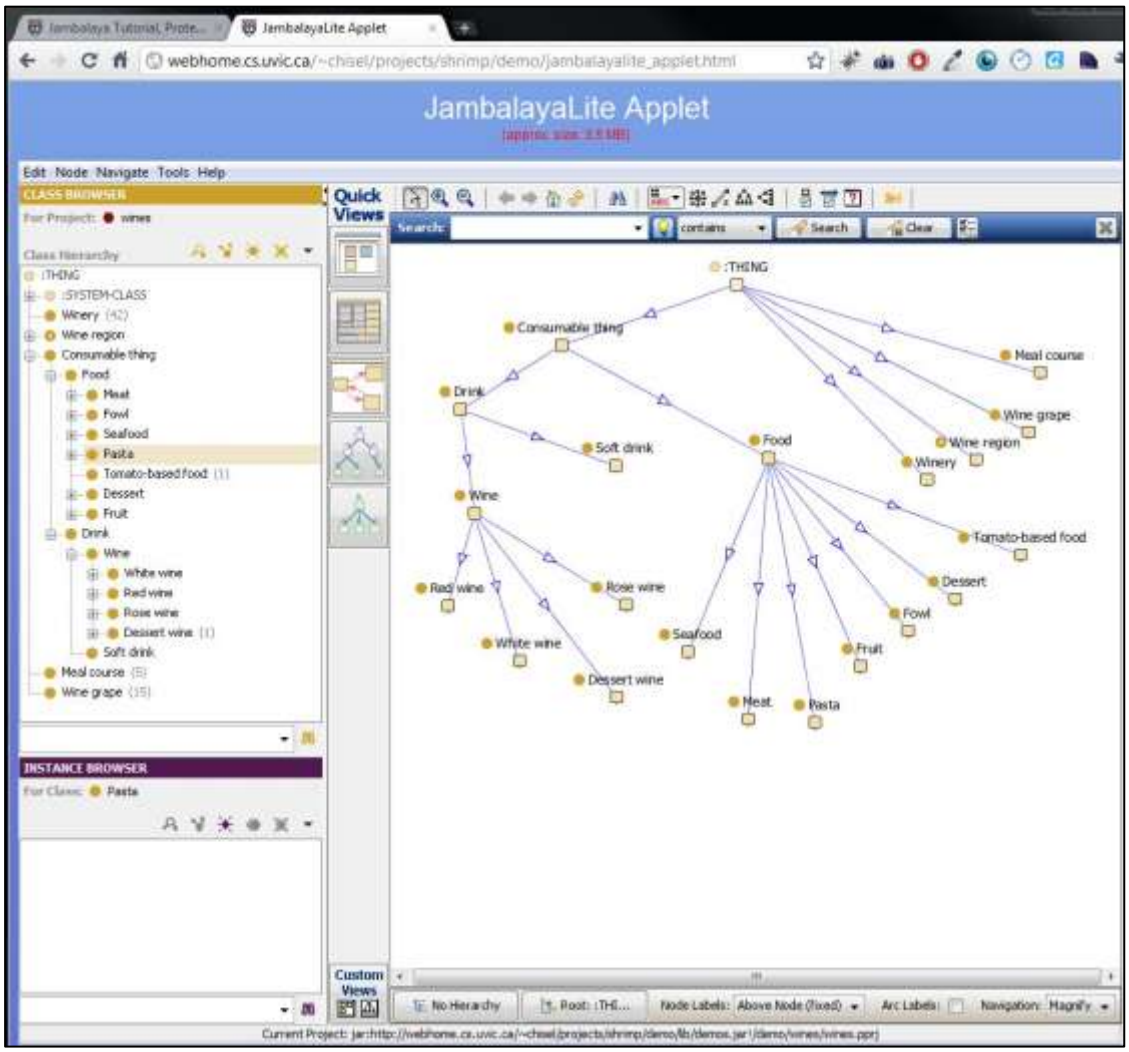

Figure 15: Jambalaya plug-in applet demo[12]

Our second choice Jambalaya has following features:

---

[11] http://www.thechiselgroup.org/jambalaya (Accessed 11th June, 2011)

[12] http://webhome.cs.uvic.ca/~chisel/projects/shrimp/demo/jambalayalite_applet.html (Accessed 11th June, 2011)

- Classes and instances are represented as nodes in a graph
- Different types may be represented using different colors
- Directed edges (arcs) are used to represent relationships between concepts and instances.
- Jambalaya offers a Zoomable User Interface (ZUI)
- It has nested changeable hierarchy
- “Hyper linking” between concepts
- It offers Magnify, fisheye, select, zoom in, zoom out
- Back, forward, home and refresh buttons
- Advanced keyword search
- Various layouts like, radial, spring, vertical and horizontal tree
- Filtering / unfilter nodes and arcs
- Customization of node color and tooltip by any attribute
- Expand/collapse children
- Can be embed in a web page as java applet
- Source code is released in SourceForge[13] which offers customization and extension based on requirement

6.1.3 jOWL TouchGraph visualization (Experimental)

TouchGraph[14] is a software developed, to visualize graphs which supports different types of relationships. David Decraene tried to visualize OWL data such as TouchGraph but based on javascript. He has made a challenging attempt to rely purely on Document Object Model (DOM) –HTML manipulations and did not use Scalable Vector Graphic (SVG) elements or flash, which are used to draw/embed more advanced graphics in HTML pages. His attempt seems to be promising and attractive. Graph works in a way that nodes repel each other, yet bonds between them keep them together. He had assumed that without bonds two nodes would keep drifting away from each other until they leave the screen. [30]

If nodes have children they can be expanded by double clicking them. Children are added in a batch of 8 at a time. Clicking node multiple times reveal more children. Demo of wine ontology can be seen in Figure 16[15]:

[13] http://sourceforge.net/projects/chiselgroup/files/Jambalaya/ (Accessed 14th July, 2011)

[14] http://www.touchgraph.com (Accessed 14th July, 2011)

[15] http://jowl.ontologyonline.org/TouchGraph.html (Accessed 14th July, 2011)

