Environments, Methodologies and Languages for supporting Users in building a Chemical Ontology

A dissertation submitted to The University of Manchester for the degree of MSc Bioinformatics in the F aculty of Life Sciences

September 2005

Louisa Casely-Hayford

TABLE OF CONTENTS

Word count: 18,856

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Ontologies are widely used by different communities for several purposes. With the advent of the Semantic Web, ontologies are becoming increasingly popular amongst members of the scientific community. This is because they provide a powerful way to formally express the nature of a domain or subject area. By defining shared and common domain theories, ontologies help both people and machines to communicate concisely, which promotes knowledge reuse and integration.

During the process of building an ontology several questions arise related to the methodologies, tools and languages that should be used in the process of development. The Council for the Central Laboratory of the Research Councils (CCLRC) is developing a Data Portal to store and retrieve experimental data from across the spectrum of the sciences. Ontologies are a key part of this effort as they are used to provide a common indexing mechanism for these data. Therefore this dissertation aims to review how simple tools and techniques can be used to gather ontological information from a community as a form of consensus building.

This project is part of a wider effort by the Council for the Central Laboratory of the Research Councils CCLRC to develop a data portal to store and retrieve experimental data from across the spectrum of the sciences. This dissertation discusses how simple tools and techniques can be used to gather ontological information from a community as a form of consensus building.

After looking at a number of tools, Protégé and the web ontology language (OWL) were chosen as the best combination for building both heavy and lightweight

ontologies. Using these tools, a Topic Map of Chemistry was converted into an ontology. Due to their lack of formal semantics SKOS (Simple knowledge Organisation systems) and topic maps proved unsuitable for building heavy weight ontologies.

DECLARATION

No portion of the work referred to in the dissertation has been submitted in support of an application for another degree or qualification of this or any other university of the other institution of learning.

COPYRIGHT STATEMENT

ACKNOWLEDGEMENT

Many thanks to the head of the CCLRC e-Science Data Management group Kerstin Kleese Van Dam, for her support and guidance throughout the duration of the project. Special thanks to Mr. Shoaib Sufi, the Deputy Group Leader of the CCLRC e-Science Data Management group, for his constructive suggestions and comments. I would like to express my deepest gratitude to my supervisor, Dr. Stevens. His advice, guidance and insight helped me enormously throughout this project.

CHAPTER 1

INTRODUCTION

Ontologies are becoming increasingly popular amongst the scientific community as they can be used by people, databases and applications to share information in a domain or area of knowledge[1]. An ontology defines terms that can be used to describe an area of knowledge[2].

During the process of building an ontology several questions arise related to the methodologies, tools and languages that should be used in the process of development. This project aims to show how simple tools and techniques can be used to gather ontological information from a community as a form of consensus building.

A topic Map of chemistry will form the basis of this project and the main methodologies, languages and tools used for building ontologies will be described and evaluated. In this chapter, a brief review of ontologies and their use as an indexing system in the CCLRC DATA Portal is covered. The objectives and structure of this dissertation are then described in the last sections.

1.1 Ontologies.

The word ontology itself stems from Philosophy, where it means a systematic explanation of being[3]. To date there are several existing definitions of the word ontology. A definition by Gruber has become the most quoted explanation in literature by the ontology community[4]. He defined an ontology as "an explicit specification of a conceptualization"[5]. Ontologies are widely used for different purposes (natural language processing, e-commerce, knowledge management, intelligent information integration and the semantic web) and by different communities (knowledge engineering, bioinformatics, databases and software engineering)[6].

1.2 The Role of Ontologies in the CCLRC Data Portal

The Council for the Central Laboratory of the Research Councils (CCLRC) runs a range of large scale experimental, computing and data facilities for the UK Science community[7]. The areas of science supported by these facilities vary from Physics, Chemistry, Biology and Geology to Materials Science and Astrophysics.

To enable easier access to all the different types of data produced at these facilities the CCLRC e-Science centre are currently implementing a new data management system. The management of data is achieved using two web based service portals, the Data Portal and the DATA insertion Portal[7]. The

Data Portal functions to provide access to multidisciplinary data, whiles the Data Insertion Portal is used to upload data into the system and annotate the data[7].

Currently in order to access data scientists at CCLRC have to manually search the experimental, data, computing and analysis facilities available world wide. Consequently, the data Portal aims to provide infrastructural support to scientists through customisable, community oriented portals web portals[7].
The storage and management of scientific data generated at its facilities is an important responsibility of CCLRC, and the full value of these data resources will only be realised if they are easily searchable, accessible and reusable. Ontologies are a key component of this process as they provide a common indexing mechanism for these data across a wide range of communities.

Ontologies are critical for applications that want to search across or merge information from diverse communities[9]. This is because the ontology formally represents common knowledge and interests that people share within their community[10]. Therefore the role of the ontology will be to function as a back- bone for the community's web portal and it will be used to support the major tasks of the portal.

1.3 Objectives

The key objective of this project is to choose an editing environment, language and methodology that can easily be used by domain experts in the scientific community to build an ontology. A Topic map of Chemistry will form the basis of this task. Currently a Topic Map functions an indexing system for the CCLRC data portal in the domain of Chemistry.

For many years conventional knowledge organization tools such as controlled vocabularies and thesauri have been used to efficiently represent complex indexes such as medical dictionaries[11]. Even though they possess concepts and relationships similar to ontologies, they are far less expressive than ontologies[12]. Ontologies provide deeper semantics in the way they model domains[13], therefore the second objective is to convert this Topic Map into an Ontology in the domain of Chemistry using the key-words provided by Chemists or domain experts.

Once the Ontology has been implemented in the Data Portal it will have to be maintained and updated and scientists or domain experts require a method of editing the Ontology either in XML or OWL format. Thus the final objective of the project is to evaluate graphical methods of editing the Chemical ontology.

1.4 Structure of the Dissertation

This rest of this dissertation is organized as follows. The second chapter describes exactly what an ontology is, the types and uses of ontologies and what is involved in building an ontology. The third chapter describes the methodologies or methods, tools or development environments and languages used for building ontologies. The fourth chapter evaluates the different methodologies, tools and languages for building an ontology and describes what is involved in choosing an editor and implementation language. Chapter 5 explores other methods of building controlled vocabularies including Topic Maps, Ontology management systems and SKOS (Simple knowledge Organisation systems).

At present most of today's tools support design and development, but there is still a lack of tools that facilitate the deployment and maintenance of ontologies. Consequently the problems with deployment of ontologies will be highlighted and discussed. In Chapter 7 the conversion of the Topic Map of Chemistry into an Ontology will be described and discussed. The final chapter discusses what was achieved, what outcomes were successfully accomplished and what further developments can be made to the Chemical ontology.

CHAPTER 2

BACKGROUND AND LITERATURE REVIEW

In this chapter, the background and literature that relates to this dissertation is surveyed. The primary topics covered are, a review of the existing definitions of the word ontology, and the different types and uses of ontologies. The next main focal point is an explanation of what is involved in building an ontology and the role of ontologies in Semantic Web Portals. Finally the existing methodologies, tools and languages for building ontologies as well as ontology web standards will briefly be reviewed. By covering these subjects the task of converting the Topic map of Chemistry into an ontology will be much simpler.

2.1 The Council for the Central Laboratory of the Research Councils

(CCLRC) was founded in 1995, and is one of Europe's largest multidisciplinary research organisations supporting scientists and engineers world-wide. It operates world-class large-scale research facilities, provides advice to the government on their development and manages international research projects in support of a wide cross-section of the UK research community[7].

The Daresbury Laboratory, one of the three CCLRC laboratories, was formed in 1962 as part of the National Institute for Research in Nuclear Science

(NIRNS). Daresbury hosts a wide variety of resources and services that are important for the research community. The E-science is one of such services and can be broadly divided into four main areas including; Data Storage and Management, Scientific Computing Grid Development, and Grid Exploitation. The mission of e-science is "To spearhead the exploitation of e-Science technologies throughout CCLRC's programmes, the research communities they support and the national science and engineering base.[7]"

Within E-science, the Data Management group is focused on providing convenient access to secure and affordable medium to long term storage of scientific data. This will help to facilitate future cross-disciplinary activities and will constitute a major resource within the UK e-Science Grid. The scientific Metadata Model and the CCLRC Data Portal have been developed to support the data description and facilitate data reuse. The aim of the Data Portal is to develop a method for scientists to search these data resources, find the data they require and retrieve datasets via one interface and independent of the location of the data. Ontologies will provide a backbone for the Data portal and aid in the retrieval of information across the spectrum of science[7].

2.2 What is an Ontology?

The word ontology is derived from philosophy, where it is the most fundamental branch of metaphysics that deals with the subject of existence[5]. In the last decade many definitions of the word ontology have been proposed. In this section these main definitions will be reviewed, illustrating how these explanations have evolved with time.

One of the first definitions was given by Neeches and colleagues in 1991 who defined an ontology as follows: "An ontology as the basic terms and relations comprising the vocabulary of a domain or subject area as well as the rules for combining terms and relations to define extensions to the vocabulary". According to Neeches's definition, an ontology includes terms that are explicitly defined and knowledge can be inferred from the ontology[14].

A few years later in 1993 Gruber defined an ontology as "an explicit specification of a conceptualization". This definition by Gruber, is the one most quoted in literature by the ontology community. Ontologies are specifications of the concepts in a given field, and of the relationships between those concepts[9]. By defining shared and common domain theories, ontologies help both people and machines to communicate concisely[15]. In Gruber's definition 'Conceptualization' refers to an abstract model of some phenomena in the world by having identified the relevant concepts of those phenomena.

'Explicit' means that the type of concepts used, and the constraints on their use are explicitly defined. 'Formal' indicates that the ontology should be machine readable, which excludes natural language. 'Shared' reflects the belief that an ontology captures consensual knowledge meaning that it is not private to some individual, but accepted by a group[16].

Brost and colleagues slightly altered Gruber's definition; "Ontologies are defined as a formal specification of a shared conceptualization". Gruber and Brost's definitions have been merged and explained by Studer and colleagues as follows; "Conceptualisation refers to an abstract representation of a phenomenon in the world by having identified the relevant concepts of the world used, and the constraints on their use are explicitly defined"[17]. In 1995 Guarino and colleagues proposed to consider an ontology as "a logical theory which gives an explicit partial account of a conceptualization" where conceptualization is basically an idea of the world that a person or a group of people can have. Work done by Guarino and colleagues raised the bar as they created a logical theory which established how to build the ontology[18].

The following definition was put forward by Bernaras and colleagues in the framework of the KACTUS project in 1996: "An Ontology provides the means for explicitly describing the conceptualisation behind the knowledge represented in a knowledge base". This definition proposes extracting the

ontology from a knowledge base (KB), and this KACTUS method reflects the approach majority of authors use to build ontologies[19].

In 1997 Swartout and colleagues proposed another strategy for building ontologies which would be to reuse large ontologies like SENSUS[20] to build domain specific ontologies: "An Ontology is a hierarchically structured set of the terms for describing a domain that can be used as a skeletal foundation for a knowledge base". This definition specifies that at the same ontology can be used for building several knowledge bases (KBs). Inferencing mechanisms and merging and sharing of KBs is simplified in systems built with the same ontology[21].

Figure 2.1 Amigo. On the left is a snapshot of AmiGo a browser used for searching GO ontologies. Search results are shown on the right and include

synonyms, definitions, links to other pages and an option to view the hierarchy.

An example of an ontology widely used by biologists to search for associations of genes and gene products is GO. The Gene Ontology (GO) project is a collaborative effort to address the need for consistent descriptions of gene products in different databases. Three ontologies form the backbone of GO (biological process, cellular components, molecular functions). They describe gene products in terms of their associated biological processes, cellular components and molecular functions in a species-independent manner. Several browsers have been created for searching GO. The AmiGo browser searches by GO term and gene products (Figure 2.1), and results include the GO hierarchy for the term, definitions and synonyms for the term and external references[20].

2.3 Types of Ontologies and their Uses

There are several different types of ontologies and they can broadly classified into two main types; heavyweight and lightweight[22]. Lightweight ontologies are simple taxonomic structures of primitive or composite terms together with associated definitions. As the meanings of terms being used by the community is known in advance by all members, they possess few axioms. A database schema which formally describes records in a database would be an example of a "lightweight" ontology[22]. On the other hand heavyweight

ontologies are extensively axiomatized and therefore represent ontological commitment explicitly. The purpose of the axiomatization is to prevent terminological and conceptual uncertainty caused by unintended interpretations. Every heavyweight ontology can have a lightweight version and many domain ontologies are heavyweight because they should support heavy reasoning (e.g. for integrating database schemata, or to drive complex corporate applications) [23].

Guarino proposes that "top-level" ontologies describe all general concepts such as space, time, matter, object, event and action[24]. A domain ontology is an ontology tied to a specific domain like medicine, chemistry or automobiles[25]. Task ontologies are those ontologies that describe vocabulary related to a generic task or activity like diagnosing or crystal structure prediction. Borge and colleagues characterize reference ontologies (or more recently foundation ontologies) as rich axiomatic theories whose focus is to clarify the intended meanings of terms used in specific domains[27]. Application ontologies in contrast, provide a minimal terminological structure to fit the needs of a specific community. The main aim of an application ontology is to "fit the needs of a specific community", therefore it can be more of a lightweight ontology[26].

Since the early nineties ontologies have become an increasingly popular research topic. Initially ontologies were primarily investigated by Artificial Intelligence research communities like knowledge engineering, natural-language processing and knowledge representation[28]. Currently ontologies are becoming widespread in a range of fields.

Ontologies are used as central controlled vocabularies that are integrated into catalogues, databases, web publications and knowledge management applications. An example of a collaborative effort to generate large simple ontologies is DMOZ[20]. This Open Directory Project is the largest, most comprehensive human-edited directory of the Web, and is constructed and maintained by a vast, global community of volunteer editors[20]. Large ontologies are essential components in many online applications including search engines such as Yahoo and Lycos and e-commerce like Amazon and eBay[29]. The Unified Medical language system (UMLS) is an example of a more sophisticated heavy-weight ontology on medical terminology. To date, a greater extent of companies like Cycorp, are making portions of heavy-weight ontologies available, and this promotes re-use and sharing of knowledge[20]. Well-structured heavy-weight ontologies allow querying of complex relationships within large data sets, and facilitate automatic inference of new knowledge[29].

Ontologies are made up of four main components; these include concepts, relations, axioms and instances[30]. Concepts represent entities in the domain and can be abstract or concrete, elementary or composite, real or fictitious. In short a concept can be any thing about which something is said and therefore could be a description of a task, function, action, strategy or reasoning process[31]. Concepts are also called Frames and Classes[31]. Relations represent a type of interaction between concepts of the domain. Axioms represent sentences that constrain relations[31]. Instances represent the entities or individuals of a certain concept. However not all ontologies are made up of all four components, and in every ontological environment the level of detail and constraints on each concept differs from one system to another[31].

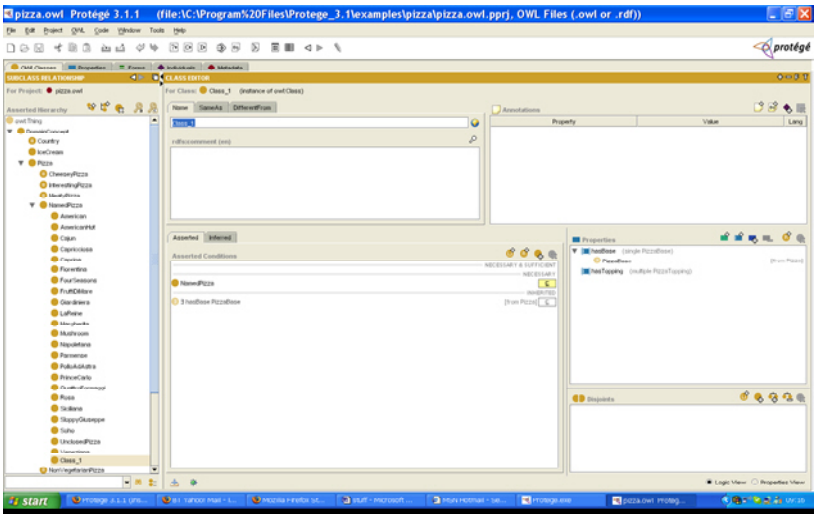

Figure 2.2 Snapshot of Protégé OWL interface. This is an example of an editor used to build ontologies. The protégé OWL interface is made up of several tabbed panes where you can switch between editing properties, classes and individuals.

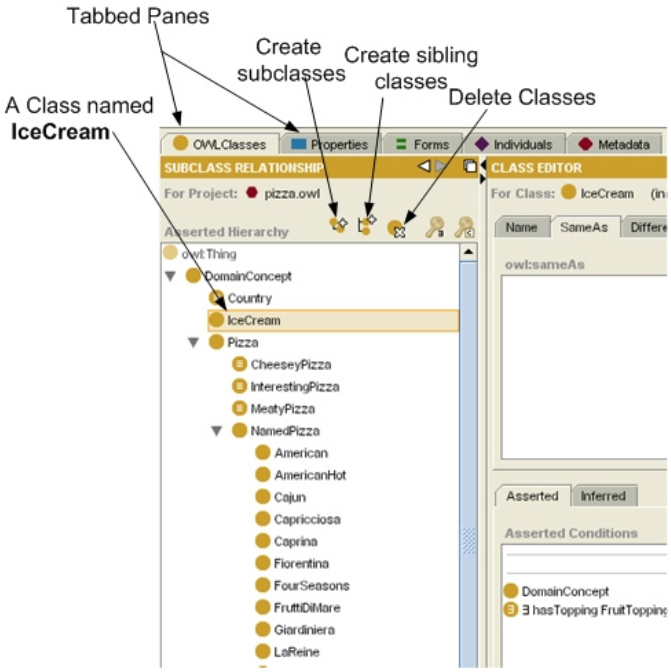

Figure2.3 Editing in Protégé OWL. Classes can be added and deleted with the click of a button

2.4 Building an Ontology

The four basic aspects to consider when building an ontology are content (the content of the ontology), the language in which it is implemented, the methodology which has been followed to develop it and the software tools used to build and edit the ontology[4]. During the process of building an ontology several questions arise related to what methodologies, tools and languages should be utilised in the process of development.

The following are a list of some basic questions to be considered when building an ontology:

1. What is the purpose of the ontology?

It is important to decide what the ontology will be used for. Will the ontology be a heavy-weight ontology or a light weight ontology? Is it going to be an application based ontology a domain ontology?

