A Comparative Study Ontology Building Tools for Semantic Web Applications

Bhaskar Kapoor[1] and Savita Sharma[2]

[1]Department of Information Technology, MAIT, New Delhi INDIA

bhaskarkapoor@gmail.com

[2]Department of Computer Science Engineering, MAIT, New Delhi INDIA

savita.tanu@gmail.com

ABSTRACT

Ontologies have recently received popularity in the area of knowledge management and knowledge sharing, especially after the evolution of the Semantic Web and its supporting technologies. An ontology defines the terms and concepts (meaning) used to describe and represent an area of knowledge.The aim of this paper is to identify all possible existing ontologies and ontology management tools (Protégé 3.4, Apollo, IsaViz & SWOOP) that are freely available and review them in terms of: a) interoperability, b) openness, c) easiness to update and maintain, d) market status and penetration. The results of the review in ontologies are analyzed for each application area, such as transport, tourism, personal services, health and social services, natural languages and other HCI-related domains. Ontology Building/Management Tools are used by different groups of people for performing diverse tasks. Although each tool provides different functionalities, most of the users just use only one, because they are not able to interchange their ontologies from one tool to another. In addition, we considered the compatibility of different ontologies with different development and management tools. The paper is also concerns the detection of commonalities and differences between the examined ontologies, both on the same domain (application area) and among different domains.

KEYWORDS

Semantic/intelligent Web, Ontologies, Ontology building tools, Protégé 3.4, IsaViz, Apollo, SWOOP

1. INTRODUCTION

Semantic Web [1] is intended to guide the current web to a place where it is more useful for human consumption. It contributes several mechanisms that can be used to classify information and characterize its context for intelligently retrieving information on web. This is mainly done using knowledge representation languages that create explicitly domain conceptualizations, such as ontologies [2],[3]. This conceptualization consists of a set of concepts, their definition and the relationships between them. In recent years, much progress has been made in developing ideas and tools to enable the growth of ontologies. Here, Ontology is widely viewed as the backbone to support various types of information management including information retrieval, storage, and sharing on web.

The development of ontologies demands the use of various software tools [4]. A range of open-source and commercial tools are available which assist in the development of various ontologies called Ontology Editors. These tools can be applied to several stages of the ontology life cycle including the creation, implementation, and maintenance of ontologies. This paper emphasis on the role of ontology editor tools for the ontology construction and presents a case study.

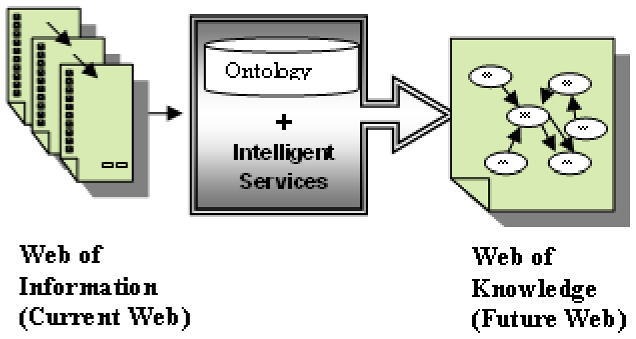


Figure1: Realization of Current Web to Future Web [1]

Ontologies are becoming the corner stone of the Semantic Web.Ontologies aim at capturing domain knowledge in a generic way and provide a commonly agreed understanding of a domain. They are shared conceptualizations of a domain and they possibly include the representations of these conceptualizations [5]. They are used to facilitate efficient exchange of information among people, now used for communication among software agents.Ontologies are independent from the applications that use them. This leads to easier software and knowledge maintenance, and contributes to the semantic interoperability between applications.Today a variety of developing environments exist for building ontologies like Protégé 3.4, IsaViz, Apollo, and SWOOP[6],[7],[8],[9].

Protégé 3.4 [6] is a knowledge based ontology editor providing graphical user interface. It is chosen because it provides better flexibility for meta-modeling, enables the construction of domain ontologies; customize data entry forms to enter data. It is typically targeted at the knowledge engineering and conceptual modeling without knowing or thinking about syntax of output language.

IsaViz [7] is a visual environment for browsing and authoring RDF models as graphs. This tool is offered by W3C Consortium. IsaViz [41] was developed by Emmanuel Pietriga.IsaViz imports RDF/XML and N-Triples, and exports RDF/XML, N-Triples, Portable Network Graphics (PNG) and Scalable Vector Graphics (SVG). Therefore, it is possible to import ontologies to other editors, for instance, Protégé or OilEd. The IsaViz environment is composed of four main windows: the IsaViz RDF Editor window, the Graph window, the Definition window and the Attribute window.

Apollo [8] is a user-friendly knowledge modeling application. The modeling is based around the basic primitives, such as classes, instances, functions, relations etc. Internal model is build as a frame system according to the internal model of the OKBC protocol. Apollo’s class system is modeled according to the OKBC. The knowledge base consists of ontology’s that are hierarchically organized. Ontology can inherit other ontology’s and then use classes of inherited ontology’s as its own. Every ontology inherits at least one ontology – a default ontology, which contains all primitive classes: Boolean, integer, float, string, list etc. Class contains slots of two types: non template and template slots.