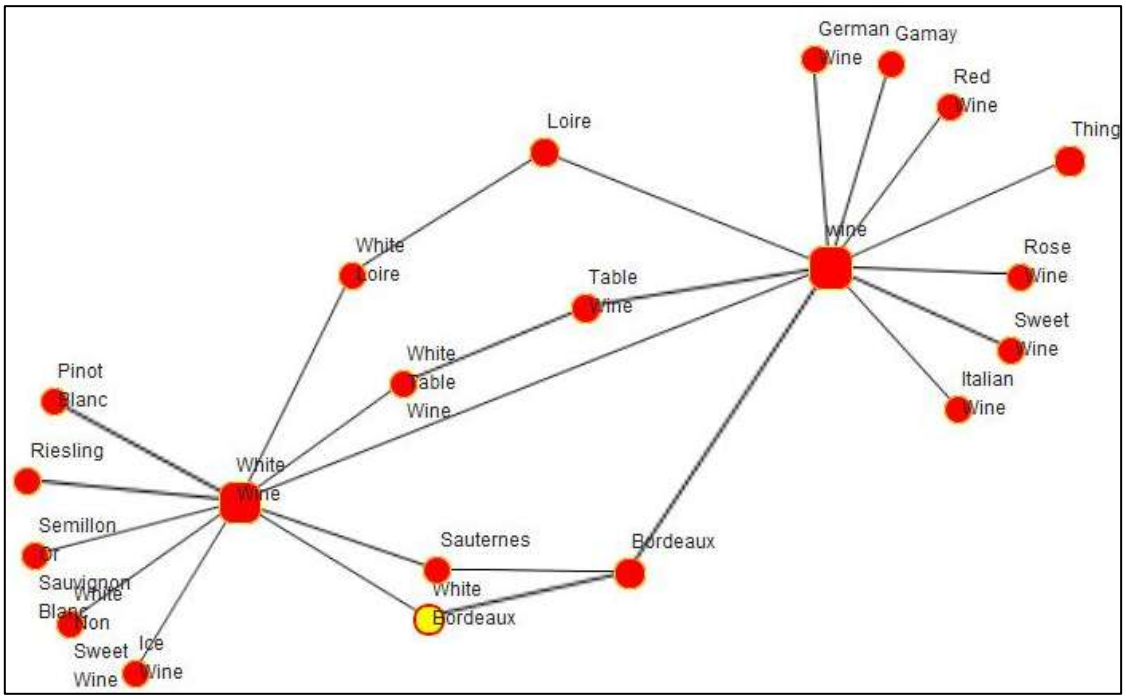

Figure 16: jOWL TouchGraph visualization (Experimental) demo [16]

6.1.4 Plone ontology

Replacement for the existing keyword mechanism in Plone is Plone ontology. It maintains a relationally structured vocabulary of keywords for content classification, searching and navigation. It uses GraphViz[17] open source visualization software for visualization. Example of Plone ontology can be shown in Figure 17:

Plone ontology has the following features[18]:

- It classifies content with keywords from an expandable ontology
- Related content is displayed in a portlet, even if not classified with the same keyword
- Within the Plone CMS tools it is possible to collaborate, build, extend and manage ontologies
- Users may propose terms and relations to other terms
- Relation properties may be specified
- Import / export of OWL ontology
- For classification or keyword adding javascript is available

[16] http://jowl.ontologyonline.org/TouchGraph.html (Accessed 11th June, 2011)

[17] http://www.graphviz.org/ (Accessed 11th June, 2011)

[18] http://plone.org/products/ploneontology/ (Accessed 11th June, 2011)

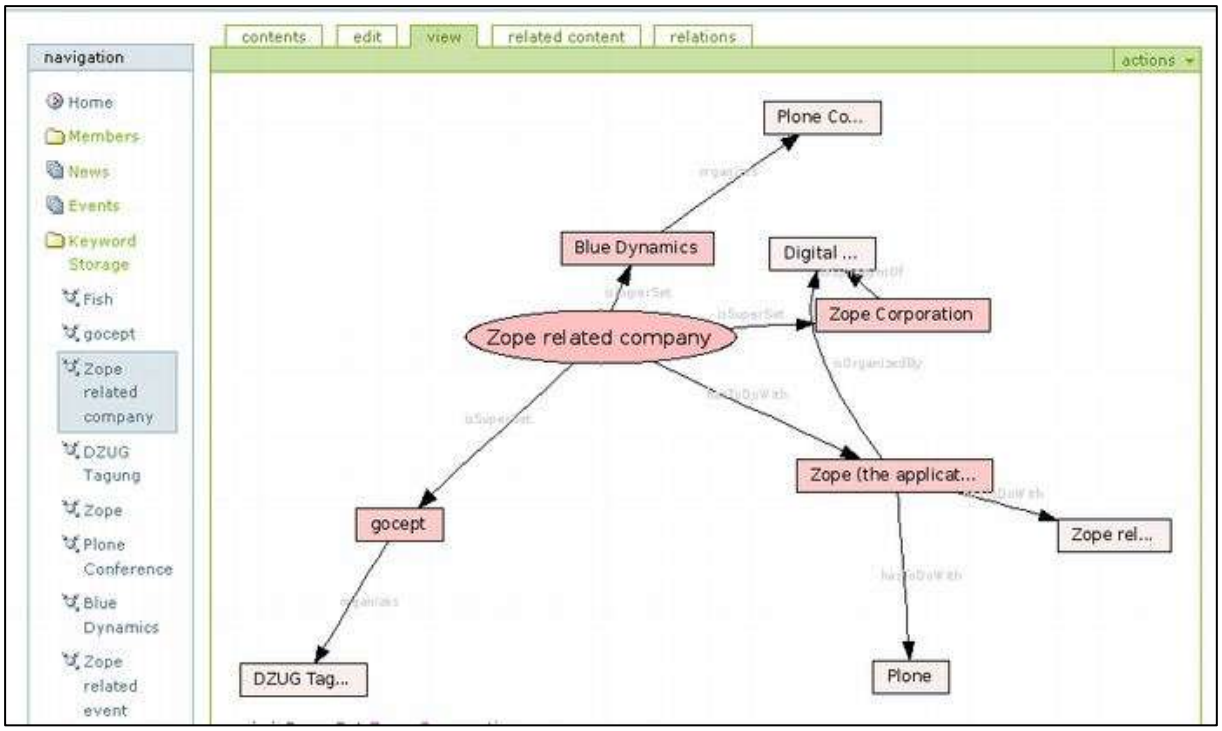

Figure 17: Plone ontology demo [19]

6.2 Evaluation of visualization methods

We evaluated the demos available on the websites of each of the visualization methods. Since plone ontology does not offer an online demo, the evaluations are based on the information provided on the website[20]. In this section we will see the results of the evaluation of each component and analysis of results will be done in subsequent section.