2. Which methodologies or methods can be used for building ontologies?

Numerous guidelines for building methodologies have been proposed by various groups. After deciding what type of ontology will be created, any of these strategies can be followed during the process of development.

3. Which tools give support to the ontology development process?

There are several tools to support ontology building, and example of such a tool is shown in Figure 2.2 and 2.3. When choosing a tool several issues come to light, and these have to be taken into account in the choice of development environment. Some of these issues include; ease of use, diverse user support i.e. naïve users or well experienced users , lifecycle issues such as merging or integration of ontologies and knowledge acquisition, maintenance and versioning, interoperability, collaborative support, security, extensibility and scale[20].

4. Which language(s) should be used to implement the ontology?

An ontology must be encoded in some language and if a light weight ontology is being created few issues arise. However with a heavy-weight more complex ontology, the expressive power and reasoning ability of a representation language have to be considered[30].

5. Do tools have translators for different ontology languages or formats?

The semantic web intends to create a universal medium for information exchange by giving meaning (semantics) in a manner understandable by machines, to the content of documents on the Web[32]. Therefore it is important for editors to be interoperable with current web standards like XML and RDF. Editors with translators to support multiple input and output formats promote interconnectivity between different tools.

6. What expressiveness does the ontology language have?

As ontologies become more complex and lean towards a heavy weight structure, they need to express more information, and as a result their expressive requirements grow. Very expressive ontology languages like OWL or Ontolingua[33] allow ontologists to specify first order logic constraints

between terms and more detailed relationships such as disjoint classes, inverse relationships and part-whole relationships.

7. Do the tools or language(s) have an inference engine?

In heavy-weight ontologies inference becomes very important. This is because inference engines can provide deductions based on the constructs represented in the language[30]. For example, in order to separate a group of classes they are made disjoint from one another. This guarantees that an individual which has been asserted to be a member of one of the classes in a group cannot be a member of any other classes in that group[31]. Consequently, if a language supports the notion of stating that two classes are disjoint, then a reasoning engine should be able to be built such that it enforces the constraint that the classes are disjoint [31]. Thus, an inference engine should be able to warn a user if he or she is creating an instance or subclass of two disjoint classes. When building heavy-weight ontologies, an editor with a good inference engine would lessen time wastage, as a user would be made aware of inherent errors and this would in turn lead to a reduction in cost.

Overall it can be seen that ontologies can be embodied in several ways including lists of words; taxonomies, database schema, frame languages and description logics[5]. The most critical issues involved in the choice of methodology, language and editing environment are; firstly to decide on the

type of ontology to be built (light-weight or heavy-weight), and secondly on the use of the ontology (e.g. as an indexing system in a web portal or as a simple taxonomy).

2.5 Methodologies, Languages and editing environments

There are a growing number of methodologies that specifically address the issue of the development and maintenance of ontologies[35]. The following is an overview of a series of approaches reported in literature for developing ontologies. In 1990 Lenat and Guha published general guidelines based on the Cyc development[36]. Later on in 1995, experience from developing the Enterprise Ontology[37] and the TOVE Toronto Virtual Enterprise) project ontology both in the domain of enterprise ontology, led to the proposal of a methodology which was later refined[38]. At the 12th European conference for Artificial Intelligence, Bernaras et al.[39] presented a method used to build an ontology in the domain of electrical networks as part of at the Esprit KACTUS project[19]. Around the same period the methodology METHONTOLOGY, appeared and was later extended[40]. In 1997, a new method was proposed for building ontologies based on SENSUS ontology[20]. A few years later, the On-To knowledge methodology appeared as a result of the On-To knowledge project[41].

The following is an example of a Methodology inspired by the software engineering V-process model (Figure 2.4 and 2.5).

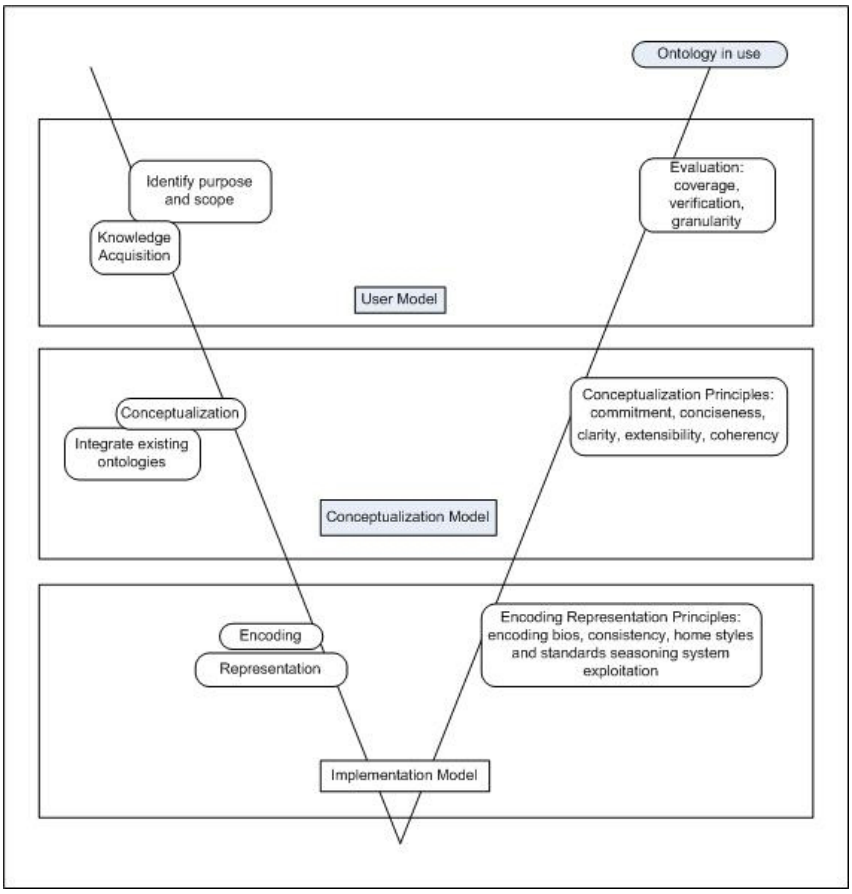

Figure 2.4 Illustration of a skeletal methodology and life cycle for building ontologies inspired by the V-process model[30].

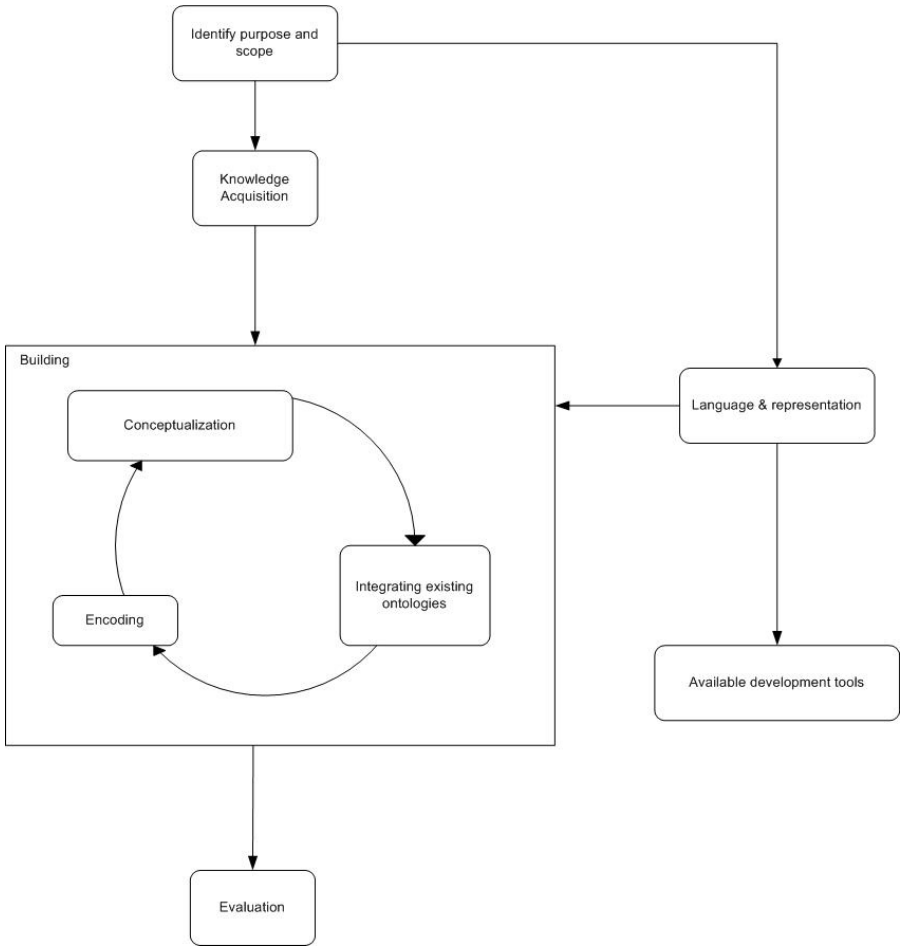

Figure 2.5 Ontology Life cycle. This flowchart illustrates the stages the V-process model goes through in its life cycle.

The V-process model goes through a life cycle and its stages are the following:

1. Identify purpose and scope: It is important that the scope and purpose of the ontology be identified in order to develop a requirements specification. A well structured requirements specification is important in the design, evaluation and re-use of the ontology[30].

2. Knowledge Acquisition: This is the process of acquiring domain knowledge from which the ontology will be built. Knowledge can be acquired from specialist biologists; database metadata; standard text books; research papers and other ontologies. For instance with the EcoCyc and Ribo Web ontologies, the bulk of knowledge represented in was collected from the research literature on E. coli metabolism and ribosomal structure respectively[30].

3. Conceptualisation: This involves identifying the key concepts that exist in the domain, their properties and the relationships that exist between them; identifying natural language terms to refer to such concepts, relations and attributes; and structuring domain knowledge into explicit conceptual models. In this stage the ontology is described using some informal terminology. Gruber suggests writing lists of the concepts to be contained within the ontology and considering the re-use of all or parts of the conceptualisations and terminologies of other ontologies[30].

4. Integrating: This entails use of an existing ontology. For example a generic ontology such as MBO gives a deeper definition of concepts of the domain. This task is made difficult as many existing ontologies

hold a level of implicit assumption due to the widespread use of domain experts in building ontologies[30].

5. Encoding: This involves the representation of the conceptualisation in some formal language, e.g. frames, object models or logic. This includes the creation of formal competency questions in terms of the terminological specification language chosen (usually first order logic)[30].

6. Documentation: Informal and formal complete definitions, assumptions and examples are vital to promote the appropriate use and re-use of an ontology. Additionally, documentation is important for a more detailed definition of the exact meaning of terms within the ontology[30].

7. Evaluation: This is the process of determining the appropriateness of an ontology for its intended application. Evaluation can be achieved by assessing the competency of the ontology to satisfy the requirements of its application, including determining the consistency, completeness and conciseness of an ontology. Conciseness implies an absence of redundancy in the definitions of an ontology and an appropriate granularity[30].

Currently ontologies are implemented in a large variety of languages. These different ontology languages provide different facilities. At the beginning of the 1990s a group of AI-based (artificial intelligence) languages were designed for execution of ontologies[4]. These more classic languages include KIF-based Ontolingua (Gruber 1993), LOOM, OCML and Frame Logic (FLogic). More recent ontology implementation languages include; XML, RDF, RDF Schema, XOL, SHOE, OIL, DAML+OIL and OWL[42]. These languages are in a constant state of evolution and are called "web- based" languages. As XML has emerged as a standard language to exchange information on the web, they have been created based on XML to implement ontologies[43]. The World Wide Web Consortium (W3C) recommends XML, XML schema, RDF (Resource Description Framework), RDF schema and OWL as standards for the Semantic Web[44]. The Semantic Web extends the ability of the World Wide Web through the use of standards, markup languages and related processing tools[44].

Ontology development or engineering tools include suites and environments that can be used to build a new ontology from scratch or by reusing existing ontologies[34]. Since the mid-nineties there has been an exponential increase in the development of technological platforms related with ontologies. Some of the older environments which are in a state of stable development include

Ontosaurus, Ontolingua and WebOnto[4]. More recent tools include; OntoEdit, OilED, WebODE, Protégé, Chimera, SWOOP and DAG-Edit[4].

2.6 The Role of Ontologies in Semantic Web (SW) Portals

Web Portals are designed to use distributed applications, different numbers and types of middleware, and hardware to provide services from a number of different sources[45]. An example would be a health portal whose users include doctors, nurses, hospital management, government, insurances, and, of course patients. Each user will have different views onto the focal theme and web portals typically provide personalized capabilities to their users. Ontoweb is an example of such a web portal[46].

In Semantic Web (SW) portals, sorting and indexing techniques vastly improve the system's data storing and retrieving capabilities[47]. However, a simple indexing system like a controlled vocabulary may not provide the community with sufficient ability to search for the content that its members require. An ontology provides term definitions of the domain of interest and can be applied in different ways to enhance functionalities of a SW portal[48]. The most common types of ontology used in SW portals are domain ontologies and application ontologies[49].

Appropriate ontology management facilities are essential for long-term usability of a web portal. To build and maintain an ontology for a web portal the following should be considered; editing, automatic classification, inference mechanisms, interoperability with current web standards and versioning[50].

An appropriate editing facility has to be provided. This can either be an ontology editor like Protégé OWL or OntoEdit or an editing facility integrated into the portal. Furthermore the system should provide means of keeping the relation between the schema and the instances consistent, in case of a change to the one or the other. For example support is needed to update instance data automatically in case a property is added to the corresponding class or concept[50].

Ontologies are like software, they will need to be maintained as they will change over time[20]. Consequently it is important to have the facility to achieve ontology versioning to allow tracking of earlier versions of the ontology by enumerating different ontology versions. Additionally authors may want to also keep track of version information for classes, properties, and individuals[49]. OWL, a W3C recommended web ontology language, provides the syntactic ability to link to a previous version of the ontology[44].

With Semantic web portals it's important to have inferencing and reasoning systems in place[50]. A reasoner can be used to check cardinality constraints and class member ship or an inference engine could interpret symmetric or transitive relationships[47].

To facilitate interoperability and information exchange with other SW portals and semantic web applications, the ontology management systems of a SW portal should support the import and export of different Semantic Web ontology languages[50]. Syntactical interoperability is the first step towards a semantic one, enabling sharing of dynamically evolving ontologies in a peer to peer fashion[50].

2.7 Topic Maps

Currently a list of Chemistry topics in the form of a Topic Map serve as an indexing system to annotate and classify data from chemistry experiments in the data portal. In this section a brief review of Topic maps will be given[72].

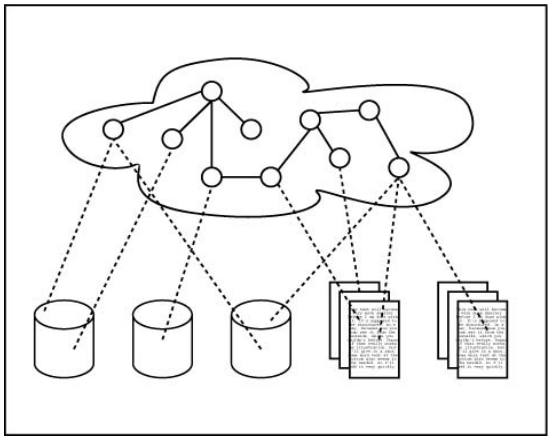

Figure 2.6 Topic Map. The topic map (the cloud at the top) describes the information in the documents (the little rectangles) and the databases (the little "cans") by linking into them using URIs (the lines). Circles represent Topics and the lines between circles indicate associations between topics.

A topic Map takes the key concepts described in a database or document and relates them together independently of what is said about them in the information being index (Figure2.6). This makes the information structure free of constraints[72]. Topic Maps are made up of topics, associations, and occurrences. Topics are at the heart of topic maps and represent the things the topics are about (circles in the Figure 2.6). For example, in a topic Map about Chemistry, topics representing "Crystal Structure", "Aspirin" and "atomic weight" could be found.

In Topic Maps, relationships between topics are modelled with associations (lines between topics)[72]. Associations are given a type, therefore for example the relationship between the "Aspirin" the element and "atomic weight" can be "possess". Whiles the relationship between "Crystal structure" and

"Aspirin" can be "predicted". With associations, each topic involved in the association is said to play a role defined as role type. In topic maps, the statement "Crystal structure of Aspirin has been predicted" and "Aspirin's Crystal structure was predicted" are the same. Therefore the association can be navigated in both directions and associations need not be restricted to two topics.

Occurrences are the information resources relevant to a topic. Thus for "crystal structure", occurrences could be the reference of a journal or URL from which the information was obtained. This gives the added advantage of giving the user an extra set of links to different resources on the Topic[72].

An additional feature of topic maps is types. For example types for the topic "Aspirin" could be "drug", "crystal structure" or "analgesic". Types in a topic map are also topics. Therefore the resulting model can capture any type of information and is infinitely extensible and adaptable. One advantage of the topic map approach is that in the process of creating a topic map for a set of existing data (documents or databases), several concepts in the data set will be already covered without having actual identities of their own. This facilitates searching and once you have found what you are looking for you can also learn about it.

One example would be to create a topic Map of Organic Chemistry. If information on "carbohydrates" was searched for, there would be a topic representing the concept "carbohydrate" and from that topic information would be presented in the following way; "Carbohydrates are a group of organic compounds " (topic type), "Carbohydrates are produced naturally by green plants from carbon dioxide and water"(association), "carbohydrate literature" (occurrence) and so on.

Charles Goldfarb called topic maps "the GPS of the information universe" because they make information findable by giving every concept in the information its own identity and providing multiple redundant navigation paths through the information space[13].

In all there are wide range of services for building ontologies and an ontologist's choice of editor, methodology and language depends on the type and use of ontology being constructed. Domain and application ontologies are the two types of ontologies used mainly in Semantic Web Portals. When an ontology is created to be deployed as an indexing system in a SW portal, other facets of ontology management have to be considered. These factors include reasoning, inference, ontology versioning and management, knowledge acquisition and finally interoperability with current Web standards. In the following Chapters an appropriate editing environment and language will be

chosen for the purpose of converting the existing Topic Map of Chemistry in the CCLRC data portal into an ontology.