SWOOP [9] is a Web-based OWL ontology editor and browser. SWOOP contains OWL validation and offers various OWL presentation syntax views. It has reasoning support and provides a multiple ontology environment. Ontologies can be compared, edited and merged. Different ontologies can be compared against their Description Logic-based definitions, associated properties and instances. SWOOP's interface has hyperlinked capabilities so that navigation can be simple and easy. SWOOP does not follow a methodology for ontology construction.

2. METHOD FOR BUILDING ONTOLOGY

This section presents, in direct chronological order, the most well known approaches for building ontologies [10] from scratch, as well as reusing ontologies that are stored in ontology libraries. First the main set of criteria used to compare different approaches of this type is presented. Then, a brief description of each approach is provided, presenting who has elaborated it and the proposed steps and activities

There is no one correct methodology for developing ontologies (See the appendix 'A' for comparison of all methods). Developing ontology is usually an iterative process. We can start with a rough first pass at the ontology and then revise and refine the evolving ontology . Ontology is a model of a real domain in the world and the concepts in the ontology must reflect this reality. After defining an initial version of the ontology, we can evaluate and debug it by using it in applications or problem-solving methods or by discussing it with experts in the field. As a result, we will almost certainly need to revise the initial ontology. This process of iterative design will likely continue through the entire lifecycle of the ontology. Developing an Ontology may include-

Developing an Ontology may include:

- Selection of Domain and Scope
- Consider Reuse
- Find out Important Terms
- Defining Classes and Class Hierarchy
- Defining Properties of Classes and Constraints
- Create Instances of classes

To construct an ontology one must have an ontology specification language, of which there are several to choose. Among many ontology languages, the Web Ontology Language (OWL) is the widely accepted as standard for representing and sharing knowledge in the Semantic Web context.[11] OWL is based on Resource Description Framework (RDF) and the DARPA Agent Markup Language (DAML) [12].OWL would use the RDF meaning of classes and properties (rdfs: Class, rdfs: subClassOf, etc.) and would add some very powerful modeling primitives to extend the expressiveness. OWL also provides an owl: imports construct which syntactically includes the complete referenced ontology into the importing ontology [13],[14].This construct does not allow partial reuse but can only handle complete ontologies.When starting out on an ontology project, the first and reasonable reaction is to find a suitable ontology software editor. These tools can help acquire, organize, and visualize the domain knowledge before and during the building of a formal ontology.Ontologies on the Web require more expressiveness. Classes are the focus of most ontologies. Classes describe concepts in the domain. Slots describe properties of classes and instancesDeveloping ontology includes [15]:

1. Defining classes in the ontology.
2. Arranging the classes in a taxonomic (subclass–super class) hierarchy.
3. Defining slots and describing allowed values for these slots.
4. Filling in the values for slots for instances.

3. Evaluation Framework for Methodologies and Methods

The set of criteria used to compare different approaches is based on the framework which adapts in [16] which adapts the IEEE 1075-1995 standard for software development process [17].such process is broken down in other processes (management processes, development-oriented processes, etc.). Processes are made by activities. For each activity of the framework, we set up whether it is proposed or not, and if it is described in detail. The following kinds of processes are distinguished:

- ***Project management processes.*** They create the framework for the project and ensure the right level of management throughout the entire product life cycle. Activities related to project initiation (participants, scheduling, etc.), project monitoring and control, and ontology quality management belong to this group of processes.

- ***Ontology development-oriented processes.*** Produce, install, operate and maintain the ontology and retire it from use. They are divided into three groups:

  -Pre-development processes. They are performed prior to the actual ontology development. They involve activities related to the study of the ontology installation environment, and to feasibility studies.

  -Development processes. These are the required processes for building the ontology. They include: requirements, which are comprised of iterative activities directed towards developing the ontology requirements specification; design process, the goal of which is to develop a coherent and well-organised representation of the ontology that meets the requirements specification; and implementation process, which transforms the design representation of an ontology into an implementation language.

  -Post-development processes. They are related to the installation, operation, support, maintenance and retirement of an ontology. They are performed after the ontology construction.

- ***Integral processes.*** These processes are needed to successfully complete ontology project activities. They ensure the completion and quality of project functions. They are performed at the same time as ontology development-oriented processes and include activities that do not output ontology, but are absolutely necessary to obtain a successful system.

At present the construction of ontologies is very much an art rather than a science [18]. The attempt is to formalize the ad-hoc process consists of the some basic steps [19].To supports methodology and to guide users step by step through the ontology engineering process an effective tool is desired. Along with the development of the methodology we therefore extended the core functionalities of Protégé.

4. Ontology Building Tools

In this section, with reference to a Survey[1] published on XML site that covers Software Tools that have Ontology editing capabilities and are in use today. These ontology building tools (I.e. Protégé 3.4, IsaViz, SWOOP and Apollo) may be useful for building ontology schemas (terminological component) alone or together with instance data. Concise descriptions of each software tool were compiled and then reviewed by the organization currently providing the software for commercial, open, or restricted distribution. The descriptions are factored into a dozen different categories covering important functions and features of the software. These categories are summarizing the results.