Functional criterion

Elements Display

1. **Classes:** FlexViz, Jambalaya, jOWL and Plone all have given score 5 as the first three shows all the classes. FlexViz does not show all the classes directly but on demand by expanding the nodes. Jambalaya applet shows all the classes directly or on demand. jOWL shows all the classes by double clicking on the node. Plone also shows all the classes in the screenshot available on the website.

2. **Instances:** FlexViz and Jambalaya have been assigned a score 5, Plone and jOWL a score 1. In FlexViz and Jambalaya instances can be viewed in a

[19] http://plone.org/products/ploneontology/ (Accessed 11th June, 2011)

[20] http://plone.org/products/ploneontology/ (Accessed 14. June 2011)

different color and filtered. Jambalaya has an instance browser which displays instance related information in it. Plone ontology does not indicate any information about instances; therefore we assign it a score of 1. jOWL does not show any way to identify or differentiate between class and instance.

3. **Taxonomy (Isa relations):** FlexViz, Jambalaya and Plone have assigned a score 5 and jOWL 1. FlexViz, Jambalaya and Plone display relationships between nodes and can be customized, while jOWL does not show any taxonomy for the relationship between nodes. It just shows that nodes are connected but taxonomy is missing.
4. **Multiple inheritance:** All four methods have given score 5. Jambalaya, jOWL and Plone show multiple inheritances clearly. FlexViz has also given a score 5 because it is a graph based method where arcs can be defined from starting node to end node and this has been tested locally also (Figure 18).

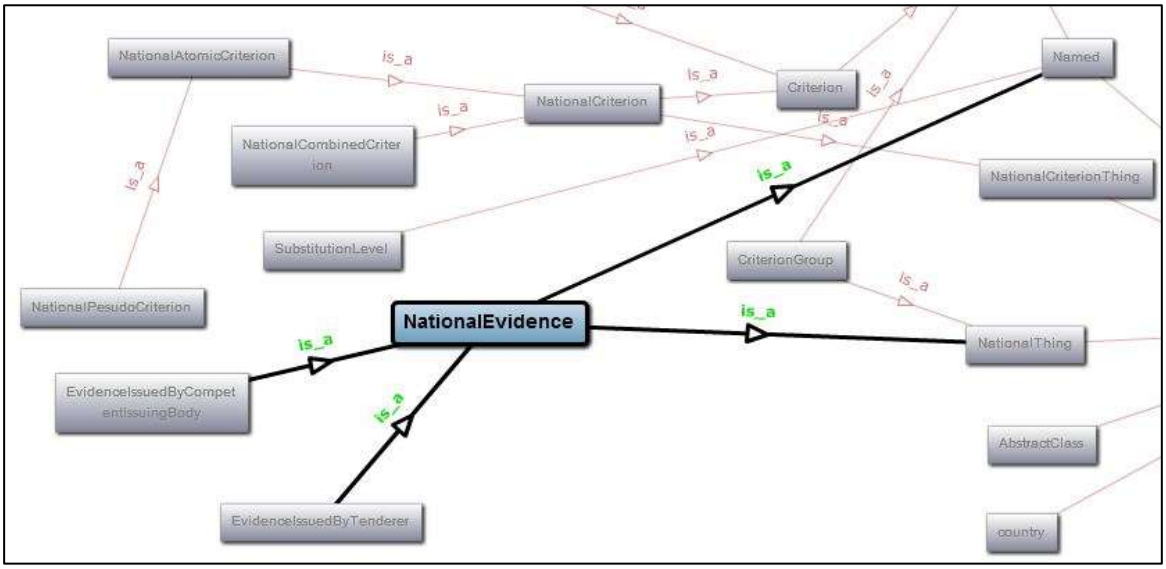

Figure 18: Multiple inheritance in FlexViz

5. **Role relations:** FlexViz, Jambalaya and Plone have given score 5 and jOWL a score 1. In FlexViz, Jambalaya and Plone Role relations are displayed and can be customized. In jOWL nodes are connected but their relationship is not evident in visualization.

6. **Properties:** FlexViz, Jambalaya and Plone have given score 5 as they display the properties on a side bar. jOWL has given score 1 as it does not display properties of a class or instance.

Navigation

1. **Overview:** FlexViz, and Jambalaya have given score 5 and jOWL a score 1. Plone has given score 2 as it does give an overview but it is not complete.

2. **Zoom:** FlexViz and Jambalaya have given score 5 as they both offer zoom function. jOWL and Plone have score 1 as there is no zoom option.

3. **Filter:** FlexViz and Jambalaya have score 5 as they both have an option of filtering unwanted elements. For example in FlexViz one can filter orphan node, instances, classes or relationship. jOWL and Plone have given a score 1 as they offer no filtering.

4. **Details-on-demand:** FlexViz and Jambalaya have given score 5. Details of a particular node or selected nodes can be displayed. jOWL and Plone have given score 1 while they do not have this possibility.

5. **Search:** FlexViz and Jambalaya have given score 5. Keyword search can be done within the displayed ontology. jOWL has given score 1 while it does not have this possibility. Plone has given a score 3 because from the features written in website it is evident that it has a keyword search option but how well it works cannot be checked.

6. **History:** FlexViz, Jambalaya have back forward button therefore given score 5. jOWL and Plone have given a score 1.

7. **Extract:** FlexViz and Jambalaya have given a score 2 as information is displayed upon extraction but it is not complete while jOWL and Plone have score 1

8. **Editing:** FlexViz and Jambalaya have given a score 2 as in FlexViz editing is possible in the code file and not directly in the SWF file whereas in Jambalaya editing is possible in Protégé plug in. Plone has given a score 3 as features written in webpage claims that editing is possible with the CMS tool. jOWL has score 1.

9. **Query customization:** FlexViz, jOWL and Plone have given score 1 and Jambalaya a score 2 as queries can be customized in Protégé plug in, not directly in applet.

Non-functional criterion

Software

1. **Web-based:** All four methods have given score 4 as all are web-based to allow ontologies to be displayed within the browser.

2. **Software availability:** FlexViz, Jambalaya and Plone have given score 5 as there codes are freely available. jOWL has given score 3 not 1 because it uses a java script library which can be downloaded.

3. **Future Support/Development:** The latest version of FlexViz was released in December 2010 but neither support site is very active nor a good documentation is available therefore it is given score 4. Jambalaya has given a score 2 has the last release in March 2009 and again has little support and documentation. jOWL has also given a score 2 as it was developed in January 2009 but since then no development is done. Plone has given a score 1 as the available latest release was in July 2007 with a support for Plone version 2.5; meanwhile currently Plone version 4 is available.