CHAPTER 3

METHODOLOGIES TOOLS AND LANGUAGES FOR BUILDING ONTOLOGIES

In this chapter the main methodologies, tools or development environments and languages used for building ontologies will be described. The aim of this is to present an introduction into the various tools methodologies and languages currently existing.

3.1 Methodologies for building Ontologies

Although large-scale ontologies already exist, ontology engineers are still needed to construct the ontology for a particular task or domain, and to maintain and update the ontology to keep it relevant and up-to-date.

An ontology can be created from scratch, from existing ontologies only, from a body of information sources only; or a combination of the last two approaches. Ontological Engineering is still fairly an immature discipline and several research groups propose various methods more commonly known as methodologies for building ontologies. There is no consensus between these groups and each employs its own methodology. Consequently the most current and popular methodologies used in ontology development will be described and discussed (Table 3.1).

Criteria	Uschold & King	Grüninger & Fox	METH -ONTOLOGY	Amaya Berneras et al. (Kactus)	SENSUS	On-To-Knowledge
Detail of methodology	Very Little	Little	A lot	Very little	Very little	Very Little
Recommendation for formalization	None	Logic	None	None	None	None
Strategy for building applications	Application Independent	Appl. semi-dependent	Appl. Independent	Appl. Dependent	Appl. Semi-dependent	Appl. Dependent
Strategy for identifying concepts	Middle-out	Middle-out	Middle-out	Top-down	Top-down	Top-Down
Recommended life cycle	None	None	Yes	None	None	Yes

Table 3.1 A comparison between the methodologies. Application-dependent: the ontology is built on the basis of an application knowledge base, by means of a process of abstraction. Application-semi dependent: possible scenarios of ontology use are identified in the specification stage. Application-independent: the process is totally independent of the uses to which the ontology will be put in knowledge-based systems, agents, etc.

The **Ushold and King's** methodology is based on the development of the Enterprise Ontology, an ontology of enterprise modelling processes. The guidelines proposed by this methodology are: to identify the purpose of the ontology, to build the ontology, to evaluate it and finally to document it. During the building activity the authors propse capturing knowledge, coding it and integrating other existing ontologies inside the current one. The following three strategies are proposed for identifying the main concepts in the ontology: a top-down approach, in which the most abstract concepts are identified then specialized into more specific concepts; a bottom up approach, in which the most specific concepts are identified first then generalized into more abstract concepts; and a middle-out approach in which the most

important concepts are identified first then generalised and specialised into other concepts. This methodology is application independent as the ontology development process is completely independent of the purpose of the ontology[51].

The next methodology is one developed by Gruninger and Fox and is based on the development of the Toronto Virtual Enterprise (TOVE) project ontology within the domain of business processes and activities modelling. This system was inspired by the development of knowledge-based systems using first order logic. In this method an informal description of the specifications to be met by the ontology is made and then formalised. This methodology adopts a middle-out strategy for identifying concepts and is semi-application dependent. This means that possible scenarios of ontology use are identified earlier on[38].

The approach of Amaya Berneras et al. is set within the Esprit kactus project. This methodology employs a top-down approach for identifying strategies. This process is application dependent. This means that each time an application is being created, the ontology representing the knowledge required for that particular application is built. This ontology can be developed by reusing others and can also be integrated into the ontologies of later applications[39].

METHONTOLOGY is a methodology created in the Artificial Intelligence Lab from the Polytechnic University of Madrid. It is used for building ontologies either from scratch or by a process of reusing or re-engineering other existing ontologies. The METHONTOLOGY framework enables construction of ontologies at the knowledge level and includes: identification of the ontology development process, a life-cycle based on evolving prototypes and particular techniques for carrying out each activity. The ontology development process identifies which tasks should be performed when building ontologies (i.e. scheduling control, quality assurance, specification, knowledge acquisition, and conceptualisation and configuration management). The life cycle identifies the stages through which the ontology passes throughout its lifetime, as well as the inter-dependencies with other ontologies. This system adopts a middle-out strategy and is application independent[2]. Application-independent means the process is totally independent of the uses to which the ontology will be put in knowledge-based systems, agents, etc

The SENSUS-Based Methodology was developed at the IST (Information Sciences Institute) to provide a broad-based conceptual structure for developing machine translators. SENSUS uses a top-down approach for deriving domain specific ontologies from giant ontologies and is semi-

application dependent[20]. Application-semidependent means the possible scenarios of ontology use are identified in the specification stage[35].

The On-To-Knowledge methodology includes the identification of goals that should be achieved by knowledge management tools and is based on an analysis of usage scenarios. The steps proposed by the methodology are: kick-off, where specified, competency questions are identified, potentially reusable ontologies are studied and a first draft of the ontology is built; refinement, where a mature and application-oriented ontology is produced; evaluation, where requirements and competency questions are checked, and the ontology is tested in the application environment; and ontology maintenance[52]. This method is application dependent this indicates that the process by which the ontology is built is totally independent of the uses to which it will be put in knowledge-based systems, agents e.t.c[34].

3.2 Ontology Implementation Languages

An ontology language is a formal language by which an ontology is built. At present several Ontology implementation languages exist and they can be divided into three main types; vocabularies defined using natural language, object based-knowledge representation languages such as frames and UML, and languages based on first order predicate logic such as Description Logics[42].

The more classic languages developed at the beginning of the 1990s include KIF-based Ontolingua, LOOM, OCML and Frame Logic (FLogic). The KR (knowledge representation) paradigm underlying these languages was based on first order logic (KIF) or a combination of frames and first order logic (i.e. OCML, Ontolingua, OCML and FLogic) or on DL (Loom). These languages follow a syntax based on LISP (LISt Processing) with the exception of Flogic and are in a phase of stable development[5].

More recent ontology implementation languages include; RDF, RDF Schema, XOL, SHOE, OIL, DAML+OIL and OWL. These languages are in a constant state of evolution and are called "web- based" languages (Figure 3.1). As XML has emerged as a standard language to exchange information on the web, they have been created based on XML to implement ontologies[43].

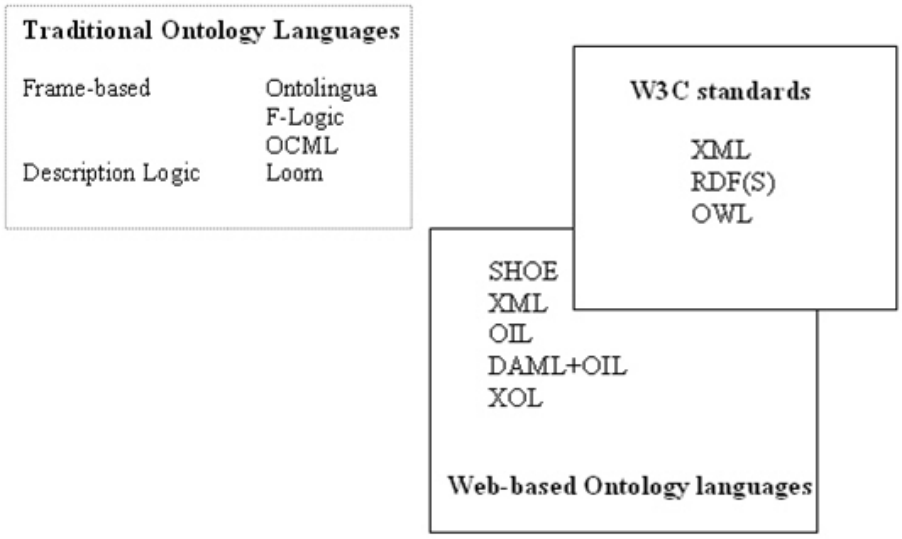

Figure 3.1 ; Classification of Ontology Implementation Languages

3.3 Classic Ontology Specification Languages

Ontolingua was created in the early 1990s to support the design and specification of ontologies with a clear logical semantics based on KIF. Ontolingua extends KIF with additional syntax to capture intuitive building of axioms and is based on first order predicates logic. It allows representation of meta-knowledge and non-monotonic reasoning rules. Loom was developed at the same time as Ontolingua and is based on DL and provides automatic classifications. OCML was developed later in 1993 it is a frame based language and can be considered as a kind of operational Ontolingua. This is because it provides deductive and production rules and function evaluation facilities for its constructs. FLogic was developed in 1995 and combines frames and first logic. Its inference engine Ontobroker can be used for constraint checking and deducting new information[33].

3.4 Web-Based Ontology Specification Languages

With the birth of the Internet web based ontology languages were created. The World Wide Web Consortium (W3C) recommends a number of standards as part of the Semantic Web stack these include XML, RDF(S), DAML+OIL and OWL(Figure 1.2). XML and RDF form the foundations of the semantic web, and OIL, DAML+OIL and OWL have been developed as extensions to RDF[53]. UML is also viewed as a language for designing an ontology[42]. Software engineers can design an ontology by organizing object classes in a class hierarchy and creating relationships among them[54].

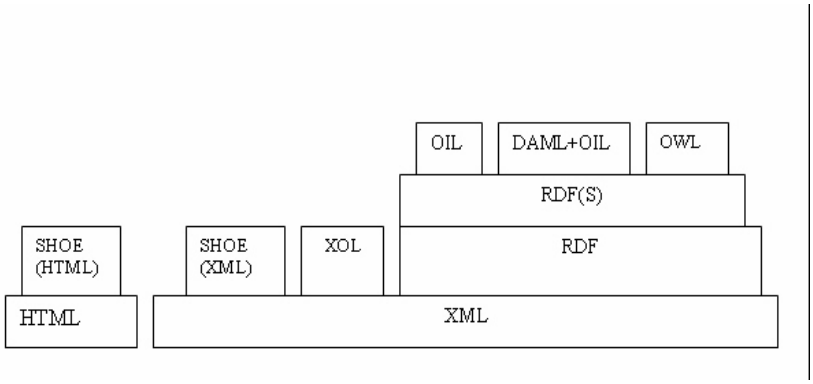

Figure 3.2 Illustration of W3C recommended stack of ontology mark-up languages for the semantic web[20].

EXtensible Markup Language (XML) is an open standard for describing data from the W3C, and is designed to be more flexible and powerful than HTML. XML Schema is a language for restricting the structure of XML documents. RDF (Resource Description Framework) was developed by the W3C as a framework for describing and interchanging metadata. RDF schema was built

as an extension to RDF, and the combination of RDF and RDF Schema is known as RDF(S)[55].

SHOE (Simple HTML Ontology Extensions) was built in 1996 at the University of Maryland, as an extension of HTML to incorporate semantic knowledge in ordinary web documents through the annotation of HTML pages. SHOE only allows representation of concepts, their taxonomies, n-ary relations and instances. SHOE does not have any pre-defined ontologies, categories, relationships, or inferences. XOL is an XML-based ontology-exchange language developed by AI centre of SRI international in 1999 for exchange of mainly biological ontologies[56].

The Ontology Inference Layer (OIL) knowledge interchange language is defined as an extension of RDF. It was developed in European IST the ON-To-Knowledge project and is both a representation and exchange language for ontologies. The language is unified by primitives from frame-based languages, formal semantics and reasoning services from description logics. The ontology description is divided into 3 layers object level (concrete instances), first-meta level (ontological definitions), and second meta-level (relations) as well as a limited set of axioms[57].

DAML+OIL specification was created by a joint committee from the US and the EU in the context of the DARPA project (DARPA Agent Mark-up Language) released in 2000. It adds a DL based and KR primitives to RDF(S). Both OIL and DAML+OIL allow representing concepts, taxonomies, binary relations, functions and instances[58].

OWL the Web ontology language is an emerging ontology language standard that has been optimized for data exchange and knowledge sharing. OWL is the mainstream tool for modelling ontologies and was developed by the Web Ontology working group in 2001. It is used when information contained in documents needs to be processed by applications, as opposed to situations where the content only needs to be presented to humans. OWL is a DL based language and can be used explicitly to represent the meaning of terms in vocabularies and the relationships between those terms[59].

Topic Maps (TMs) are an ISO (International Organization for Standardization) standard for the representation and interchange of knowledge, with an emphasis on the findability of information[43]. Topic maps have a standard XML-based interchange syntax called XML Topic Maps (XTM), a Topic Map Constraint Language (TMCL), as well as a standard API called Common Topic Map Application Programming Interface (TMAPI) [43].

3.5 Ontology Development tools

Ontology development or engineering tools include suites and environments that can be used to build a new ontology from scratch or by reusing existing ontologies[34]. Since the mid-nineties there has been an exponential increase in the development of technological platforms related with ontologies[4]. Some of the older environments which are in a state of stable development include Ontosaurus, Ontolingua and WebOnto. More recent tools include; OntoEdit, OilED, WebODE, Protégé, Chimera and DAG-Edit. A general description of these Ontology Development Tools is shown in Table 3.2.

Editor	Ontolingua	OntoSaurus	Chimaera	OiLED	WebOnto	OntoEdit Professional	OntoEdit Free	DAD-Edit	WebODE	Protege	SWOOP
Developers	KMI (Open University)	ISI (University of Southern California)	Stanford Knowledge systems Lab	University of Manchester	KMI (Open University)	Ontoprise	Ontoprise	Berkley Drosophila Genome Project	Ontology Group (UPM)	SMI (Stanford University)	MindSwap
Release Date	Feb-2003	Mar-2002	Aug 2003	Oct-2003	May 2002	Mar-2004	May 2002	April 2004	Nov 2003	Jan-2004	March 2005
Availability	Free Web access	Open source and free Web access to evaluation version	Open source free Web access to evaluation version	Open Source	Free Web Access	Software licence	Free Web Access	Freeware	Software licence and Free Web Access	Open Source	Free web acess

Table 3.2 General Description of Ontology Development Tools

The Ontoligua server was the first ontology tool created in the early nineties. Developed in the knowledge systems laboratory (KSL) at Stanford University in the early 1990s, it was built to ease the development of Ontolingua ontologies. Initially the main module inside the ontology server was the

ontology editor and other modules like Webster (an equation solver) and OKBC (Open knowledge Based Connectivity) server were included later on[33].

Ontosaurus was developed around the same time as ontolingua by the Information Sciences Institute (ISI) at the University of South California. OntoSaurus consists of two modules: an ontology server, which uses Loom as its knowledge representation system and a web browser for Loom ontologies. Translators from Loom to Ontolinua, KIF, KRSS and C++ exist and Ontosuarus ontologies can also be accessed with the OKBC protocol[20].

WebOnto is an ontology editor for OCML (Operational Conceptual Modelling Language) ontologies and was developed at the Knowledge Media Institute KMI at Open University. This tool is a Java applet coupled with a customised web server and allows users to browse and edit knowledge models over the internet. The fact that WebOnto was able to support collaborative ontology editing was a major advantage at the time[60].

Each of the environments described above (Ontolingua server, Ontosaurus and WebOnto) was created solely for the purpose of browsing and editing ontologies in a specific language (Ontolingua, LOOM and OCML respectively). These older generation editors were hardly extensible compared to the engineering environments of today. The newer generation of ontology-

engineering environments are more advanced and ambitious than their predecessors. They possess highly extensible, component based architectures, where new modules can easily be added to provide more functionality to the environment[4].

3.6 Newer Generation Ontology development tools

WebODE is an easily extensible and scalable ontology workbench developed by the Ontology Group at the Technical University of Madrid (UPM). It is the successor of the Ontology Design Environment (ODE). WebODE is used as a Web server with a Web Interface. The core of this environment is the ontology access service which is used by all the services and applications plugged into the server. The ontology editor also provides constraint checking capabilities, axiom rule creation and parsing with the WebODE Axiom Builder (WAB) editor, documentation in HTML, ontology merge, and ontology exportation and importation in different formats (XML\RDF(s), OIL, DAML+OIL, CARIN, Flogic, Java and Jess). Its inference built in service uses Prolog and a subset of the OKBC protocol[61].

OilED is an ontology editor that allows the user to build Ontologies in OIL and DAML+OIL. OilED is a DL based tool and is installed locally. OilED was

developed by The University of Manchester, the Free University of Amsterdam and Interprice GmbH. The current versions of OilED do not support the development of large scale ontologies, in terms of migration, integration, versioning argumentation and other activities involved in ontology construction[62].

OntoEdit is developed by AIFB, University of Karlsruhe and is built on top of a powerful internal ontology model. The internal ontology model can be serialized using XML, which supports the internal file handling. It supports F-Logic, RDF-Schema and OIL. In the current version OntoEdit has an interface to the Karlsruher F-Logic Inference Engine (the backbone of OntoBroker), in the next version the FaCT system will be accessible from OntoEdit. The tool is based on a flexible plugin framework. The professional version of OntoEdit contains several additional plug-ins, a collaborative environment and inference capabilities[63].

DAG-Edit is an open source software implemented in Java that provides an interface to browse, query and edit Gene Ontology (GO) or any vocabulary with a directed acyclic graph (DAG) data structure. DAG-Edit is a DL (Description Logics) based environment and was created for the development of GO bio-ontologies[34].

Chimaera is developed in the Knowledge based systems Laboratory at Stanford University. It was built on top of Ontolingua and is a software system that supports users in creating and maintaining distributed ontologies on the web. The user interacts with Chimaera through a browser such as Netscape Navigator or Microsoft Internet Explorer. It supports users in tasks like loading knowledge bases in differing formats, reorganizing taxonomies, resolving name conflicts, browsing ontologies and editing terms[64].

Protégé is one of the most widely used editing tools and has been developed by the Stanford Medical Informatics (SMI) group at Stanford University and the information management group at. The design and development of Protégé has been driven primarily by two goals: to be compatible with other systems for knowledge representation and to be an easy to use and configurable tool for knowledge extraction. It is an open source, standalone application with an extensible architecture, which assists users in the construction of large electronic knowledge bases. The core of this environment is an ontology editor. Numerous plug-ins provide several functions including alternative visualization mechanisms, management of multiple ontologies, inference services and ontology language importation/exportation[65]. Protégé is a development environment for ontologies and knowledge-based systems. The OWL Plugin is an extension of Protégé with support for the Web Ontology Language (OWL). The protégé OWL plugin enables users to; load

and save OWL and RDF ontologies, edit and visualize classes, properties and SWRL rules, define logical class characteristics as OWL expressions, execute reasoners such as description logic classifiers and finally to edit OWL individuals for Semantic Web markup[31].

SWOOP is developed by Mindswap (Maryland Information and Network Dynamics Lab Semantic Web Agents Project) and is a free hypermedia-based Featherweight OWL Ontology Editor. It employs a web-browser metaphor for its design and usage and is meant for rapid and easy browsing and development of OWL ontologies. It has a plug-in architecture which provides some useful extensions[66].