4. **Plone Integration:** FlexViz SWF file, Jambalaya applet can be embedded in a Plone page therefore given a score 5. Plone ontology is developed for Plone CMS therefore it has a score 5. jOWL is given score 3 as it does not claim a Plone integration but it uses java script library and java script can be run on a Plone site.

5. **Collaboration:** For collaboration FlexViz, Jambalaya and jOWL have a low score because they are like ready to display and not developed for collaboration therefore given a score 1. From the information available from Plone site it is evident that it offers collaboration with CMS tool therefore given score 3.

6. **User group management:** None of the visualization methods offers user groups' management therefore given score 1.

Performance

1. **Scalability:** FlexViz is developed by Chisel group to visualize biomedical ontologies which are relatively very large. On the demo site we have tested cell ontology which has more than 2000 nodes therefore given a score 3. Katifori [1] in his survey grouped Jambalaya within 1000 nodes group therefore has given score 2. jOWL has been tested on wine ontology which has above 200 nodes hence given a score 2 as well. Plone does not provide any such information hence given score 1.

2. **Loading time:** Loading time of FlexViz and jOWL is very fast. It loads almost immediately and therefore given score 5. Jambalaya has two applet version [21] one full version with a size of 14.5 MB. We have tried to load it several times but it could not be loaded at all. The lighter version of Jambalaya applet is 3.5 MB large and took an average time of 20.05 seconds[22] and therefore given a score 1. It is difficult to calculate loading time for Plone as there is no demo available, hence given a score 1.

---

[21] http://webhome.cs.uvic.ca/~chisel/projects/shrimp/demo/applets.html (Accessed 15. June 2011)

[22] Stopwatch is used to measure the loading time.

3. **Reaction time:** Reaction time of FlexViz, Jambalaya and jOWL is very short therefore given a score 5. Plone ontology reaction time cannot be calculated therefore given score 1.

Complete overview of the evaluation scores can be seen in Table 3Table 3: Evaluation results. We can see from the results table that scores of FlexViz and Jambalaya are higher than the other two. They both have almost equal scores in functional criterions but FlexViz has higher scores in non-functional criterions. For web-based visualization method performance and future support are very important. Despite being verry efficient Jambalaya fails to perform optimally therefore we have chosen FlexViz to test and implement.

Criteria			FlexViz	Jambalaya	jOWL	Plone
Functional	Elements Display	Classes	5	5	5	5
		Instances	5	5	1	1
		Taxonomy (Isa relations)	5	5	1	5
		Multiple inheritance	5	5	5	5
		Role relations	5	5	1	5
		Properties	5	5	1	5
		Total group weight	**30**	**30**	**14**	**26**
	Navigation	Overview	5	5	1	2
		Zoom	5	5	1	1
		Filter	5	5	1	1
		Details-on-demand	5	5	1	1
		Search	5	5	1	3
		History	5	5	1	1
		Extract	2	2	1	1
		Editing	2	2	1	3
		Query customization	1	2	1	1
		Total group weight	**35**	**36**	**9**	**14**
Non-functional	Software	Web based	5	5	5	5
		Software availability	5	5	3	5
		Future Support/Development	4	2	2	3
		Plone Integration	5	5	3	5
		Collaboration	1	1	1	3
		User group management	1	1	1	1
		Total group weight	**21**	**19**	**15**	**22**
	Performance	Scalability	3	2	2	1
		Loading time	5	1	5	1
		Reaction time	5	5	5	1
		Total group weight	**13**	**8**	**12**	**3**
Total weight			**99**	**93**	**50**	**65**

Table 3: Evaluation results

6.3 Testing

Figure 19 shows the steps taken for the testing of FlexViz package.

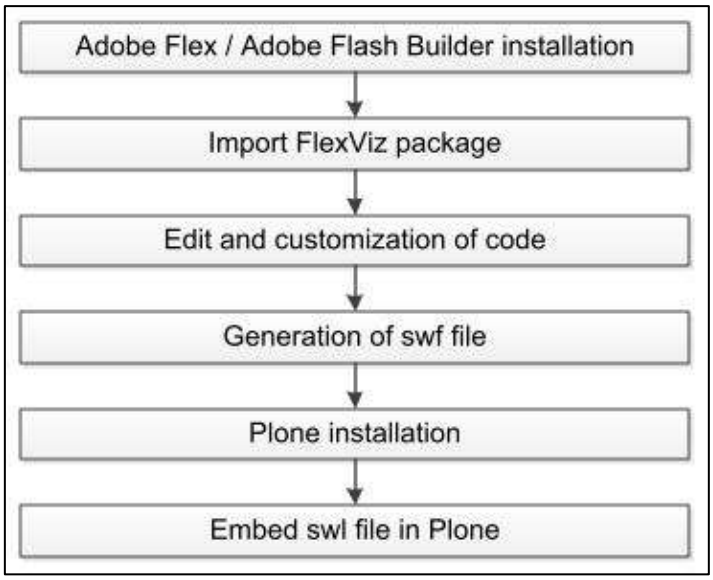

Figure 19: Testing the implementation

1. Adobe Flex installation

As we have discussed in Section 6.1.1 that FlexViz is written in Adobe Flex. Therefore, foremost step of testing is installation of Adobe Flex.

Adobe Flex is a software development kit (SDK) for the development of cross platform internet application based on Adobe Flash platform. Flex applications can be developed using free Flex SDK or Adobe® Flash® Builder™ software. For our testing we have used Adobe Flash Builder 4.5 Premium 60 day's trial version, which can be downloaded from the website of Adobe[23].

2. Import FlexViz package

Next step after installing Adobe Flash Builder locally is downloading FlexViz source code from sourceforge[24]. For our testing we have used FlexViz version 2.3.0 released on 22nd December 2010. After downloading FlexViz source code, it is imported in Adobe Flash Builder.

FlexViz source code contains the following 5 projects (Figure 20):

- Ca.uvic.cs.chisel.flexviz_flex4: the main library project
- Ca.uvic.cs.chisel.flexviz.layouts: the optional library project which contains the layout algorithms
- Ca.uvic.cs.chisel.flexviz.test_flex4: the flex project which contains the sample applications
- Flex.util_flex4: a flex library project containing useful basic static functions

---

[23] https://www.adobe.com/cfusion/tdrc/index.cfm?product=flash_builder (Accessed 23.06.2011)

[24] http://sourceforge.net/projects/flexviz/files/FlexViz/ (Accessed 24.06.2011)

- Flex.utils.ui_flex4: a utility flex library project that contains useful user interface components.

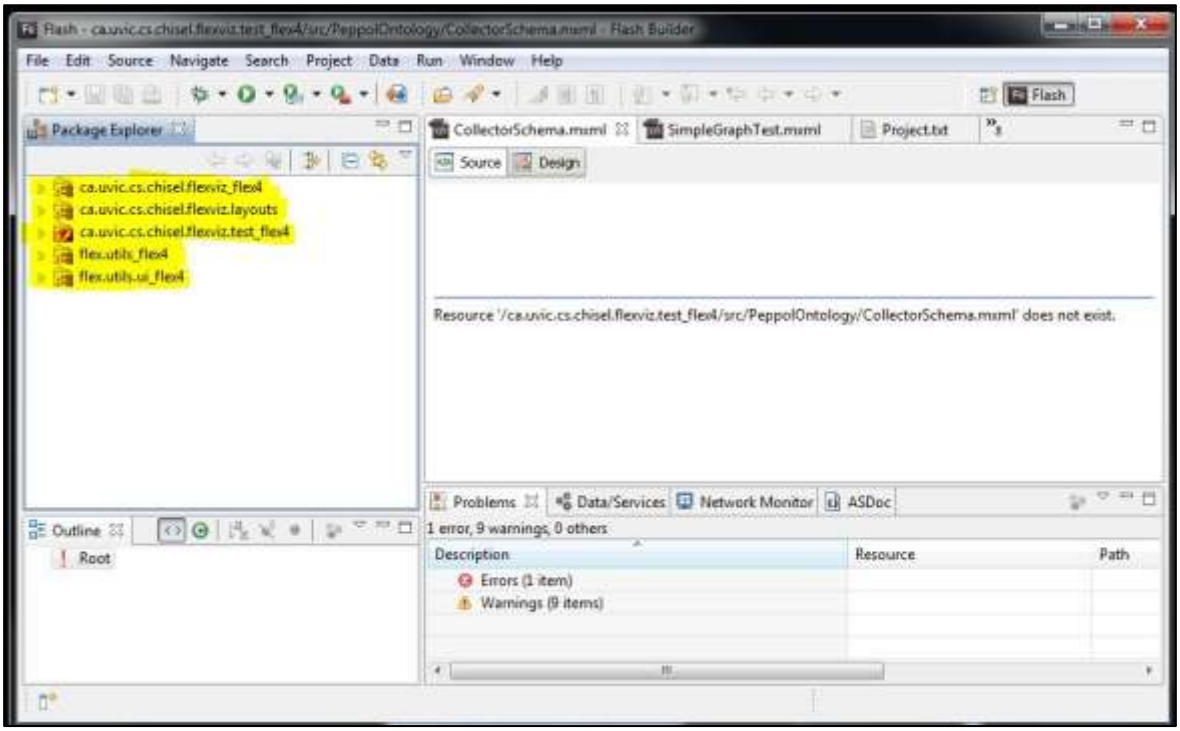

Figure 20: Adobe Flash Builder 4.5 overview

3. Edit and customization of code

A Flex-Application can be developed with the help of Magic eXtensible Markup Language (MXML) which is a declarative XML-based language and object-oriented ActionScript programming language. MXML is used to describe user interface layout and behaviours while ActionScript is used to create client logic.

FlexViz is a generic graph library project and does not know anything about ontologies. It only understands graphs: nodes, arcs and layouts.

The main classes of FlexViz project are:

- FlexGraph component, which provides the basic canvas
- ExtendedFlexGraph, which provides the canvas plus toolbar, searchbar, and right hand pane which contains the node/arc filter panels

The graph contains a model (IGraphModel/DefaultGraphModel) object which stores all the graph data elements. Each data element is then mapped to a visual UIComponent that is of the type specified by the nodeRenderer and properties. It uses the ca.uvic.cs.chisel.flexviz.layouts library project to run the layouts.

The graph uses the ca.uvic.cs.chisel.flexviz.test project to run the sample applications that use the FlexGraph and ExtendedFlexGraph components.

First step of editing begins with listing the different types of nodes and arcs required for the respective ontology and define them in SimpleGraphModel.as in ca.uvic.cs.chisel.flexviz.test_felx4.simplemodel. Node types like class, interface etc will be defined in nodeType and arc types like ***is_a, part_of*** etc. can be defined in arcType.

In the next step list of all nodes and arcs are defined in the mxml file (Figure 21)

```
var thing:DefaultGraphNode = model.addSimpleNode("Thing");
var nationalEvidence:DefaultGraphNode = model.addSimpleNode("NationalEvidence");
var nationalAtomicCriterion:DefaultGraphNode = model.addSimpleNode("NationalAtomicCriterion");
var tendererStructureSubject:DefaultGraphNode = model.addSimpleNode("TendererStructureSubject");
var instantiableClass:DefaultGraphNode = model.addSimpleNode("InstantiableClass");
var named:DefaultGraphNode = model.addSimpleNode("Named");
var euCriterion:DefaultGraphNode = model.addSimpleNode("EUCriterion");
var nationalCriterion:DefaultGraphNode = model.addSimpleNode("NationalCriterion");
var criterionCollector:DefaultGraphNode = model.addSimpleNode("criterionCollector");
var evidenceCollector:DefaultGraphNode = model.addSimpleNode("evidenceCollector");

// is-a

model.addSimpleArc(nationalEvidence, thing);
model.addSimpleArc(nationalAtomicCriterion, thing);
model.addSimpleArc(tendererStructureSubject, thing);
model.addSimpleArc(instantiableClass, thing);
model.addSimpleArc(named, thing);
model.addSimpleArc(euCriterion, thing);
model.addSimpleArc(nationalCriterion, thing);
model.addSimpleArc(criterionCollector, named);
model.addSimpleArc(evidenceCollector, named);
```

Figure 21: Graph editing for ontology mapping

Each node is defined as a variable with the name of the node (Figure 21), thereafter each arc is defined from the originating node to the end node. For example in Figure 20 first arc is ***is_a*** relationship between nationaEvidence and thing, directing ***is_a*** relationship from nationalEvidence to thing.

In the same manner all the logical parts of VCD ontology are mapped and saved in different mxml files.