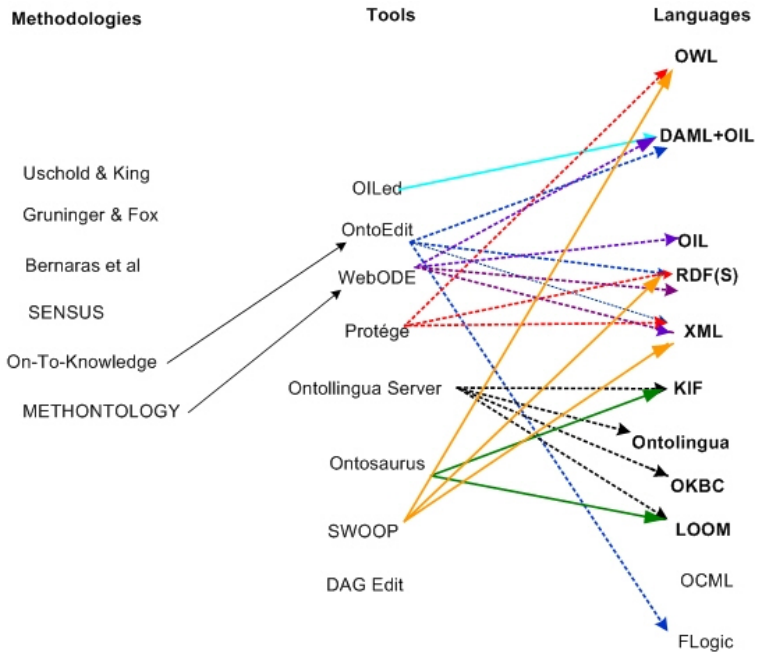

Figure 3.3 Relationship between methodologies, languages and tools for building ontologies.

3.7 Graphical Ontology Editors.

Growl is a visualisation and editing tool for OWL and DL ontologies and provides graphic representation for all OWL constructs and all common DL expressions. It was created at the Gunds Institute for Ecological Economics at the University of Vermont. Growl is a stand alone editor implemented as a java applet. It provides advanced methods of navigation within large ontologies[67].

OntoTrack is an ontology authoring tool used for mouse enabled-graphical editing of ontologies in OWL Lite. It is developed at the Department of AI at the University of Ulm. The ontology is laid out as a directed acyclic graph (DAG with classes as nodes and subsumption relationships as arrows either in top-down, left-right, bottom-up, or right-left orientation. OntoTrack uses an external OWL reasoner called RACER to make all implicit subsumption relationships explicitly visible to the user[68].

Protégé contains a tool called OWLViz that can be used to visualise ontologies built using the Protégé OWL plugin OWLViz uses the same colour scheme as Protégé-OWL and so that primitive and defined classes can be distinguished, computed changes to the class hierarchy may be clearly seen, and inconsistent

concepts are highlighted in red. OWLViz enables the class hierarchies in an OWL Ontology to be viewed and incrementally navigated, allowing comparison of the asserted and inferred class hierarchy[69].

Altogether several methodologies, languages and tools for building ontologies have been covered and figure 3.3 shows the relationships between them. With regards to methodologies, there are only two editors WebODE (METHONTOLOGY) and OntoEdit (On-To-Knowledge) that support methodologies. Classic ontology implementation languages include Ontolingua, LOOM, OCML and Frame Logic (FLogic), whereas more recent web based languages include RDF Schema, XOL, SHOE, OIL, DAML+OIL and OWL. Older ontological editing environments have a stronger relationship with a specific language, while newer environments are standalone and more interoperable with a variety of languages. In the next Chapter, the recent languages and ontology editors described in this chapter will be evaluated.

CHAPTER 4

CHOOSING AN ONTOLOGY LANGUAGE AND EDITING ENVIRONMENT

In this Chapter the ontology implementation languages and editing environments described in the previous chapter will be discussed and evaluated. The aim of this evaluation is to enable an editing environment and language to be chosen for building the Chemical Ontology. A large number of free and commercial ontology development tools exist on the market today and each has its own strengths and weaknesses.

The evaluation process began with the selection of some known tools and languages for building ontologies. The selection process involved a review of current literature with reference to prior knowledge of members of the information management group at the University of Manchester and information system technologists at CCLRC. Additionally Google searches were undertaken for completeness. Protégé, Chimaera, WebODE, OilED, SWOOP, OntoEdit and DAG-Edit were selected to be evaluated according to the following group of criteria. In the last section of this Chapter, tools for graphically editing the ontology will be assessed, this aims investigate

methods by which scientists themselves can edit their ontologies rather than relying on ontologists and software developers.

4.1 Criteria for evaluating Ontology Editors

1. Software architecture and Knowledge representation
2. Functionality
4. Interoperability and extensibility
3. Inference services
4. Scalability
5. Usability

Software architecture and Knowledge representation considers the editor's architecture (standalone, client/server and n-tier application, failure tolerance and backup management[4]. The Knowledge Representation (KR paradigm) underlying the knowledge model of the tools will be presented[4]. Editors can be based on for example traditional frame based KR models or description logics.

Functionality is the degree to which the user can add, modify or remove classes, instances and properties. This includes whether the editor allows

undoing and redoing of the addition, modification and deletion of sub or super classes. The developer should be able to undo changes made to the ontology as this would decrease unintentional errors incurred during ontology development. These are essential if an ontology is to be built in a professional manner. Additionally transparency is important that a change in one part of the ontology should be transparent in the other parts. For instance changes made to a super class are transparent in all the sub class instances. This would prevent errors and reduce time wastage.

Interoperability includes information about the tools interoperability with other ontology development tools and languages as well as translations to and from other ontology languages[11]. It is extremely important for ontology editors to allow import and export of data into different formats including languages like XML, RDF and OWL. This promotes use of the ontologies by a wide range of software developers. It increases interoperability between different tools. Furthermore storage of the ontology in database format is crucial as this will allow information to be stored in a central unit that can easily be accessed by members of the community.

The editors should be extensible, in built or external facilities like plug-ins make editors more flexible creating numerous additional features to the basic editor. For example an external facility to draw or create a visual model of the

ontology would enable users to understand the ontology content better if it is visually displayed[29].

Inference services are used to deduce new knowledge from the ontology or to check inconsistencies with its formal axioms. Inference services can either be built in or external[30].

Scalability is important as it deals with the issue of the capability of a system to increase performance under an increased load when new resources are added[50]. The editor should allow the addition of more classes, properties, relations and instances to an existing ontology. However some editors might not allow the addition or modification of more classes, properties, relations and instances if they have been populated[61].

Usability includes analysis of Graphical editors, help systems, collaborative working function and the provision of reusable ontology libraries[18].

Learnability and Availability: The tool should be readily available and easy to access either from the internet or other sources. Installation of the software must be easy and straightforward. It is also important to know the time required to learn how to use a particular ontology editor as all projects have deadlines. Help systems are crucial and it is important for tools to be associated with either manuals or online tutorials. Collaboration is essential in the process of building very large and extensive ontologies[16].

Reuse: The degree to which existing ontology files, classes, instances and slots can be reused is significant. Reuse of ontologies as a whole creates homogeneity, reuse of classes facilitates sharing of knowledge and shortens development time of the ontology thereby decreasing development costs. Furthermore reuse of slots and instances reduces the cost of development as there is no need to re-formulate data entry.

Graphical User interface: It is very important to have a user friendly interface in that it should support easy browsing of the ontology in the editing/development mode as well as in the presentation phase. Ontology creation is an iterative process therefore its important to move from one action to the other with few mouse clicks as opposed to going through a long tedious process. For example the user could want to add the relations and properties between classes simultaneously. It is important that the design tools are easy to use so that developers can easily learn and use the tools effectively.

Collaboration is essential in the process of building very large and extensive ontologies as large knowledge based systems require more than one individual in their development.

Editors		Chimaera	OiLED	OntoEdit/Onto Studio (Free)	DAG-Edit	WebODE	Protege	SWOOP
Graphical User Interface	Interface Clarity	-	0	+	0	+	++	++
	Interface consistency	+	+	+	+	+	+	+
	Local installation	No	Yes	Yes	Yes	Yes	Yes	Yes
	Stability	+	+	+	+	+	+	+
	Help system	+	-	-	+	+	+	0
	Undo/Redo	No	No	No	No	No	No	No
Importing and Exporting	Loading Ontology in different formats	+	+	+	0	+	+	++
	Saving Ontology in different formats	+	0	+	0	+	+	++
Functionality	Multiple inheritance	Yes	Yes	Yes	Yes	Yes	Yes	Yes
	Consistency checking	+	-	-	_	-	+	+
	Resusable ontologies	+	+	+	+	+	+	+
	Example ontologies	+	-	-(None)	+	+	+	+
	Ontological help	+	+	-	+	+	+	-
Collaboration		+	-	-	-	+	+	+

Table 4.1 Summary of the results of evaluation of tools. A plus sign (+) means positive, more than one plus sign means a more significant positive (++), a zero (0) means reasonable and a minus (-) is negative evaluation.

4.2 Summary of Results of Evaluation of Editors

The editing environments were evaluated according to the group of criteria described previously and the results of this evaluation can be viewed in Table 4.1.

Software Architecture and knowledge representation

Many of the tools are moving towards Java-based platforms: SWOOP, DAG-Edit, OiLEd, Protégé 2000 and OntoEdit/OntoStudio. WebODE and Chimaera are web based and possess client/server architecture. Backup management functionality is provided by WebODE, and extensibility facilities exist in OntoEdit, SWOOP, WebODE, Protégé and Chimaera. There are only two plug-ins available in SWOOP therefore it is not very extendible. WebODE, Commercial OntoEdit and Protégé allow ontology storage in databases. In DAG-Edit files can be saved in GO-Flat File, GO MySQL database, DO RDF Flat File and GO Postgres database[13]. The conceptual models of WebODE, OntoEdit, Protégé and Chimaera are frame-based. Whereas Ontosaurus and OilED are DL (Description Logic) based. The conceptual model behind DAG-Edit is the directed acyclic graph (DAG) model and SWOOP is not based on any KR paradigm.

Functionality

All tools allow editing functionality on different levels and representation of classes, attributes, instances and axioms. These classes, attributes and instances can be created, added, deleted, viewed and retrieved. Protégé, Ontolingua Server and Ontosaurus provide flexible modelling components like meta-classes. Multiple inheritances are supported by all tools, and

WebODE and OntoEdit support the METHONTOLOGY and On-to-knowledge methodologies respectively[19]

Classes, properties and relations can be added easily with the click of a mouse button in Protégé, however it is a difficult to import and export into different formats. Loading ontologies is easiest in SWOOP followed by Protégé then much more complex in the remaining editors. On the other hand, adding classes and properties to an existing hierarchy is difficult in SWOOP. When a super or subclass is added to the existing ontology several boxes pop up and this can be time consuming to close. Chimaera is the most difficult to use when adding classes individuals and relations. In Protégé it is difficult to view individuals of classes although an individuals tab has recently been added to the user interface.

Importing and Exporting of ontologies into different formats is generally difficult with all the tools. It was easiest to view ontologies in different formats in SWOOP as within the interface the ontology can be viewed directly in OWL, OWL abstract and RDF/XML formats. Viewing these formats was simple as they were only a mouse click away and did not involve a lengthily process as with the other editors. One unique feature of SWOOP that stood out from all the other editors was a Wiki-style in-line editing facility. In this editable mode it is possible to modify your ontology directly in RDF/XML

format. Additionally while in concise format, the ontology can be annotated, classes added, and imports made with the click of a button. Another unique feature is the extensive undo features in SWOOP, an element all the other editors do not have.

Editors	Ontolingua	OntoSaurus	Chimaera	OilED	Web Onto	OntoEdit Professional	OntoEdit Free	DAG-Edit	WebODE	Protege	SWOOP
Imports from languages	Ontolingua, IDL,KIL	LOOM, IDL,ONTO, KIF,C++	Protégé OKBC	RDF(S), OIL DAML+ OIL	OCML	XML,RDF(S) Flogic,DAML+OIL	XML,RDF(S) Flogic, DAML+OIL	GO, RDF, Postgres Database	XML,RDF(S) CARIN	XML,RDF(S) XML Schema	OWL,RDF ,XML, TXT
Exports to languages	KIF 3.0, CLIPS,CLIPS, IDL sentential format CML ATP,CML rule engine,EpiKit ,LOOM,OKBC syntax, PROLog SYntax KSL Rule Engine	LOOM,LDL, ONTO,KIF, C++	DAML+ OIL, XML, KIF,OKBC ,Loom,, Prolog	DAML+ OIL, SHIQ, HTML, and Dotty	OCML Ontolingua	XML,RDF(S) Flogic,DAML+OIL, SQL-3	XML,RDF(S) Flogic, DAML+OIL, OWL	Gene Ontology, RDF, Postgres Database	XML,RDF(S) XML Schema, FLogic	XML,RDF(S) XML Schema	RDF,XML, OWL,

Table 4.2 Interoperability of Tools

Interoperability and Extensibility

All the recent tools support import and export to and from many languages in a variety of formats (Table 4.2). Protégé supports the import of text files, database tables, XML and RDF files. Work can be saved in three different formats: as standard text files, in a JDBC database, experimental XML and RDF format. SWOOP is not very extensible with only twp plug-ins currently available and two others in development. Chimaera accepts input in 15 different formats including Protégé files and OKBC. OilED supports export to

10 different formats including RDFS, DAML+OIL, SHIQ, HTML, and Dotty. SWOOP accepts input in XML, OWL and RDF.

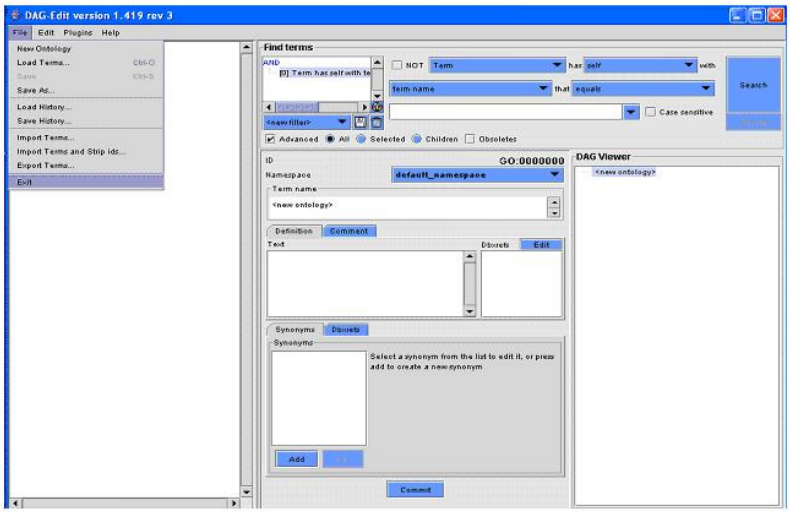

Figure 4.1 Snapshot of DAG-edit

Inference services

Protégé, WebODE, OILEd and OntoEdit Professional have in built inference engines. SWOOP has reasoning support in the form of an OWL Inference Engine. In OilED further reasoning functionality is provided by the Fact system. Protégé and WebODE also have external attached inference engines. DAG-Edit (Figure 4.1) and OntoEdit the free version (Figure 4.6) have no in-built inference services and none of the tools provide exception handling facilities. All the tools apart from DAG-Edit allow constraint and consistency checking. OiLEd and Protégé permit automatic classifications.

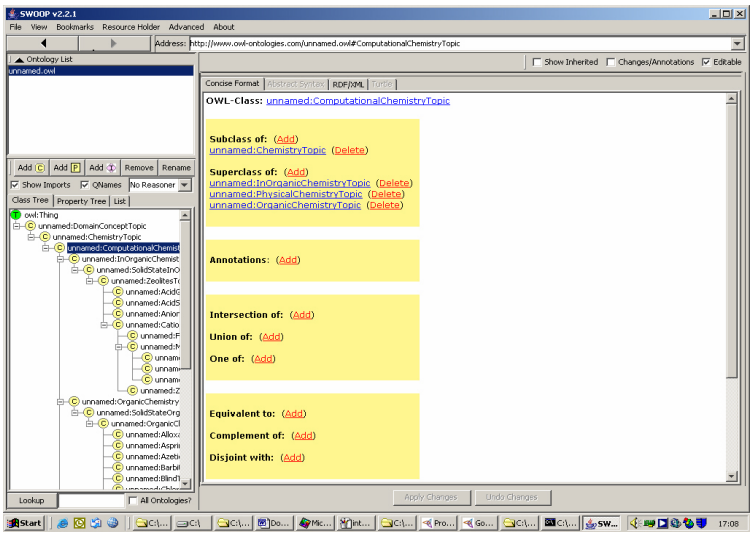

Figure 4.2 SWOOP GUI (Graphical User Interface). Taxonomy is shown on the right and the ontology in concise format on the left.

Usability

Chimaera (Figure 4.3) proved the most complex tool to learn how to use followed by DAG-Edit, OntoEdit/Onto Studio(Free version), then WebODE, Oiled, and SWOOP finally Protégé. With the web based tools (WebODE and Chimaera), there was difficultly with logging in and registering as a new user and they were also less stable as on some occasions the servers were down and editors crashed during use. All java-based tools were readily available on the internet and there were no problems with downloading and installation.

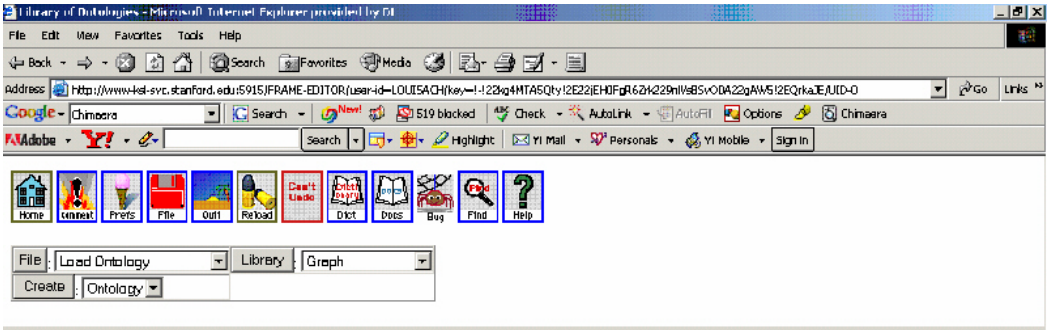

Figure 4.3 Chimaera. This is editor is form based and therefore more difficult to use.

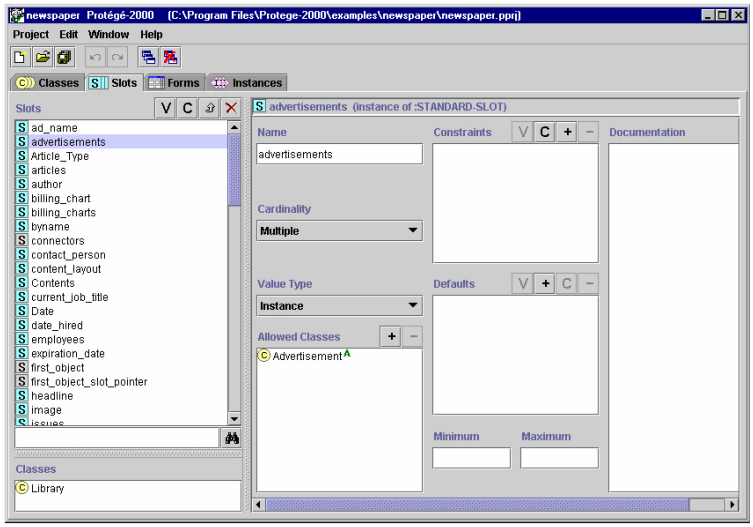

Figure 4.4 Protégé interface

Graphical User Interfaces

Protégé has the most user friendly and easy to use graphical interface (Figure 4.4). Oiled is like the notepad of ontology editors and has a clear and basic user interface (Figure 4.5). In Oiled the class hierarchy can be viewed in a pop up window when classes are double clicked on. In Protégé and WebODE the layout of the interface and visualisation of the ontology can be customised. SWOOP's main advantage lies in the display of classes in concise format which allows you to easily view sibling classes and superclasses of a particular class (Figure 4.2). Protégé and WebODE (Figure 4.7) allow graphical taxonomy viewing, pruning and zooming. In Protégé inherited slots can easily be recognized due to coded colours. Code colours and symbols are also used in Chimaera to indicate the properties of concepts. However Chimaera

has the worst looking interface (form based) and is extremely difficult and complicated to use (Figure 4.3). The hierarchy or tree structure in which the ontology is presented is not very clear and loading files is very difficult. In Oiled a list of over all concepts in the ontology is shown in alphabetical order. Both Protégé and OilED interfaces have several tabbed panes in which each pane shows relevant information about a current ontology component. DAG-Edit and Chimera show the components of the ontology within one window. The user interface of WebOnto is clear and users can create structures with multiple inheritances. Protégé, OilED, Chimaera, SWOOP and OntoEdit Professional include example ontologies which can be reused in the building phase. The Gene Ontology (GO) can be seen as an example ontology forDAG-Edit.

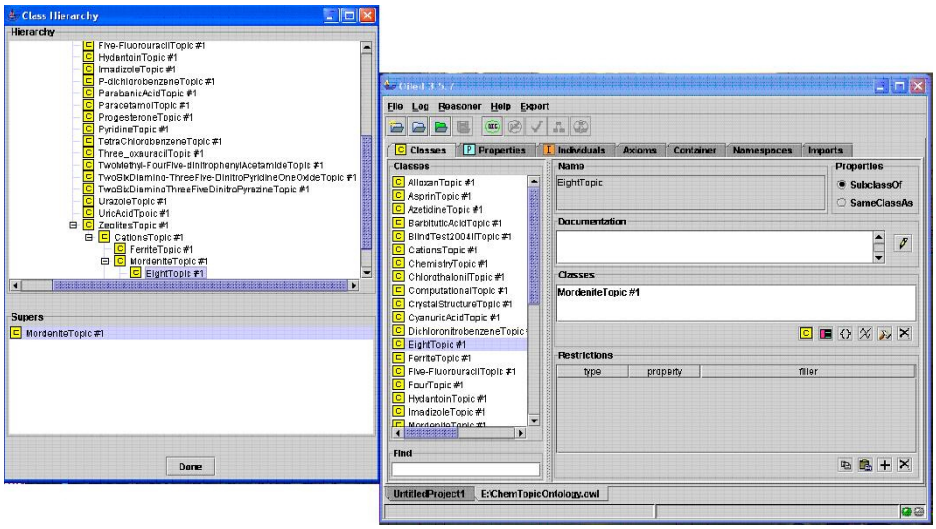

Figure 4.5 OiLed interface. Hierarchy is shown on the left.

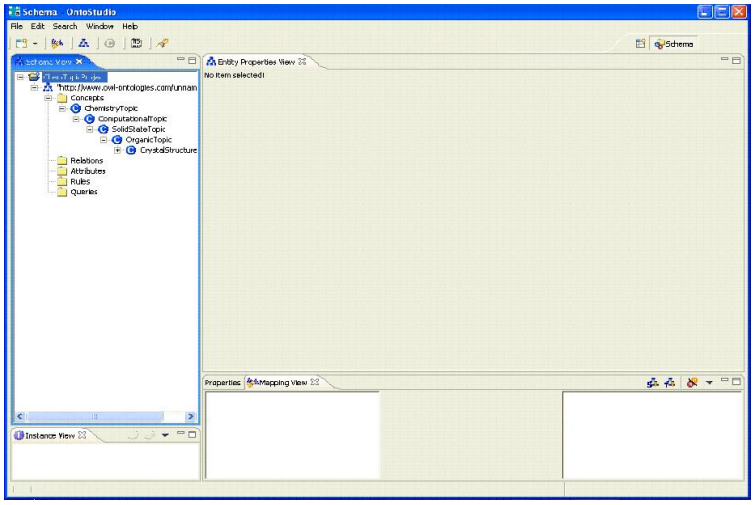

Figure 4.6 Snapshot of OntoEdit (free version) interface.

Help systems.

The Protégé help system is made up of a tutorial, user guide, shortcuts and FAQ. Additionally there are user manuals and tutorials available on the web on how to build ontologies using Protégé. SWOOP and OilED do not provide a help function in the user interface, and there is little help available on the internet. Chimaera provides the most support for advanced ontology builders in the form shortcuts which is unique.

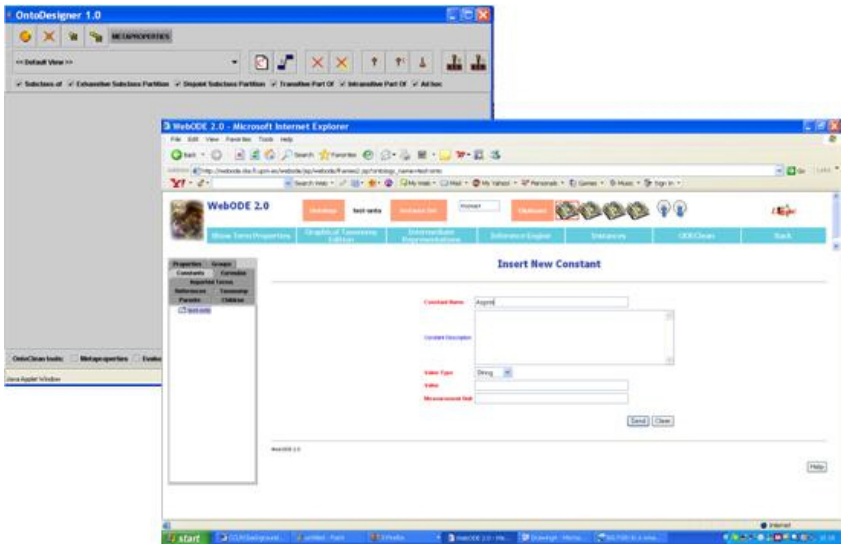

Figure 4.7 Snapshot of WebODE

Collaboration

Protégé, OntoEdit Professional, WebOnto, WebODE and Chimaera allow collaborative construction of ontologies. The classic tools Ontosaurus and Ontolingua also provide collaborative services. Chimaera provides the best support for multiple users and group members are alerted when changes have occurred in the ontology. In Protégé there is no support for multiple changes in the same component and users are not alerted about changes made to the ontology. Only one user at a time can edit an ontology with WebODE and during the editing process the ontology becomes locked. However in the locked state other users are still able to browse through the ontology[16]. One bonus feature in WebOnto is an annotation tool in the form of a "drawing pencil" that can be used to draw anything in the graphical window.

4.3 Editing the Chemical Ontology Graphically

In the next section of this chapter graphical methods of editing ontologies will be compared and discussed. Future editing of ontological components of the web portal will be carried out by domain experts mainly from a science background. Initially the editing of the chemical ontology will be restricted to the addition of classes and subclasses only. Therefore scientists require some means of editing the ontology which is currently in OWL form. The ontology is very difficult to read and understand in OWL, therefore it was exported into XML. In this format the ontology is quicker to edit and understand. Out of all the editors, SWOOP was the only ontology editor that allowed direct editing of the ontology however this tool was still not simple enough to be utilised. Consequently a series of XML editors were tried out. The editors were difficult to use, although most editors had classes and slots highlighted and colour coded. As the ontology will be edited by domain experts, it is important to find a tool for scientists that is lightweight, easy to use, understand and remember. After conducting a literature review it was found that there were currently no graphical XML editors available, however two graphical tools for editing ontologies in OWL, OntoTrack and Growl were chosen. Tools were evaluated according to the same criteria used for assessing the ontology editing environments previously described.

4.4 Evaluation of Onto Track

Whiles OntoTrack gives an impressive visualisation of the ontology some parts of my ontology were not visible as they had multiple ancestors. These parts are not shown in an expanded hierarchy but as clickable thumbnails in the editor. These thumbnails are meant to minimize the number of expanded classes visible at the same time. However this can be quite frustrating as you have to keep clicking on thumbnails to expand your hierarchy. New sub- or superclasses can be added by using the anchor mode of OntoTrack. In this mode, clickable triangle buttons appear when the mouse pointer is moved over class nodes. To create subsumption relationships between classes, arrows have to be drawn from one class to the other which makes adding super and sub classes to an existing hierarchy very difficult.

OntoTrack is compliant with current W3C web standards, OWL is the native language of this editor and ontologies can be stored and loaded in XML/RDF, N-Triple (N3) formats. This makes OntoTrack extensible and interoperable with current W3C standards. OntoTrack is designed to be almost solely an OWL lite authoring tool and cannot manage individuals, data type properties and annotation properties. This would make it difficult to edit extremely large, semantically rich ontologies, possessing up to a hundred or more classes with complex relationships between them including object and data type properties. The inability to manage OWL–DL is a disadvantage as in this

form OWL has inferencing capabilities which are extremely important when managing large domain and application ontologies especially in facilities such as web portals. OWL Lite supports those users primarily needing a classification hierarchy with simple constraints, whiles OWL DL supports those users who want the maximum expressiveness while retaining computational completeness. Consequently OntoTrack would be more suitable for graphically editing lightwieght ontologies consisting of few classes with few relationships between them.

4.5 Evaluation of Growl

Growl (figure 4.8) has a more user friendly user interface than OntoTrack and visualisation of the hierarchy is clearer. It is easy to move between classes and it provides nice views of which ever area of the ontology that you are in. In editable format, the ontology can be modified and includes visual tools for query design. Adding classes and subclasses to the existing hierarchy was difficult. The process of adding a class can take several tries as it is a mouse enabled graphical editing process. It took several attempts to add a class although removing or deleting a class was straight forward. Both Growl and OntoTrack are stable and have large, good help systems with user guides

Even though a graphical method of editing an ontology would be the best option for editing, the process of adding and removing classes is tedious and

time consuming in Growl and OntoTrack. Both of these editors will be suitable for graphically representing simple hierarchies or light-weight ontologies however they are not appropriate for building and editing heavy weight ontologies. Furthermore accessing current releases of these tools on the internet and installing OntoTrack was difficult.

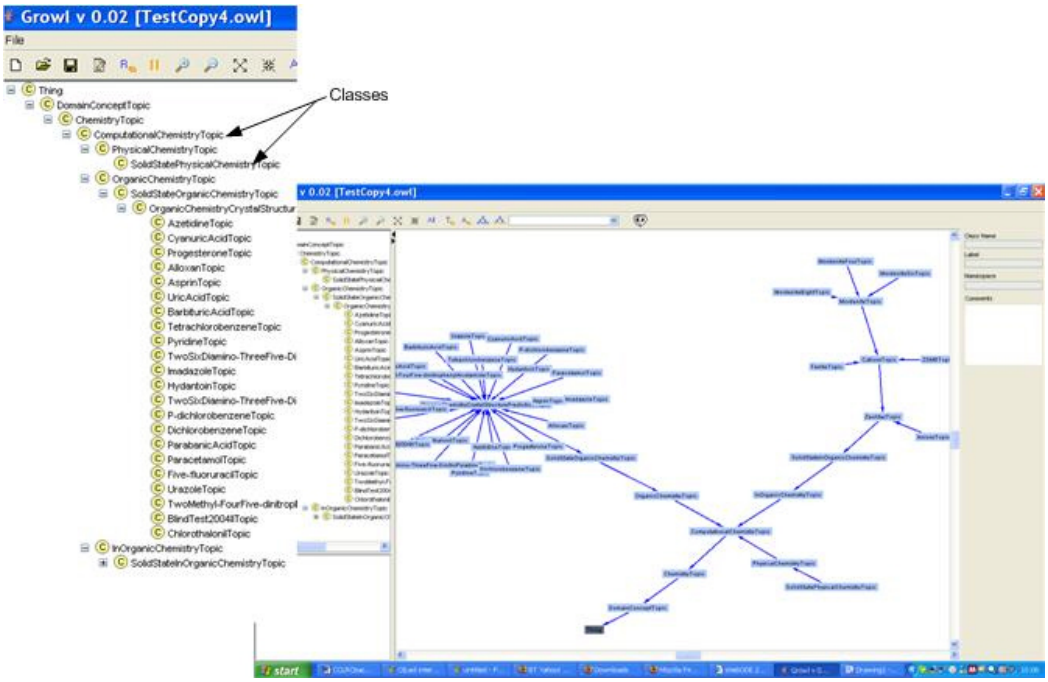

Figure 4.8 Growl Interface. The expanded hierarchy is shown on the left with a graphical view on the right

Protégé contains a tool called OWLViz that can be used to visualise ontologies built using the Protégé OWL plugin. A practical question would be; Why use not use OWLViz instead of Growl or OntoTrack to visualize the ontology. Graphical editing using a mouse to draw arrows to implement subsumption relationships is more difficult than simply clicking on a button within the Protégé user interface. Moreover the graphical visualisation of the ontology in OWLViz is comparable to that of Growl and OntoTrack.

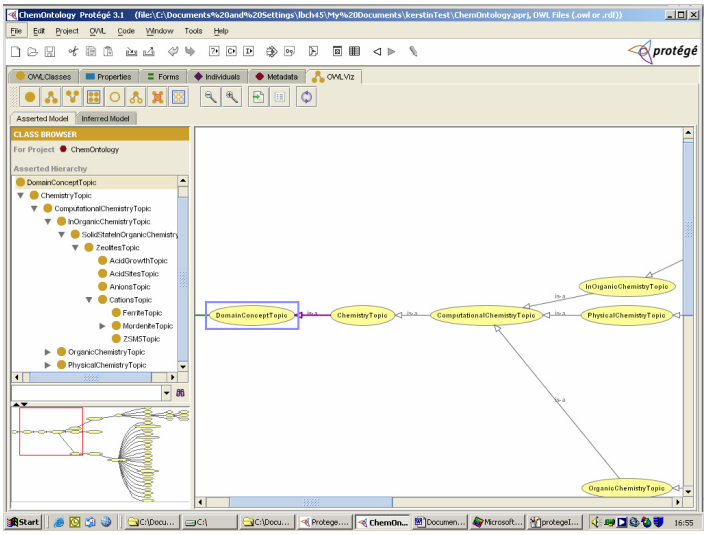

Figure 4.9 Owl viz tab in Protégé OWL interface

Instead of going through the rigmarole of downloading and installing Growl and OntoTrack the ontology can simply be visualised directly within the protégé OWL editor using OWLViz. Even though you may not be able to edit your ontology in OWLViz certain features give it an advantage over the other two graphical editors. OWLViz uses the same colour scheme as Protégé-OWL which makes the graphical interface consistent (Figure 4.9). Colour codes are used in order to distinguish between primitive and defined classes, computed changes to the class hierarchy may be clearly seen, and inconsistent concepts are highlighted in red. Additionally OWLViz has the facility to save both the asserted and inferred views of the class hierarchy to various concrete graphics formats including png, jpeg and svg.

After looking at a number of tools I picked Protégé as my editor of choice although SWOOP was a strong contender. Each tool has its weaknesses and strengths and so far as an editor Protégé's strengths out weigh it's weaknesses by far. Protégé's scalability and extensibility is what makes it standout from other tools as an engineering environment. Many ontology applications today may need to scale a few orders of magnitude larger than past applications[29]. The fact that Protégé has successfully been employed to build and use ontologies consisting of 150,000 frames demonstrates how powerful this tool is. It is important to look at scaling in terms of size of ontologies as well as numbers of simultaneous users. Not only does its user interface provide an environment that can easily be learned and remembered, but its open modular architecture makes it highly extensible. Furthermore the ever growing number of plug-ins make Protégé extremely flexible as they address most aspects of the ontology life cycle including advanced visualisation, ontology merging and version management and inference. The Graphical editors Growl and OntoTrack proved difficult to install and learn how to use. Consequently, they were time consuming and more suited to the editing of lightweight ontologies.

4.6 Choosing an Ontology Language

An ontology goes through several changes in its lifecycle and changes can occur because of technological advances, errors in earlier versions of the ontology, or release of a novel method of modelling a domain. Choosing a suitable language for building an ontololgy is dependent on the domain and type of ontology (heavy-weight or lightweight) being constructed. For ontologies designed to be utilised within an application, it is important to decide what the application needs in terms of expressiveness and inference services.

The expressivity of an encoding language is a measure of the range of constructs that can be used to formally, flexibly, explicitly and accurately describe the components of an ontology[20]. An expressive language contains a rich set of primitives that permits an extensive range of knowledge to be formalized. For large or heavyweight ontologies, representation of diverse knowledge is required. Therefore it is crucial that the ontology language should be as expressive as possible and at the same time support reasoning systems that scale optimally (i.e. it can be used to build very large ontologies)[71]. However there is usually a conflict between the expressiveness of a language and the inference capabilities associated with it[41]. Languages like Ontoligua are highly expressive but come with no means of controlling it

and as a result reasoning becomes difficult. Conversely Languages and tools based on description logic are less expressive but have more powerful reasoning support[59].