4. Generation of SWF file

Once the nodes and arcs are defined SWF files can be generated by running the mxml application. Six SWF files are generated for parts of VCD ontologies which are ready to use or embed in a webpage.

5. Plone installation

For our testing we have used Plone 4 CMS which can be downloaded from the Plone site for different platforms[25]. In our case we have installed it for Windows by following the installation quick guide available at the Plone site[26].

[25] http://plone.org/products/plone (Accessed 10th July 2011)

Once Plone CMS is installed different pages are created to display the SWF files generated for parts VCD ontologies.

6. Embed SWF file in Plone

After creating pages for each logical part of VCD ontology (Figure 22) all the SWF files can be uploaded and linked to the respective pages (Figure 23).

Figure 22: Plone Site local test

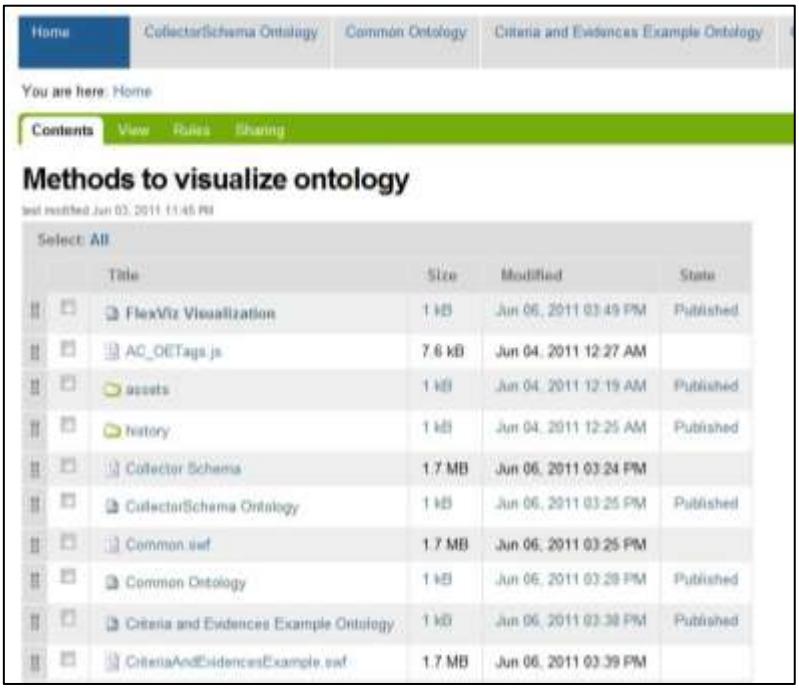

Figure 23: Uploading SWF files in Plone

Once all the SWF files are uploaded and linked with the respective ontology page, the visualization of the respective ontology can be seen (Figure 24).

---

[26] http://plone.org/documentation/manual/installing-plone/installing-on-windows (Accessed 10th July 2011)

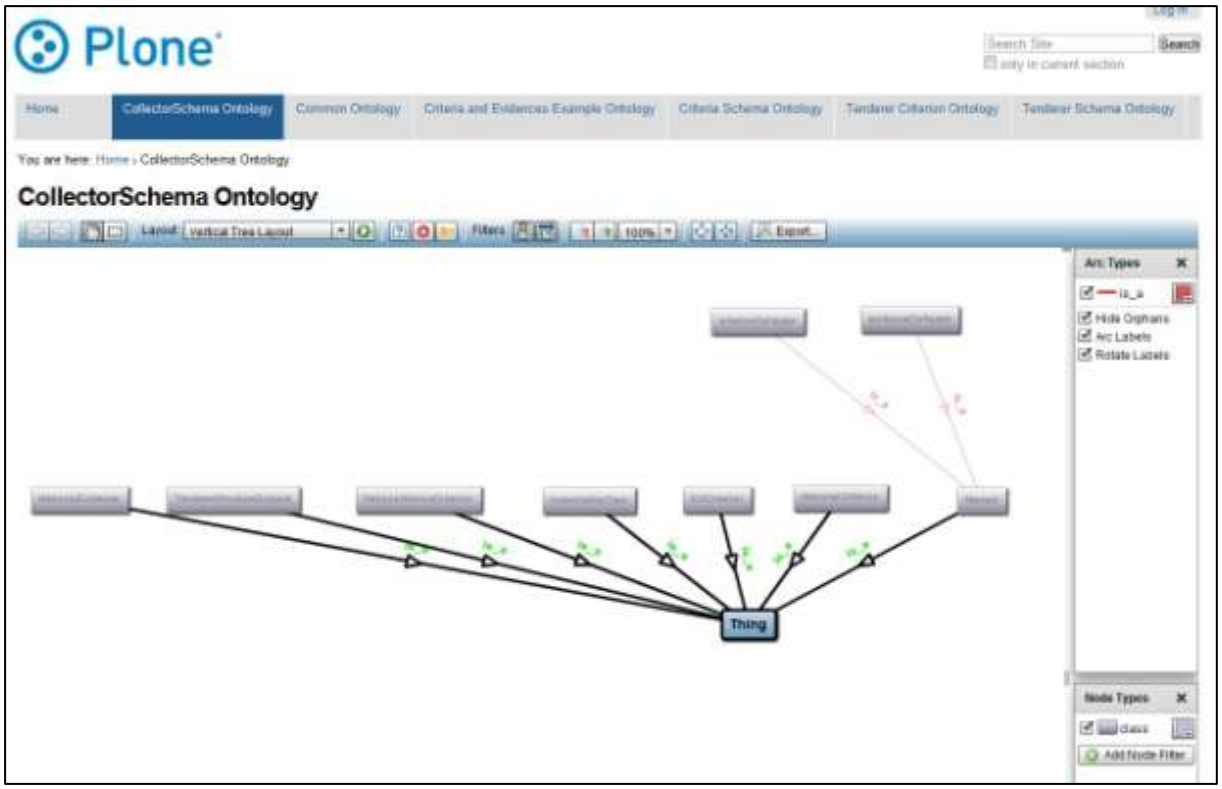

Figure 24: Visualization of CollectorSchema ontology locally in Plone

6.4 Conclusion

In this chapter we have identified four promising visualization methods based on four different technologies. Those methods are then evaluated and the highest rated method i.e. FlexViz is deployed by installing Plone CMS and integrating it within Plone. In the next chapter, we will discuss the analysis of testing and provide recommendations.