The ontology language should be interoperable with other tools and be natural and easy to learn with clear concepts and meanings. The concepts should be independent from syntax. Human readable and writable syntax is desirable as early ontology builders may work directly with the syntax. It is also important for the language to be compatible with current web and industry standards like XML, XML Schema, RDF and UML. This compatibility facilitates exchange of ontologies and data in a standard format. Furthermore compatibility with other standards eases development and deployment of the ontology. Finally the language should support the development of multilingual ontologies, and potentially provide different views of ontologies that are appropriate for different cultures[71].

Feature	Ontolingua	UML	OIL	DAML+OIL	XOL	SHOE	XML/RDF	OWL
Controlling Organisation	Stanford	OMG	W3C	W3C	Pangea/ISI International	None	W3C	W3C
Expressive Power	+++++	+	+	++	+	++	+++	++++
Number of Constructs	+++(Large)	0(Small)	0 (Small)	0(Small)	+(Medium)	0(Small)	0(Small)	++(Large)
Formal Syntax	+	+	+	+	+	+	+	+
Formal Semantics	++	+	+	+	+	-	+	++
Inference Engine	- (None)	- (None)	++	++	- (None)	++	No	Yes
Consistency Checking	Good	None	Weak	Weak	Weak	Weak	Weak	Medium
Support	+	+	+	+	0	0	++	++

Table 4.3 Summary of results of the evaluation of ontology languages. A plus sign (+) means positive, more than one plus sign means a more significant positive (++), a zero (0) means reasonable and a minus (-) is negative.

4.7 Evaluation of Ontology languages

Ontolingua, UML, XML/RDF, Oil, DAML+OIL, XOL and OWL were evaluated according to language support, expressive power, and inference capabilities. The evaluations results can be viewed in Table 4.3.

Evaluation Criteria

1. Language support includes questions on the depth of support and if the language if the language is a W3C standard?

2. Expressive power includes expressive capabilities of a language. This includes whether the language supports axioms? if metaclasses are allowed?

3. Inference capabilities includes whether there is reasoning support

4.8 Summary of Results of Evaluation

All the languages allow representation of concepts organised in taxonomies, binary relations and instances, however there are differences between the primitives available in each of the languages. The more expressive languages are Ontolingua, LOOM, OCML, OIL, DAML+OIL and OWL as they allow creating exhaustive disjoint subclass partitions. Axioms provide a good measure of the expressiveness of a language and formal axioms can be defined in Ontolingua, LOOM and OCML and FLogic[4].

UML has some limitations. For example, UML does not support the notion of conceptual equivalence. Also, UML has insufficient means for describing restrictions and constraints. Software engineers can use the Object Constraint Language (OCL), but it may be difficult to understand for business analysts. In short other ontology languages provide richer semantics than UML[74].

The main advantages of XML as a basis for an ontology specification language are firstly the definition of a common syntactic specification by means of a DTD (Document Type Definition) or XML schema, and secondly the fact that it is a standard for exchange of information across the web[72].

In relation to ontologies, RDF provides a standardized XML syntax for writing ontologies and a standard set of modelling primitives like instance-of and subclass of relationships. RDF(S) is very scalable, however it is not very expressive and just allows the representation of concepts, concept taxonomies and binary relations. Additionally RDF is fairly easy to use, but RDF Schema is more complex and difficult to learn. XOL is also a very restricted language with no inference mechanisms attatched[25].

OWL, FLogic and Oil possess sound and complete inference engines. Automatic classifications are performed by DL based languages LOOM, OIL and OWL. Exception handling is not addressed in general by any of the languages and FLogic is the only one that handles exceptions. Simple inheritance is supported by all the languages except XOL. Constraint checking is performed in all the classic ontology languages.

The major issue to be considered in ontology implementation is what the application needs in terms of expressiveness and inference services. However not all languages have the level of expressivity and reasoning capabilities required for creating a heavyweight ontology. Ontolingua is a highly expressive language. However the problem with Ontolingua is that no reasoning support has been provided, thus there is no means of controlling it.

OIL takes the opposite approach. It is a simple and limited core language, with a very powerful automated reasoning support (Fact reasoner)[62].

OWL is a language tailored to fit the specific needs of users and tailored to tackle this issue of expressivity versus inference. OWL provides three sublanguages, namely OWL Lite, OWL DL and OWL FULL. OWL Lite supports those users primarily needing a classification hierarchy and simple constraints. OWL DL supports those users who want the maximum expressiveness while retaining computational completeness (all conclusions are guaranteed to be computable) and decidability (all computations will finish in finite time)[59]. OWL Full is meant for users who want maximum expressiveness and the syntactic freedom of RDF with no computational guarantees[70]. Having knowledge that can be dynamically applied to find an answer, rather than predefined procedures, is extremely powerful when building a heavy-weight ontology. OWL provides the means to allow a machine to make the same sorts of simple inferences that human beings do. A set of OWL statements can allow you to conclude another OWL statement whereas a set of XML statements alone, does not allow you to conclude any other XML statements. Additionally, OWL offers a host of other standard properties such as equivalence ("childOf" on an English geneology site is the same as "enfantDe" on a French site), or unique properties (a social security number is associated with only one individual)[59].

Advantages of OWL over other ontology languages

OWL is the most recent development in standard languages from the W3C It is a revision of DAML +OIL Web ontology language and has more facilities for expressing meaning and semantics than XML, RDF and RDF-S. Therefore it has a richer set of operators like and, or and negation and is based on a different logical model which makes it possible for concepts to be defined as well as described[30]. Additionally the logical model makes it possible for concepts to be defined as well as described. Complex concepts can be built up in definitions out of simpler concepts. A reasoner can be used to check whether or the not all of the statements and definitions in the ontology are mutually consistent and can also recognise which concepts fit under which definitions. The reasoner helps to maintain the hierarchy correctly. This is particularly useful when dealing with classes with more than one parent[30].

Futhermore Owl allows classes to be defined precisely by listing the individuals that are the members of the class. For example, a class DaysOfTheWeek can be defined to contain to contain the individuals (and only the individuals) Sunday, Monday, Tuesday, Wednesday, Thursday, Friday and Saturday. This type of class is known as an enumerated class[30].

Moreover OWL allows classes, properties, individuals and the ontology itself to be annotated with various pieces of information or meta-data. These pieces of information may take the form of auditing or editorial information. For example, comments, creation date, author, or, references to resources such as web pages. OWL-Full does not put any constraints on the usage of annotation properties. However, OWL-DL does put several constraints on the usage of annotation properties[30].

There are also several annotation properties which can be used to annotate an ontology. Ontology annotation properties a have a range of a URI-reference that can be used to refer to another ontology. These URI-references include,

1. owl:priorVersion identifies prior versions of the ontology.
2. owl:backwardsCompatibleWith identifies a prior version of an ontology that the current ontology is compatible with.
3. owl:incompatibleWith identifies a prior version of an ontology that the current ontology is not compatible with.

This unique feature of OWL allows permits all of the identifiers from the prior version have the same intended meaning in the current version. Consequently, any ontologies or applications that reference the prior version can safely switch to referencing the new version[30]. Ontology versioning is

extremely important in Web portals as old versions of the ontology can be kept track of.

To sum up, the web ontology language OWL was chosen as it stands out from other ontology languages. This was because of its ability to provide formal semantics, built-in reasoning support and additional features such as metadata annotation and ontology versioning.

CHAPTER 5

OTHER METHODS OF BUILDING CONTROLLED VOCABULARIES TAXONOMIES AND THESAURI

In this chapter other methods of building controlled vocabularies including Topic Maps, SKOS (Simple knowledge Organisation systems) and ontology management systems will be explored. This aims to find out if other methods of building indexing systems could be viable in the CCLRC Data Portal. Currently there is overwhelming support for the design and development process of building ontologies, however there is still a lack of support for the process of deployment. Therefore the problems with deployment of ontologies will be highlighted and discussed.

A controlled vocabulary is a list of terms whose meanings are specifically defined. This list is controlled by and is available from a controlled vocabulary registration authority. All terms in the list should have an explicit, non-redundant meaning. If the same term is commonly used to mean different concepts in different contexts, its name will be explicitly defined to resolve this uncertainty. Controlled vocabularies are important practical tools in library and information science, where users are able to retrieve consistent results regardless of the term chosen[30].

A taxonomy is a set of terms that are arranged into a hierarchy and is an extension of a controlled vocabulary. A thesaurus is a networked collection of controlled vocabulary terms. Thesauri allow for expression of some relationships between terms including associative terms such as parent-child relationships. Although taxonomies and thesauri may relate terms in a controlled vocabulary via parent-child and associative relationships, they do not contain explicit guidelines to constrain the use of controlled vocabulary terms used to express something meaningful within a domain of interest[30].

5.1. Topic Maps and Simple Knowledge Organisation Systems (SKOS)

Topic Maps (TMs) are an ISO (International Organization for Standardization) standard for the representation and interchange of knowledge, with an emphasis on the findability of information. Dubbed the 'GPS of the information universe', topic maps are said to be 'ontology-agnostic' meaning they are able to support, represent and manage any kind of knowledge in any kind of ontological context, and independently of the constraints imposed by any ontology. Topic Maps are an alternative URI (Uniform Resource Identifier) based language to the W3C's RDF, RDF(S) and OWL DL. TMs provide powerful search and index functionalities[31].

Topic Maps represent information using topics (representing any concept, from people, countries, and organizations to software modules, individual

files, and events), associations (which represent the relationships between them), and occurrences (which represent relationships between topics and information resources relevant to them). Occurrences are generally outside the topic map document itself (although some of them could be inside it), and they are "pointed at" using whatever mechanisms the system supports, typically HyTime addressing or XPointers. Thus topics and their occurrences are separated into two layers. This separation gives TMs more functionality and flexibility when used for navigation or querying. Topics, associations, and occurrences can be typed, but the types must be defined by the creator of the topic maps, and is known as the ontology of the topic map[32].

Topic maps have a standard XML-based interchange syntax called XML Topic Maps (XTM), a Topic Map Constraint Language (TMCL), as well as a standard API called Common Topic Map Application Programming Interface (TMAPI)[34]. Merging Topic Maps and RDF is not yet achievable. It is possible to convert data back and forth between the two representations using simple, declarative, vocabulary-specific mappings. However semantic annotations in OWL can be translated directly into a topic map representation of the same information[33]. Recently an OWL construction for the ISO Topic Map data model has been proposed and the RDF-based Description Logic equivalent OWL DL could potentially be used to describe an explicit model for the semantic web[35].

There are several advantages to using Topic Maps especially when dealing with large quantities of information and knowledge[34]. Topic Map based applications are easily maintained and existing infrastructure can easily be developed without migrating or changing existing applications. Furthermore the concept of merging allows automated integration of topic maps from diverse sources into a coherent new topic map. New information repositories can be added individually or in groups and integration does not require large data conversions or high investments.

Disadvantages include a lack of interoperability with other tools and W3C standardised languages such as RDF/OWL DL and its lack of clearly defined formal semantics[32]. Topic maps would gain effectiveness and interoperability through explicit formalization of ontologies. OWL DL allows many constraints to be explicitly stated and until formal semantics is developed the use of topic maps will be restricted to development of lightweight ontologies. SKOS provides a model for expressing the basic structure and content of concept schemes (thesauri, classification schemes, subject heading lists, taxonomies, terminologies, glossaries and other types of controlled vocabulary). SKOS is a newer more extensible standard and provides flexible RDF scheme than the existing RDF schemes[36]. There are three RDF vocabularies under development: SKOS Core, SKOS Mapping and SKOS Extensions. The SKOS Core Vocabulary is an application of the Resource

Description Framework (RDF) that can be used to express a concept scheme as an RDF graph. There is also the SKOS web service API (SKOS API) for interacting with a KOS (knowledge organisation system) datasource[36].

SKOS-Core is intended as a complement to OWL and is easier to use than OWL[36]. It provides a basic infrastructure for building concept schemes, but does not carry the strictly defined semantics of OWL. As a result it is ideal for representing Knowledge Organisation Systems like thesauri, that cannot be mapped directly to an OWL ontology. SKOS is suitable for building lightweight ontologies and functions best in building thesauri and classification systems. However it is not suitable for building heavyweight ontologies.

5.2 Deployment of Ontologies

Most research in the field of ontology is focused on creating ontologies. Consequently most of today's tools support design and editing and there is still a lack of fully integrated environments which include facilities to manage deployment and maintain ontologies.

There is no global consensus or formalism for the deployment of ontologies, therefore it is still very much an ad-hoc process rather than a well defined engineering process. Deployment of ontologies in applications involves mapping the applications information assets (such as data schemas) to the

ontological model[37]. Reasoning is very important, and Ontologists need to have complete confidence in the reasoner. This is important for ontology deployment as with many web applications there is no human intervention to spot glitches in reasoning. Reasoning support based on DL(Description Logic) systems is sound and complete and has been highly optimised. This is because languages like OWL DL has benefited from many years of DL research[29].

In all there is still a lack of sophisticated methods and tools to support large scale ontological engineering and deployment. A common environment for effective development and deployment would be extremely valuable to the ontology community.

5.3 Ontology Management Systems

An ontology management system is equivalent to a database management system and allows an application to manipulate and query an ontology without worrying about how the ontology is stored and accessed, how queries are processed and how query results are retrieved. Ontology editing is not viewed as the most important part of an ontology management system and in instances where editing capabilities are absent it can be used in connection with an ontology editor such as Protégé 2000.

IBM Ontology Management System (also known as SNOBASE, for Semantic Network Ontology Base) is a framework for loading ontologies from files and

via the Internet and for locally creating, modifying, querying, and storing ontologies[39]. The IBM Ontology Management System provides a mechanism for querying ontologies with a user friendly programming interface for interacting with vocabularies of standard ontology specification languages such as RDF, RDF Schema, DAML+OIL, and OWL. The system uses an inference engine, an ontology persistent store, an ontology directory, and ontology source connectors. Applications are able to query against the created ontology models and the inference engine deduces the answers and returns results sets similar to JDBC (JavaTM Data Base Connectivity) [40]. Despite the fact that an ontology management system may not completely solve the main problems involved in deployment they are step in the direction of a more integrated environment, which addresses more aspects of the ontology life cycle than previously described tools. Other examples of Ontology Management systems include SymOntoX (Symbolic Ontology Management System) and knOWLer.

In all the fundamental difference between an ontology and a conventional representational vocabulary is the level of abstraction and relationships among concepts. Owing to their lack of formal semantics controlled vocabularies, thesauri, SKOS and Topic maps are not suitable for building heavy-weight ontologies but can be used for building light-weight ones. Deployment is still very much of an ad-hoc process with few guidelines, and

although ontology management systems support more aspects of the ontology life cycle, a fully intergrated editing environment would be of huge benefit to the ontological community. A practical question might be: What is the added value of an ontology compared to another knowledge representation system like for example a Topic Map? The answer is given in the next chapter where the conversion of a Topic Map of Chemistry into an ontology will be described.

CHAPTER 6

CONVERTING A TOPIC MAP INTO AN ONTOLOGY

In this chapter the conversion of a Topic Map of Chemistry into an Ontology will be described and discussed. Chemists, the domain experts, at CCLRC created a list of keywords to be used as the backbone for ontology creation and also as concepts for searching the source data. This list of keywords was used to build a Topic map that had previously been implemented in the Data Portal using XTM (XML Topic Maps). This topic Map served as an indexing system in the Data Portal and the knowledge it contained was implicit as the system would initially mainly be used by the domain experts that created the initial list of keywords. The aim of this conversion is to create an ontology to explicitly model concepts for the domain experts.

For many years simple classification systems such as thesauri have been used to organize digital resources such as medical libraries[72]. Topic Maps provide an approach that amalgamates the best of several worlds, including those of traditional indexing, library science and knowledge representation, with advanced techniques of linking and addressing. Even though topic maps have the advantage of being semantic free the CCLRC web portal requires a more flexible and versatile representation to accommodate intelligent information representation.

In applications, domain ontologies are leaps and bounds ahead of Topic Maps in that they have the ability to explicitly specify semantics and relations and to express them in a computer understandable language[75]. Ontologies allow sharing and reuse of ontologies which makes them more flexible than Topic Maps. Additionally, ontologies hold the promise of automatic knowledge acquisition and sharing and reuse of existing knowledge structures[10]. The ultimate goal of converting the existing Topic Map of Chemistry into an ontology is to enhance information representation and information retrieval systems in the Data Portal.

6.1 The Topic Map of Chemistry

The following Figures give an overview over the data and the associations used in the Topic Map. Topics have three kinds of characteristics: names, occurrences and roles in associations. All the topics in Figure 6.1 are of the same kind, and possess explicit names.

Figure 6.1 List of Topics present in Topic Map.

TOPIC_ID	TOPIC	PARENT
1	Chemistry	
2	Computational	1
3	Solid State	2
4	Organic	3
5	Crystal Structure Prediction	4
6	Aspirin	5
7	Pyridine	5
8	Paracetamol	5
9	Parabanic Acid	5
10	Zeolites	3
11	Acid Sites	10
12	Acid Growth	10
13	Azetidine	5
14	Chlorothalonil	5
15	P-dichlorobenzene	5
16	Barbituric acid	5
17	Alloxan	5
18	Cyanuric acid	5
19	Urazole	5
20	2,6-diamino-3,5-dinitro pyridine-1-oxide	5
21	2,6-diamino-3,5-dinitro pyridine	5
22	5-fluorouracil	5
23	Dichloronitrobenzene	5
24	Hydantoin	5
25	Imidazole	5
26	Blind Test 2004 II	5
27	2-methyl-4,5-dinirtophenyl acetamide	5
28	Protegesterone	5
29	Tetrachlorobenzene	5
30	Uric Acid	5
31	Cations	10
32	Mordenite	31
33	Ferrite	31
34	ZSM5	31
35	16	32
36	8	32
37	4	32
38	Isobarbituric acid	5
39	3-oxauracil	5

Fig 6.2 Topic Map Before conversion. The topic Map appeared in the form of a table with a topic ID and parent indicating the class to which each of the topics belonged. For example the parent of the topic Computational is 1, thus Computational is the sub class of Chemistry.

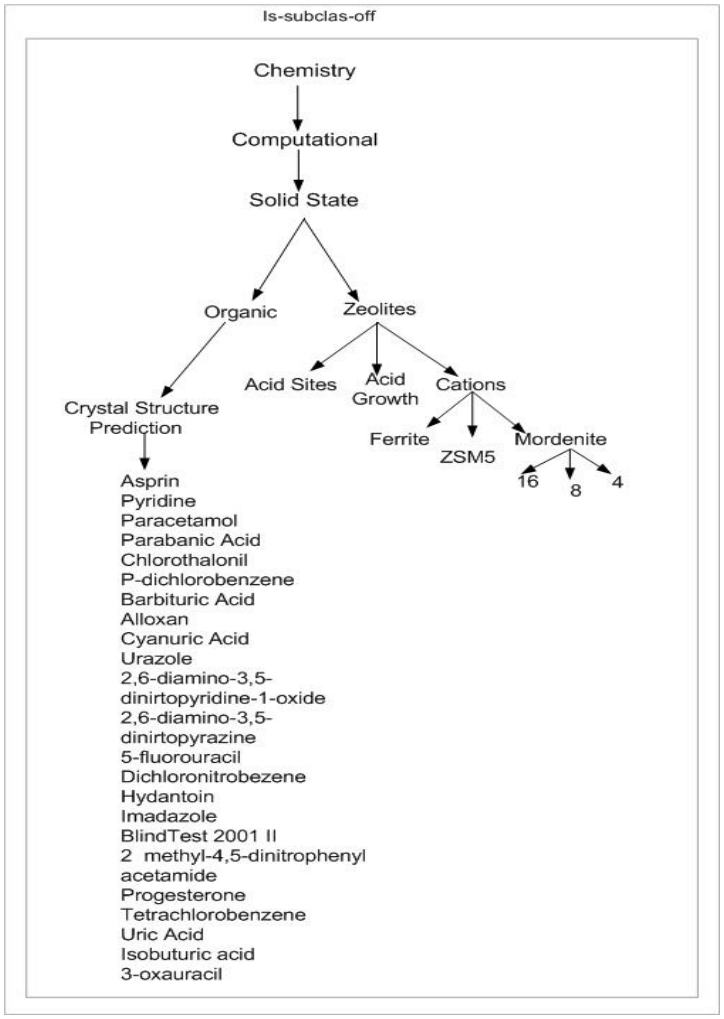

Figure 6.3 An illustration of associations between Topics. Arrows represent subclass associations between topics.

In Figure 6.3, Computational is a sub class of Chemistry and Solid state a subclcass of Computational. This subclass association type is implicit this means that if "a" is an instance of A and A is a subclass of B then "a" is also an instance of B[75]. Figure 6.2 shows the appearance of the Topic Map before conversion.

Design considerations

In the conversion of the Topic map of Chemistry into an Ontology the set of criteria outlined by Gruber in 1993 served as a guide[84,85]. The ontology should be enriched with the following elements compared to the existing Topic Map

1. Higher levels of conception of descriptive vocabulary.

2. Deeper semantics for class/subclass and cross-class relationships;

   In the generation of the ontology adding classes and subclasses alone does not provide deep enough semantics. Relationships between classes have to be explicitly defined.

3. Ability to express such concepts and relationships in a description language; A formal description language based on Description Logic (DL) provides the ability to reason over descriptions[83].

4. Reusability and "share-ability" of the ontology.

6.2 Building the chemical ontology

The Protégé editor and OWL DL language were used to build the Chemical ontology, as the full expressivity of the OWL language was not needed in the case of our ontology. OWL-DL was used because it is much more expressive than Owl Lite and is based on Description Logics. The protégé OWL plug-in doesn't discriminate between OWL-Lite and OWL-Full[31]. OWL ontologies

consist of Individuals, Properties and classes (concepts) which roughly corresponds to Protégé instances, slots and classes. A Top-down approach was used to build the Chemistry concept model. This approach was a natural choice as in the Topic Map state a structure was already in place. Consequently the Topic Map hierarchy was adopted whenever possible, and only a few new terms were added to the list of key words previously generated by the domain experts.

The Topics within the Topic map (figure 6.1) were directly modelled as Classes, for example the Topic Chemistry was represented as a class named ChemistryTopic. Classes are a concrete representation of concepts, and in OWL classes are interpreted as sets that contain individuals[31]. The Topics within the Topic Map were represented as classes because they corresponded to naturally occurring Topics in the domain of chemistry. Classes were organised into a superclass-subclass hierarchy which is also known as a taxonomy. In most cases the existing hierarchy was kept in place, thus ChemistryTopic was made a superclass of ComputationalChemistryTopic and ComputationalChemistry Topic was made a subclass of super class ChemistryTopic. This means that all individuals of ComputationalChemistryTopic are a kind of ChemistryTopic, and all members of the class ComputationalChemistryTopic are a kind of Chemistry

Topic. There were no instances created in the ontology because the Topics themselves were modelled as classes.

Properties allow general facts about the members of classes and specific facts about individuals to be asserted and also represent relationships between two individuals[31]. There are two main types of properties, object properties and data type properties. Datatype properties are relations between instances of classes and RDF literals and XML Schema datatypes. Object properties are relations between instances of two classes[31]. There were no individuals created as the chemical ontology was solely made up of classes, as a result there were no properties between individuals.

The only relationships existing between classes and subclasses were strictly "is-a" kind of relationships. The final ontology is shown in Figure 6.4. This final structure leaves the essential flexibility of the Topic Map vocabulary intact and can simply be enriched by adding more sub or super classes at nodes and improving properties between classes and individuals. Moreover the structure of the ontology reflects the underlying simplicity of the Topic Map, and it can be exported into OWL, XML or RDF formats which will facilitate the manipulation and retrieval of data.

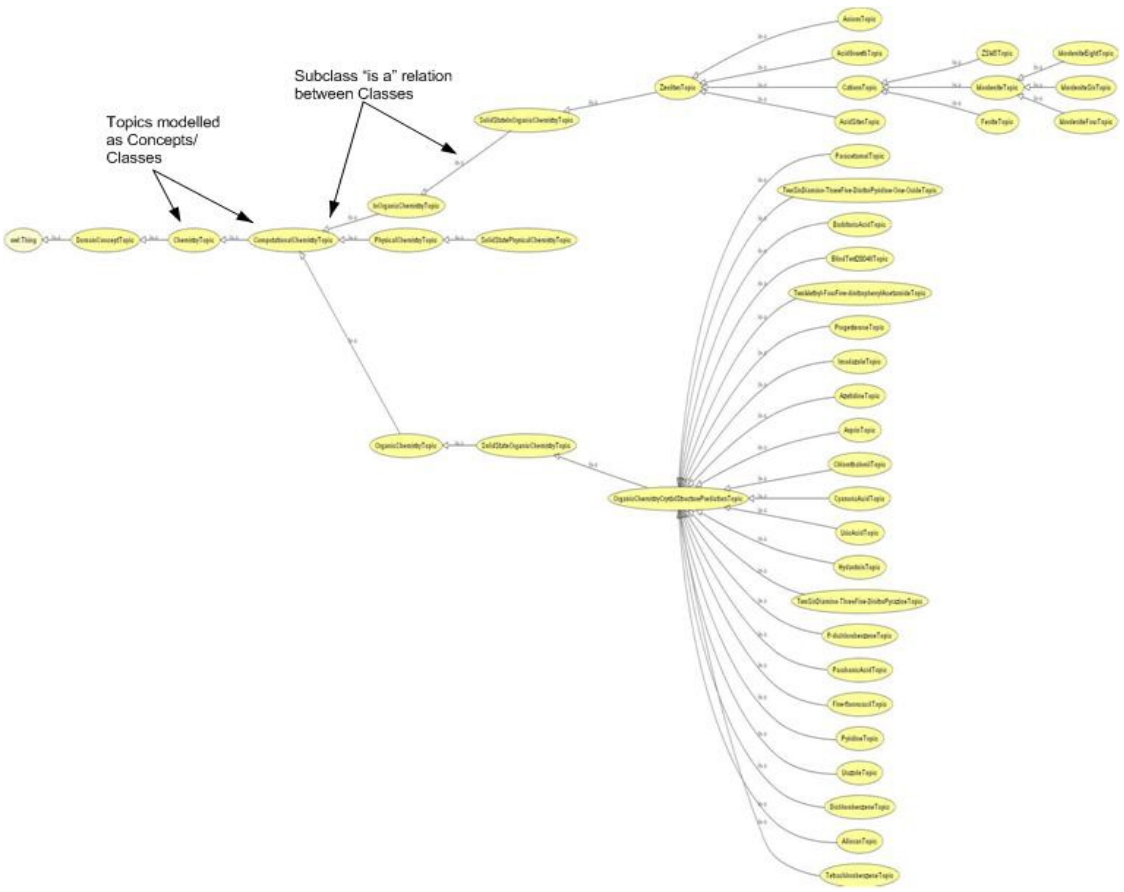

Fig 6.4 The Chemical Ontology. An OWLViz graphical representation of the chemical ontology after conversion from the topic Map.

6.3 Problems encountered

The main problems encountered during the conversion process were how to model the topics in an ontological way, naming and creating classes and extracting relations between classes.

One of the main problems encountered when constructing the ontology was on how the topics should be modelled in the ontology. Deciding whether something is a concept of an instance is difficult, and often depends on the application[30]. For example, if CrystalStucturePrediction is a concept would Pyridine be an instance of that concept? It could be argued that Pyridine is a concept representing different instances of pyridine data involving different aspects of Crystal structure prediction.

To model the Topics it was imperative that their meanings be understood and clearly defined, in the Topic Map each of the topics e.g. Chemistry was defined as a topic or subject matter and had no relation to the meaning of the word Chemistry. If the Topic Map hierarchy and topic names were adopted fully, subclass relationships in the ontology would not have been correct or made sense. For instance in the Topic map the topics Pyridine and Aspirin are subclasses of the topic Crystal structure prediction. However if Pyridine and Aspirin are made subclasses of Crystal Structure prediction in an ontology, this does not make sense. This is because Pyridine and Aspirin are not kinds of Crystal Structure Prediction and Crystal Structure Prediction is not a kind of Pyridine or Aspirin. To solve this problem either a more explicit ontology had to be developed or class names had to be changed. As an explicit model of topics was required, each of the classes was renamed as a topic e.g.

ChemistryTopic. This enabled the "is-a" kind of relationship to be used between classes.

If Pyridine and Aspirin classes are renamed as Pyridine Crystal Structure Prediction Topic and Aspirin Crystal Structure-Prediction Topic they are now kinds of Crystal Structure Prediction Topics.

It is difficult to decide how to efficiently extract concrete relations for concepts, and. relation extraction was another problem encountered during the building process. The solution was to make all the concepts topics, and to rename each of the classes as topics. This allowed the "is-a" relation to be used, and such relations were defined manually or inducted from the hierarchical structure of the Topic Map.

The next problem was on how to construct the hierarchy. The main problem was whether to assume the Topic Map hierarchy in the ontology or to create a whole new hierarchy. The Topic Map had been designed with domain experts needs coming first, and with key words or topics and hierarchy being previously approved by chemists. As a result the topic map hierarchy was kept to as much as possible with the addition of only a few super and subclasses.

Compared to the original Topic Map of Chemistry, the major difference between the two models lies in the values added through deeper semantics in

describing Topics both conceptually and relationally. Topic Maps may relate terms in a controlled vocabulary via parent-child and associative relationships, but do not contain explicit grammar rules to constrain the use of controlled vocabulary terms, to model something meaningful within a domain of interest. Domain knowledge of the chemical ontology can be modelled implicitly or explicitly however, because the data portal would be used initially by scientists at CCLRC, and as the system was designed to be used by domain experts it is subject to a varying degree of implicit assumptions. Even though the ontology constructed was a very basic, it possesses deeper semantics than the Topic Map and is reusable and sharable as it has been constructed in OWL, a W3C standard. Additionally, the ontology's structure leaves the essential flexibility of the TopicMap® vocabulary intact.

6.4 What is the added value of an ontology compared to a Topic Map?

In the CCLRC Data Portal, the ontology will be used to annotate and classify studies, and ontologies bring added value by providing a rich representation of vocabulary. For example in the Topic Map, topics are simply labelled as Chemistry or Crystal Structure Prediction however both users and annotators have no indication of what this actually means or the information it contains. The ontology provides users with more valuable information, for example the OrganicCrystalStructureTopic class can have information on experimental

data, journals and collaborative efforts. An ontology would enrich the representation vocabulary for such information.

For many people, an explicit hierarchy needs to be included in an ontology for something to be called an ontology, however there are some who consider controlled vocabularies, glossaries and thesauri to be ontologies[82]. Ontologies provide an explicit hierarchy which other classification systems like topic maps, catalogues, glossaries, and thesauri do not provide. Strict subclass hierarchies are necessary for exploitation of inheritance[29]. Without true subclass (or true "is a") relationships, certain kinds of deductive uses of ontologies become problematic[29]. For example, a "CrystalStructurePrediction" property might be restricted to have a filler that is a number (or a number in a certain range) and a "hasBeenPredicted" property may be restricted to have fillers that are crystal structures that have already been predicted. The following shows the importance of supporting strict "is a" or subclass relationships. For instance if "AcidGrowth" were a subclass of "CrystalStructure" it would inherit the property "hasBeenPredicted" and the value of the restriction material that was stated. This would not be correct as Acid growth is not a crystal structure which has been predicted. Hence ontologies add value to existing vocabularies such as top maps, thesauri or taxonomies through additional information that defines how objects can be classified and related to one another.

Ontologies permit the use of a formal description language based on description logic (DL) like OWL DL. DL languages are used to describe situations using various kinds of individuals (as instance in a collection of objects), related by roles (as relation), and grouped into concepts (a primitive specified elsewhere)[83]. Formal description languages have the ability to reason over descriptions, and this is extremely important in the long term maintenance of ontologies.

Topic Maps have no reasoning capabilities. However OWL DL is a formal description language recommended by the W3C that possesses reasoning and expressive power. Interoperability with current Semantic Web standards allows easy integration of other ontologies, and if a more complex heavy weight ontology is being built the need might arise to use branches from other existing ontologies. Therefore ontologies add value to existing systems like topic maps through formal description languages which provide reasoning capabilities. Through the use of web standards like OWL, ontologies promote share-ability and re-usability of knowledge via intergration of whole or parts of existing ontologies.

To conclude, thesauri and controlled vocabularies are not explicit enough to model heavyweight ontologies, and Topic maps may be flexibile however they lack formal semantics. Ontologies have higher levels of conception of

descriptive vocabulary, deeper semantics, the ability to express concepts and relationships in a formal description language, and provide re-usability and sharability of knowledge in heterogeneous systems like the semantic web. Conversion of a Topic Map to an ontology can be done by making knowledge explicit (i.e. useful and understandable) outside of the domain of a small expert group. This widens the narrow field of the original context and forms a potential model for conversion and expression of existing Topic Map based systems.

CHAPTER 7

CONCLUSION

In this project, methodologies, languages and tools were surveyed and compared. Languages and tools were weighed against each other with the aid of an evaluation framework. This framework enabled the choice of a tool and language to be used in the conversion of the Topic Map of Chemistry into an Ontology. After looking at a number of tools, languages and methodologies the following conclusions were drawn;

- Currently there are very few tools to support any of the methodologies described, apart from METHONTOLOGY (WebODE) and On-to-knowledge (OntoEdit). Due to the lack of support for most of the existing methodologies, no methodology was chosen to be used in the construction of the Chemical ontology.

- There are many similar ontology editors, each with their own strengths and weaknesses. In building an ontology, the choice of editor is dependent on the type of ontology being built either light weight or heavy weight.

- Heavy weight ontologies require a robust tool that does not only address design and implementation, but is scalable (i.e. can be used to

build very large ontologies consisting of up to 10,000 frames) and has inference services like consistency checking. WebODE, Ontolingua and Protégé are suitable for building heavy and light weight ontologies. However SWOOP, DAG-Edit, OntoEdit (free) and OiLed are suitable for building light weight ontologies. Out of all the lightweight editors SWOOP is the best for building lightweight ontologies.

- Out of all the tools evaluated Protégé was chosen as the best tool for building the chemical ontology because of the following;

    1. Addresses more aspects of the ontology building life cycle than it's counterparts like versioning.
    2. It's highly flexible nature with an ever expanding list of plug-ins.
    3. It can be used for building both heavy and light weight ontologies.
    4. Protégé turned out to be the easiest editor to use and had the most help systems associated with it.
    5. Out of all the tools it possessed the best Graphical user interface and was easiest to learn how to use and remember.
    6. Also because Protégé is scalable (has been used before to build and efficiently manage ontologies consisting of a hundred

thousand frames) and interoperable with current W3C standards OWL, XML and RDF.

7. And finally as Protégé has a proven track record and is arguably the most widely used ontology editor with an organized user community (mailing lists, workshops). Protégé is used in numerous projects from different domains including medicine, law and engineering.

- In choosing a language, it was observed that before building an ontology it is important to decide what the application requires in terms of inference and expressiveness. OWL-DL was chosen as the language in which to implement the ontology, because of its expressivity and inferencing power as well as unique features like versioning and cardinality constraints.

- Due to their lack of formal semantics controlled vocabularies, thesauri, SKOS and Topic maps are not suitable for building heavy-weight ontologies. There is still a huge lack of fully integrated environments which include facilities to manage deployment and maintain ontologies. Ontology management systems are suitable for building ontologies however they are costly and difficult to use although they address more aspects of the ontology life cycle than basic editors.

- The great advantage of topic maps lies in their flexibility in the modelling of almost anything; however this is a double edged sword as there are no predefined guidelines or schemas existing to help model the many possible structures. The Conversion of a Topic Map into an ontology is difficult because there are no existing standards for this process, therefore it is very much of an ad-hoc process. Additionally the conversion of Topics into concepts or individuals and relation extraction between concepts is based on the ontologist or domain expert's judgement.

- Importing and Exporting into different formats like OWL, XML and RDF is very difficult across all editors. This is not a straightforward process because most editors load ontologies in project format and with each of the tools there is not much help available on how to load ontologies in different formats therefore a lot of time is wasted in learning how to use the tools. Graphical editors like Growl and OntoTrack are not suitable for editing heavyweight ontologies, although they can be used to edit small lightweight ones. Growl and OntoTrack can be used to graphically represent ontologies and are comparable to OWL VIz. Growl is a better graphical editor than

OntoTrack since it is easier to use and has a nicer graphical user interface.