7 Review, critical analysis & recommendations

This chapter provides a review of the evaluation and testing done by summarizing the outcomes and related issues. After the critical analysis, the limitations of this study are discussed and the chapter is concluded with recommendations.

The goal of this thesis is to identify promising candidates from a pool of available web-based visualization methods, evaluate them and choose the highest scoring method. Thereafter, the task is to integrate the chosen method with Plone CMS and test parts of the VCD ontology. Finally, we analyzed the usefulness of the tool from the perspective of associated stakeholders, such as economic operators, contracting authorities and service providers.

FlexViz was found to be the best choice for the given requirements. The outcome of the testing in one statement can be written as following:

“FlexViz offers a high-quality end product, albeit with a poor development process”

Once the SWF file is generated and embedded, it offers a very wide range of display options and has many interactive ways to navigate the ontology. It’s a great end product, but with its own advantages and disadvantages. The advantages include a user friendly appearance, display capability and navigation options. The main disadvantage is the limited options for editing the ontology. To make a small change, one has to go through the entire process from mxml to SWF to embedding in the Plone site. We believe this is not a serious drawback if the ontology is well-defined and unlikely to undergo regular changes.

For economic operators and contracting authorities, FlexViz can be a very useful tool because of its wide display and navigation options. Also, its short loading time makes this tool more attractive as compared to the other available web-based visualization tools, such as Jambalaya, which has a much longer loading time. As we have seen in Chapter 4, economic operators do not edit or develop ontology. Therefore they are not interested in the process running behind the visualization. In different scenarios, an economic operator can be a contracting authority and a contracting authority can be an economic operator. Still they both have an interest in analyzing the complete ontology, or a part of it, and for that purpose, this tool is fairly good.

For service providers, FlexViz is not a useful tool. During the development process of ontology, a developer has to edit and test several times, and the offerings of this tool in regard to those functions are not attractive at all. To make a small change, the developer must go through the entire process from mxml to a SWF file, which increases the development difficulty. Moreover, the development process is dependent on the Adobe Flex framework, which makes it difficult to collaborate in real time with other collaborators in distant locations.

This tool is not recommended during the development process, but once the ontology is developed, FlexViz can be used to analyze it.

Another significant drawback of the FlexViz application is its lack of ability to read owl/rdf files, which creates a wide gap between ontology development and

visualization. Classes and their relationships must be mapped manually in the mxml file. This can be a very severe limitation when the ontology is very large and complex. Once an SWF file is generated after the compilation of the mxml file, the SWF file cannot be extended, edited or changed. One has to start from the mxml file again, which is then used to generate an SWF file, which is then embedded in Plone. From the creation of the mxml file to the embedding of SWF file, this becomes a long and difficult process, which also requires basic programming knowledge.

User-friendliness and interactivity makes FlexViz a tool with great potential. Since the source code is available, it offers developers the opportunity to develop it according to their requirements. The author has the following two recommendations, which can enable developers to exploit the potential of this tool to a greater extent.

1. A module in FlexViz can be developed which will enable users to directly upload the OWL or RDF ontologies and visualize them.
2. Alternatively, encourage the development of a plug-in or interface which can convert OWL or RDF files to mxml files. Such a plug-in could reduce the effort of the manual mapping of classes, and ontologies can be analyzed quickly.

8 Conclusion

In today's information age, the explosion of information raises new challenges to the management, organization and structuring of information so that all users understand it in the same way. This need for a common understanding is satisfied by ontology. Ontologies are represented through ontology representation languages, which are machine readable and hard for humans to understand without visualization tools. Collaboration and development over the Web created a need for web-based visualization methods, which is the focus of this study.

In the beginning of this work, we raised three research questions. Here are the answers to those questions:

1. What are the requirements for ontology visualization?

Answer: After researching the literature, we found that visualization tools should display the elements of ontology and offer a means to navigate within the ontology. Moreover, for web-based tool have a few additional advantages, such as software independence, platform independence, ease of updating and bugs fixing, etc. This was discussed in Chapters 2 and 3.

2. How is the comparative analysis and evaluation carried out?

Answer: The answer to this question is found in Chapter 5, which is by designing an evaluation matrix, the basis of which is Sommerville's [34] division of functional and non-functional criteria. Further functional criteria are divided into two subgroups: elements display (adapted from katifori [20]) and navigation (adapted from schneidermann [35]). Non-functional groups are divided into subgroups of software issues and performance, which play a vital role in web-based methods.

3. How can a chosen method be tested?

Answer: A chosen method is tested by integrating it within Plone CMS, using parts of the e-Procurement VCD ontology.

The outcome of this thesis in a sentence is:

There exists no web-based visualization tool which suits the requirements of all user groups or stakeholders; every method is useful for one user group or another.

The recommended tool, FlexViz, is a very interactive tool, but has a complicated process of converting ontology into an SWF file.

Jambalaya is also a very interactive tool, which performed poorly in loading and reaction time. However, Jambalaya can be useful for developers who edit and test ontologies. If, in the future, the loading time of the Jamabalya applet can be reduced, then it can be a useful web-based applet that can fulfill the requirements of vast user groups.

Experimental jOWL TouchGraph is a promising tool. It is in an initial development stage, but it offers the possibility of embedding emerging technologies, such as Ajax.

Plone ontology also offers a unique and complete web solution. It allows the importing/exporting of OWL files, which makes it suitable for use in collaboration over the Web However, it has not been undergoing development for several years. If developed further, it could also meet the requirements of many user groups.

After evaluating four visualization methods and testing FlexViz with parts of the e-Procurement VCD ontology, we found that most of the tools are still in the development phases. The interest groups, requirements and applications of ontologies are so vast that it is difficult to bring them under one umbrella. There is no “one method for all.” This gives researchers many unanswered questions to investigate.

9 Bibliography

[1] Akrivi Katifori, Constantin Halatsis, George Lepouras, Costas Vassilakis, and Eugenia Giannopoulou, "Ontology visualization methods - a survey.," , 2007, pp. Bd. 39. Nr. 4. S. 10:1 - 10:25.

[2] Iikka Niskanen, "An interactive ontology visualization approach for the domain of networked home environments," 2007.

[3] Nicola Guarino, Daniel Oberle, and Steffen Staab, *What Is an Ontology? in "Handbook on Ontologies"*, 2nd ed., Steffen Staab and Rudi Studer, Eds.: Springer, 2009.