In all building an ontology is a difficult process and the right combination of editor and language should be chosen for building a heavyweight ontology based on a number of factors. These factors include collaboration and distributed workforce support, interoperability, scale, versioning, security, maintenance, ease of use of the editor, diverse user support for naïve and experienced users, extensibility, expressive power of the language and finally inference services.

It is worth noting that very many publications at the moment focus on ontology editing and building, however few focus on the issue of deployment and long term management and maintenance of ontologies.

7.1 Future work

The main aim of the conversion of the Topic Map of Chemistry was to create an explicit ontology that could be used as an indexing system in the CCLRC Data Portal. At the moment the Chemical ontology is very basic and more lightweight and further extension could not be completed due to time constraints. Therefore there are future plans to develop this ontology further to create a more explicit Chemical ontology through the provision of deeper

semantics. Additionally ontological templates for representing different domains in science will be developed, using experience from the extension of the Chemical ontology. Eventually once completed, the Chemical ontology will be implemented in the CCLRC Data Portal, and maintained by domain experts, for the purposes of enriched representation and sharing of knowledge, and intelligent information retrieval.

References

[1] B. Chandrasekaran, J.R.J., V. Richard Benjamins (1999) What are Ontologies, and Why do we need Them? IEEE Intelligent Systems 14, 20-26.

[2] M. F. Lopez, A.G. Perex., J.P. Sierra, A. P. Sierra (1999) Building a Chemical Ontology Using Methontology and the Ontology Design Environment.
IEEE Intelligent Systems &their application 14, 37-46.

[3] Honderich (1995) The Oxford Companion to Philosophy, pp. 1009 Oxford University Press, Oxford.

[4] O. Corcho, M. Fernandez-Lopez, A. Gomez-Perez (2003) Methodologies, tools and languages for building ontologies. Where is their meeting point?
Data and Knowledge Engineering 46, 41-64.

[5] T.R. Gruber (1992) ONTOLINGUA. A Mechanism to Support Portable Ontologies. Knowledge Systems Laboratory, Stanford University,

[6] V. Devedzic (2002) Understanding Ontological Engineering. Communications of ACM 45, 136–144

[7] G. Drinkwater, S.S. (2004) The CCLRC Data Portal. In: All Hands Meeting, Nottingham

[8] G. Drinkwater, S.Sufi, A. Manandhar, R. Tyer, K. O'Neill, M. Doherty, M. Williams, A. Woolf (2003) CCLRC Data Portal Architecture. Proceedings of UK e-Science All Hands Meeting

[9] Gruber, T.R. (1994) Toward principles for the design of ontologies used for knowledge sharing? In: International workshop on Formal Ontology (Guarino, I.R.P.N., ed.), Padova Italy.

[10] A. Maedche, S. Staab (2001) Ontology Learning for the Semantic Web. IEEE Intelligent Systems 16, 1-18.

[11] D.L. McGuinness, (1998) Ontological Issues for Knowledge-Enhanced Search. In: Frontiers in Artificial Intelligence and Applications, IOS-Press, Washington D.C.

[12] A. Dogac, G. Laleci, Y. Kabak, I. Cingil (2002) Exploiting Web Service Semantics: Taxonomies vs. Ontologies. Software Research and Development Center. IEEE Data Engineering Bulletin.

[13] A. Cregan, (2005) Building Topic Maps in OWL-DL. In: Conference of IDEAllianc Extreme Markup Languages 2005: Proceedings, Montréal.

[14] R. Neches, R. Fikes., T. Finin, T.R. Gruber, R. Patil, T. Senator, W.R. Swartout (1991) Enabling technology for knowledge sharing. AI Magazine 12, 36–56.

[15] T.R. Gruber, (1993) A translation approach to portable ontologies Knowledge Acquisition 5, 199-220.

[16] A.J. Duineveld, R.Stoter., M.R. Weiden, B. Kenepa, and V.R. Benjamins. (1999) Wondertools? A comparative study of ontological engineering tools. . In: n Proceedings of the 12 th International Workshop on Knowledge Acquisition, Modeling and Mangement (KAW'99, Kluwer Academic Publishers, Banff, Canada.

[17] R. Studer, V.R.Benjamins., D. Fensel (1998) Knowledge engineering: principles and methods. Data and Knowledge Engineering 25, 161–197.

[18] N. Guarino, M.C., P. Giaretta (1995) Towards very large knowledge bases. Knowledge Building and Knowledge Sharing. In: Ontologies and knowledge bases: towards a terminological clarification, in:N. Mars (Ed.) (Mars, N.J.I., ed.), IOS Press, Amsterdam.

[19] A. T. Schreiber, B.Wielinga., W. Jansweijer (1995) The KACTUS view on the 'O' word. University of Amsterdam, The Netherlands

[20] B. Swartout,.R. Patil, K. Knight, T. Russ, (1997) Toward Distributed Use of Large-Scale Ontologies. In: AAAI Symposium on Ontological Engineering, Stanford.

[21] D. M. Sánchez, J.M.C.E. Marcos (2003) On models and ontologies. Departamento de Informática, Rey Juan Carlos University, Madrid, Spain.

[22] A. Gangemi, S.G., P. Mika, Daniel Oberle, S. Lamparter, M. Sabou, S. Staab, D. Vrandecic, Core Ontology of Software Components, Core Ontology of Services. http://cos.ontoware.org/

[23] A. Gomez-Perez, M.Fernandez-Lopez., O. Corcho-Garcia (2004) Ontological Engineering, pp. 9-17, Springer-Verlag London Limited.

[24] N. Guarino, (1998) Formal Ontology and Information Systems. In: Formal Ontology in Information Systems (FOIS'98), pp. 3-15, Trento, Italy.

[25] K. Mahesh, S.N. (1995) A Situated Ontology for Practical NLP. In: Proc. Workshop on Basic Ontological Issues in Knowledge Sharing, . In: International Joint Conference on Artificial Intelligence (IJCAI-95), pp. 19-20, Montreal, Canada.

[26] C. Masolo, A. Gangemi , N. Guarino , A. Oltramari, L. Schneider (2002) The Wonder-Web Library of Foundational Ontologies and the DOLCE ontology. Laboratory for Applied Ontology. http://citeseer.ist.psu.edu/cache/papers/cs/26864/http:zSzzSzwww.ladseb.pd.cnr.itzSzinforzSzOntologyzSzPaperszSzWonderWebD17V2.0.pdf/masolo02wonderweb.pdf

[27] S. Borgo, A.Gomez-Perez., N. Guarino, C. Masolo, A. Oltramari (2002) Ontology Roadmap. ISTC-CNR, Padova, Italy.

[28] A. Maedche, S.Staab (2001) Ontology Learning for the Semantic Web. In: IEEE Intelligent Systems. University of Kalsruhe Kalsruhe, Germany.

[29] D.L. Mc Guinness, (2002.) Ontologies Come of Age. Knowledge Systems Laboratory, Stanford University.

[30] R.Stevens, C.Goble, S. Bechhofer (2001) Ontology-based Knowledge Representation for Bioinformatics. University of Manchester.

[31] M. Horridge, H. Knublauch, A. Rector, R. Stevens, C. Wroe (2004) A Practical Guide To Building OWL Ontologies Using The. Protege-OWL Plugin and
CO-ODE Tools. Edition 1.0.

[32] T. Berners-Lee, J.Hendler, O. Lassila, E. Meaning, (2001) The Semantic Web. Scientific American. http://cpdp.uab.es/documents/docencia/casanovas_pompeu/semantic_web.pdf

[33] A. Farquhar, R.Fikes., J. Rice (1997) The Ontolingua Server: a Tool for Collaborative Ontology Construction. International Journal of Human-Computer Studies 46, 1-19.

[34] P. Lambrix, M.Habbouche, M.Perez (2003) Evaluation of ontology development tools for bioinformatics., Department of Computer and Information Science, Linkopings university, Linkoping, Sweden.

[35] F. Lopez, A.Fomez-Perez. (1999) Overview of methodologies for building ontologies. In: IJCAI-99 Workshop on Ontologies and Problem-Solving

[36] D.B. Lenat, R.V.Guha, ed. (1989) Building Large Knowledge-Based Systems: Representation and Inference in the Cyc Project, Addison-Wesley Longman Publishing Co., Inc

[37] M. Uschold, M.K. (1999) Towards a Methodology for Building Ontologies. In: IJCAI99 Workshop on Ontologies and Problem-Solving Methods Stockholm.

[38] M. Gruuninger, M.S.Fox (1995.) Methodology for the design and evaluation of ontologies, in: Workshop on Basic Ontological Issues in Knowledge Sharing. Montreal.

[39] A. Bernaras, I. Laresgoiti, J. Corera (1996) Building and reusing ontologies for electrical network applications. In: Proceedings of European Conference on Artificial Intelligence (ECAI) p. 298–302. Budapest, Hungary.

[40] A. Goomez-P eerez, M.F.-L., A. de Vicente (1996) Towards a Method to Conceptualize Domain Ontologies. In: ECAI96 Workshop on Ontological Engineering, Budapest, pp. 41–51.

[41] S. Staab, H.P.S., R. Studer, Y. Sure (2001) Knowledge processes and ontologies. IEEE Intelligent Systems 16 26–34.

[42] O. Corcho, A.Gomez-Perez (2000) A Roadmap to Ontology Specification Languages In: Lecture Notes in Computer Science, pp. 80, Springer-Verlag GmbH, France.

[43] T. Bray, J.Paoli., C. M. Sperberg-McQueen, E. Maler. (2000) Extensible Markup Language Extensible Markup Language (XML) 1.0, second edition. W3C Recommendation. http://www.w3.org/TR/REC-xml

[44] (2005) The World Wide Web Consortium (W3C)

[45] S. Staab, A.Maedche (2001) Knowledge portals: Ontologies at work. University of Karlsruche, Karlsruche, Germany

[46] P. Spyns, D. Oberle, R. Volz , J. Zheng, M. Jarrar, Y. Sure , R. Studer , R. Meersman (2003). OntoWeb- A Semantic Web Community Portal. Lecture Notes in Computer Science Publisher: Springer-Verlag GmbH ,ISSN: 0302-9743 , Vol 2569 / 2002, pp. 189 - 200

[47] S. Staab, J.A., S. Decker, M. Erdmann, A. Hotho, A. Maedche, H. Schnurr, R. Studer, Y. Sure (2000) Semantic community Web portals. Computer Networks, Volume 33, 473-491.

[48] N. F. Noy, D.L. McGuinness, (2001 (2002) Predicting How Ontologies for the Semantic Web Will Evolve. . Communications of the ACM 45, 48 - 54

[49] A. Maedche, S.S., N. Stojanovic, R. Studer (2001) SEAL - A Framework for Developing SEmantic portALs. In: In Proceedings of the 18th British National, Conference on Databases., Oxford.

[50] H. Lausen, M.S., R. L. Hernández, Y. Ding, S. Han, D. Fensel (2003) Semantic Web Portals – State of the Art Survey. Next Web Generation Group, IFI – Institute for Computer Science, University of Innsbruck.

[51] M. Ushold, R.M.U., M. King,(1995) Towards a Methodology for Building Ontologies. Presented at: \Workshop on Basic Ontological Issues in Knowledge Sharing"; held inconjunction with IJCAI-95Articial Intelligence Applications Institute, University of Edinburgh

[52] S. Staab, R.Studer (2003) On-To-Knowledge Methodology In: Handbook on Ontolgies, pp. 117-132, Springer.

[53] D. Fensel, O.L., F van Harmelen, I Horrocks, J. Hendler, D.L. McGuinness (2000) The Semantic Web and Its Languages. IEEE Intelligent Systems 15, 67–73.

[54] M. Fernandez-Lopez (2002) Onto Web; A survery on Ontology Tools. In: Ontology-based information exchange for knowledge management and electronic commerce IST-2000-29243, Deliverable 1.4

[55] S. Decker, S.M., F. Van Harmelen, D. Fensel, M.Klein, J. Broekstra, M. Erdmann, I. Horrocks (2000) The Semantic Web: The Roles of XML and RDF
IEEE Internet Computing Volume 4, 63 - 74.

[56] S. Luke, J.Hendler (2000) Coping with Changing Ontologies in a Distributed Environment. Department of Computer Science, University of Maryland, Maryland.
http://www.cs.umd.edu/projects/plus/SHOE/pubs/shoe-aaai99.pdf

[57] I. Horrocks, D.Fensel., F. Harmelen, S. Decker, M. Erdmann, M. Klein (2000) OIL in a Nutshell. In: ECAI Workshop on Application of Ontologies and PSMs, Berlin.

[58] F. van Harmelen, P. Patel, I. Horrocks (2001) Reference Description of the DAML & OIL Ontology Markup Language.

[59] D. L. McGuinness, F. Harmelen (2004) OWL Web Ontology Language Overview W3C Recommendation.

[60] J. Domingue (1998) Tadzebao and Webonto: Discussing, Browsing and Editing Ontologies on the Web. In: Proc.11th Knowledge Acquisition Workshop (KAW98) (1998, ed.), Banff, Canada.

[61] J.C. Arpirez, O.Corcho, M. Fernaandez-Loopez, A. Goomez-P eerez (2001) WebODE: a scalable ontological engineering workbench. In: First International Conference on Knowledge Capture (KCAPÕ01), pp. pp 6–13, ACM Press, Victoria.

[62] S. Bechhofer, I.H., C. Goble, R. Stevens (2001) OilEd: a reason-able ontology editor for the Semantic Web. In: Joint German/Austrian conference on Artificial Intelligence, Lecture Notes in Artificial Intelligence, pp. 396–408, Springer, Berlin.

[63] Y. Sure, M.E., J. Angele, S. Staab, R. Studer, D. Wenke (2002) OntoEdit: collaborative ontology engineering for the semantic web. In: First International Semantic Web Conference (ISW), Lecture Notes in Computer Science,, pp. 221–235, Springer, Berlin.

[64] D. L. McGuinness, R.F., J. Rice, S. Wilder (2000) The Chimaera Ontology Environment. . In: Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence, pp. 1123 - 1124, Austin, Texas.

[65] N.F. Noy, R.W.Fergerson, M.A. Musen (2000) The knowledge model of protege-2000: combining interoperability and flexibility. In: 12th International Conference in Knowledge Engineering and Knowledge Management (EKAWÕ00), Lecture Notes in Artificial Intelligence (Springer, ed.), pp. 72, Berlin.

[66] A. Kalyanpur, E.S., B. Parsia, J. Hendler (2004) Hypermedia Inspired Ontology Engineering Environment: SWOOP. In: International Semantic Web Conference (ISWC), pp. 7-11, Hiroshima, Japan.

[67] Krivov, S. Growl, http://www.uvm.edu/~skrivov/growl/

[68] T Liebig, O Noppens (2004).OntoTrack: Fast Browsing and Easy Editing of Large Ontologies. Proceedings of the 3rd International Semantic Web Conference.

[69] Manchester, T.U.O. (2004) OWLViz – A visualisation plugin for the Protégé OWL Plugin.
http://www.coode.org/downloads/owlviz/OWLVizGuide.pdf

[70] J. Z. Pan, I.Horrocks (2003) RDFS (FA): A DL-ised sub-language of RDFS
In: Proc. of the 2003 Description Logic Workshop.

[71] I. Horrocks, D.F., J. Broekstra, M. Crubezy, S. Decker, M. Erdmann, W. Grosso, C. Goble, F. Van Harmelen, M. Klein, M. Musen, S. Staab, and R. Studer (2000) The ontology interchange language oil: The grease between ontologies. Dep. of Computer Science, Univ. of Manchester.

[72] Rath., H.H. (2004) Emplolis GmbH. Topic Maps are Emerging why should I care? XML http://www.xml.com/

[73] R. McEntire, P.K., N. Abernthy,D. Benton, G. Helt, M. DeJongh, R.Kent, A Kosky, S. Lewis, D. Hodnett, E.Neumann, F.Olken, D.Pathak,P.Tarczy-Hornoch,L Toldo,T. Topaloglou. . Ammerican Association for Artificial intelligence. 1999 (1999) An Evaluation of Ontology Exchange Languages for Bioinformatics. In: Ammerican Association for Artificial intelligence.

.

[74] X. Su, L. Ilebrekke. (2003) A Comparative Study of Ontology Languages and Tools. Advanced Information Systems Engineering: 14th International Conference, CAiSE 2002 Toronto, Canada, 27-31

[75] Pepper, S. (2000) The TAO of Topic Maps. In: AS Ontopia - Proceedings of XML Europe, Paris, France. http://www.ontopia.net/topicmaps/materials/tao.html

[76] MetaModel.com (2002) What are the differences between a vocabulary, a taxonomy, a thesaurus, an ontology, and a meta-model? http://www.metamodel.com/article.php?story=20030115211223271

.

[77] A. Cregan, (2005) Extreme markup Languages, School of Computer Science and Engineering. http://www.mulberrytech.com/Extreme/Proceedings/xslfo-pdf/2005/Cregan01/EML2005Cregan01.pdf

[78] A Miles, N Rogers, D. Beckett (2004) Simple Knowledge Organisation System (SKOS) W3C Working Draft http://www.w3.org/2004/02/skos/core/#specs

[79] S. El-Ansary , A.R. (2001) A Distributed Environment for Development and Deployment of Ontologies of Knowledge- Based Systems. In: 3rd International Conference on Information Reuse and Integration(IRI-2001), Las Vegas, Nevada.

[80] Technologies, A.E. (2003) Ontology Management System. http://www.alphaworks.ibm.com/tech/snobase/

[81] IBM alphaWorks (2003) Snobase Ontology ManagementSystem. http://xml.coverpages.org/ni2003-11-03-a.html

[82] R. J. Brachman (1983) What ISA Is and Isn't: An Analysis of Taxonomic Links in Semantic Networks, IEEE Computer 16, 30-6.

[83] R.J.Brachman, D.L.McGuinness., P.F Patel-Schneider, A. Borgida (1999) Reducing' CLASSIC to practice: knowledge representation theory meets reality. Artificial Intelligence 114, 203-237.

[84] T.R. Gruber, (1993a) A translation approach to portable ontology specifications." Knowledge acquisition. 5, 199-220.

[85] T.R. Gruber (1993b) Toward principles for the design of ontologies used for knowledge sharing

APPENDIX