[4] Ned Hall. (2010, January) Stanford Encclopedia of Philosophy Fall 2010 Edition. [Online]. http://plato.stanford.edu/archives/fall2010/entries/lewis-metaphysics/

[5] Diana Marcela Sanchez, Jose Maria Cavero, and Esperanza Marcos Martinez, *"The road toward ontologies" in ONTOLOGIES : A handbookof priciples, concepts and applications in information*. London: Springer pp. 3-20, 2006.

[6] Maria A. Wimmer, "Ontology for an e-participation virtual resource centre," in *ICEGOV '07 Proceedings of the 1st international conference on Theory and practice of electronic governance*, 2007, pp. 89-98.

[7] T., R. Gruber, "A Translation Approach to Portable Ontology Specifications," Knowledge Systems Laboratory Technical Report KSL 92-71, 1993.

[8] C. A. Nisha. (2011, June) Master of Technology Project. [Online]. www.seminarprojects.com/Thread-ontology-visualization-full-report

[9] Santtu Toivonen. (2011, June) www.cs.uta-fi. [Online]. www.cs.uta.fi/sat/lectures/lecture-14-02/sat-lecture-14-02.ppt

[10] Natalya F. Noy and Deborah L McGuinness, "Ontology Development 101: A guide to Creating Your First Ontology," 2001.

[11] Nocola Guarino, "Formal Ontology and Information Systems," in *FOIS'98*, Trento, Itlay, 1998, pp. 3-15.

[12] Oscar Corcho and Asuncion Gomez-Perez, "A Roadmap to ontology specification languages," in *EKAW '00 proceedings of the 12th European workshop on knowledge acquisition, modeling and management*, London, 2000, pp. 80-96.

[13] Nordman Kore, "Standardization of Ontologies," 2009.

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

[15] Vijay Raghavan. (2011, July) Laboratory for Internet Computing. [Online]. http://lincweb.cacs.louisiana.edu:90/group-seminars/seminarspring2008/ontology-and-semantic-web-series/understanding-ontology-and-ontology-languages-for

[16] Grigoris Antoniou and Frank van harmelen, *A Semantic Web Primer, second edition*, 2nd ed.: The MIT Press, 2008.

[17] B. McBride, "The Resource Description Framework(RDF) and its Vocabulary Description Language RDFS," in *The Handbook on Ontologies in Information Systems*.: Springer Verlag, 2003, pp. 51-66.

[18] Grigoris Antoniou and Frank van Harmelen, "Web Ontology Language: OWL," in *The Handbook on Ontologies in Information Systems*.: Springer Verlag, 2003,

pp. 67-92.

[19] Deborah L. McGuinness and Frank van harmelen. (2009) www.w3c.org. [Online]. http://www.w3.org/TR/owl-features/

[20] Johann Rath Bergh, "Ontology comprehension," University of Stellenbosch, Master Thesis 2010.

[21] Ioannis Papadakis and Machalis Stefanidakis, "Visualizing Ontologies on the Web," in *Studies in Computational Intelligence, Volume 142*.: Springer-Verlag, 2008, pp. 303-312.

[22] Andreas Rosendahl, "Visualization of knowledge in the eParticipation ontology,".

[23] V Geroimenko and C (Eds) Chen, *Visualizing the Semantic Web, XML-based Internet and Information Visualization*.: Springer, 2004.

[24] Nadia Catenazzi, Lorenzo Sommaruga, and Riccardo Mazza, "User-friendly ontology editing and visualization tools: the OWLeasyViz approach," in *13th International Conference Information Visualisation*, 2009.

[25] A. Katifori, E. Torou, C. Vassilakis, G. Lepouras, and C. Halatsis, "Selected results of a comparative study of four ontology visualization methods for information retrieval tasks," in *Second International Conference on Research Challenges in Information Science, 2008. RCIS 2008.*, Marrakech, 2008, pp. 133 - 140.

[26] (2011, May) Chisel, computer human interaction and software engineering lab. [Online]. http://www.thechiselgroup.org/jambalaya

[27] Margaret-anne Storey, Mark Mausen John Silva, Neil Ernst, Ray Fergerson, and Natasha Noy, "Jambalaya: Interactive visualization to enhance ontology authoring and knowledge acquisition (2001)," 2001.

[28] H. Alani. TGVizTab. [Online]. http://users.ecs.soton.ac.uk/ha/TGVizTab/

[29] Jason Wood, Ken Brodlie, and Helen Wright, "Visualization over the World Wide Web and its application to environmental data," in *Proceedings of the 7th conference on Visualization '96*, 1996.

[30] David Decraene. (2011, June) Online Ontology Visualization. [Online]. http://ontologyonline.blogspot.com/2009/01/experimental-touchgraph-visualization.html

[31] (2011, Mar.) Peppol Pan-European public procurement online. [Online]. http://www.peppol.eu/work_in_progress/wp2-virtual-company-dossier/vcd-brochure/VCD%20System%20Brochure.pdf

[32] Ansgar Mondorf, Damiel M. Schmidt, and Maria A. Wimmer, "Ensuring sustainable operation in complex environment: The PEPPOL project and its VCD system," in *MCIS 2010 Proceedings Paper 61.*, 2010.

[33] Ansgar Mondorf and Maria A. Wimmer, "The European VCD Service: Facilitating Public Procurement through Criteria-to-Evidence Mapping," in *What Kind of Information Society? Governance, Virtuality, Surveillance, Sustainability, Resilience*.: Springer Boston, 2010, pp. 73-85.

[34] Wolfgang Groiss. (2011, May) http://www.peppol.eu. [Online]. http://www.peppol.eu/work_in_progress/wp2-virtual-company-dossier/vcd-artefacts-1/the-european-vcd-system-1/ontology-and-reasoning-specification/Ontology%20and%20Reasoning%20specification.pdf/view

[35] Ian Sommerville, *Software Engineering 7th Edition; pp 115-129*.: Pearson Education, 2004.

[36] Ben Shneiderman, "The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations," , 1996.

[37] Sergey Krivov, Richard Williams, and Ferdinando Villa, "GrOWL: A tool for visualization and editing of OWL ontologies," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 5, no. 2, pp. 54-57, June 2007.

[38] Sean M. Falconer, Chris Callendar, and Margaret-anne Storey, "FLEXVIZ: Visualizing Biomedical Ontologies on the Web," in *International Conference on Biomedical Ontology, Software Demonstration*, Buffalo, NY, 2009.

[39] Robert Lintern and Margaret-Anne Storey, "Jambalaya Express:on demand knowledge visualization," in *8th International Protégé Conference (Protege)*, 2005.