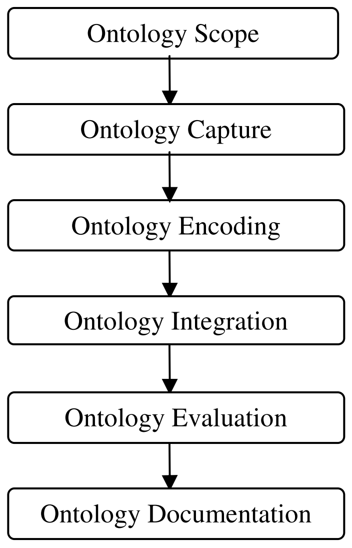


Figure 2: Ontology Construction Methodology

We have used only Four “popular and accepted” ontology authoring tools (Apollo, Protégé 3.4, IsaViz and SWOOP), taking into consideration the advantages of these tools.Tools that provide support for the different phases of the ontology engineering process are referred to as ontology building tools. These tools are used for building a new ontology either from scratch or by reusing existing ontologies, which usually supports editing, browsing, documentation, export and import from different formats, views; libraries and they may have attached inference engines, etc. [20].

The ontology editors are tools that allow users to visually manipulate, inspect, browse and code ontologies and support in this way the ontology development and maintenance task [21]. In this section, we will provide a broad overview of some of the available ontology editor tools with a brief description of each tool, presenting the group that has developed it, its main features and functionalities, its URL etc.

4.1 PROTÉGÉ 3.4

Protégé [6] is an ontology and knowledge base editor produced by Stanford University. Protégé is a tool that enables the construction of domain ontologies, customized data entry forms to enter data. Protégé allows the definition of classes, class hierarchies, variables, variable-value restrictions, and the relationships between classes and the properties of these relationships. Protégé is free and can be downloaded from http://protégé.stanford.edu [6]. Protégé comes with visualization packages such as OntoViz, EZPal, etc.; all of these help the user visualize ontologies with the help of diagrams. Stanford University is doing a magnificent job of continually improving Protégé. As part of its last update, Protégé now includes an interface for SWRL (Semantic Web Rule Language), which sits on top of OWL to do math, temporal reasoning, and adds Prolog-type reasoning rules. Stanford has a tutorial that covers the basics of using Protégé with the OWL plug-in.

The strength of Protégé is that it supports at the same time tool builders, knowledge engineers and domain specialists. This is the main difference with existing tools, which are typically targeted at the knowledge engineer and lack flexibility for meta-modeling. This latter feature makes it easier to adapt Protégé to new requirements and/or changes in the model structure. When starting out on an ontology project, the first and reasonable reaction is to find a suitable ontology software editor [4]. These tools can help acquire, organize, and visualize the domain knowledge before and during the building of a formal ontology.

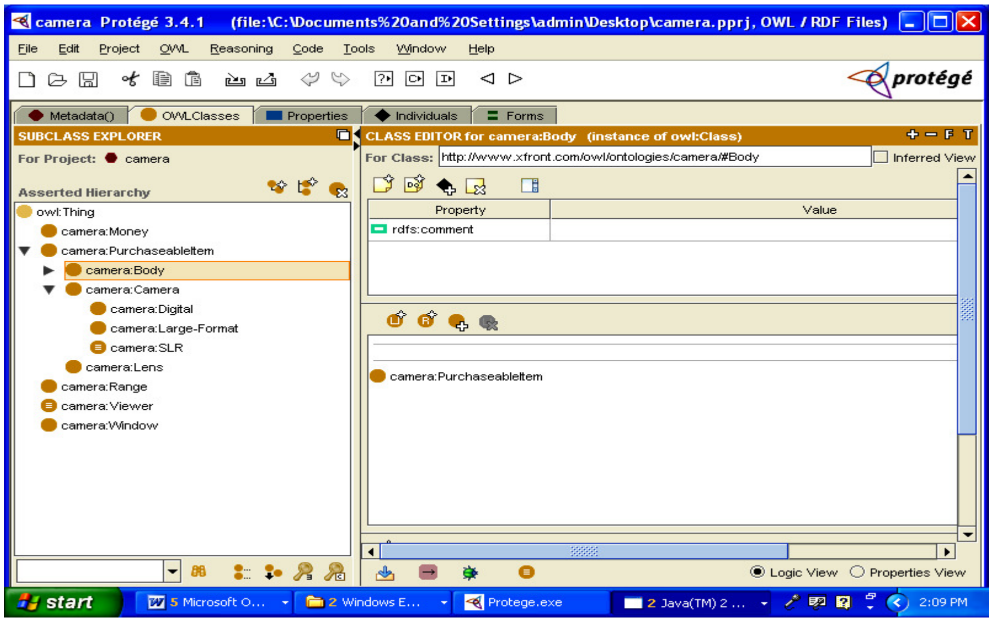

camera Protégé 3.4.1 (file:\C:\Documents%20and%20Settings\admin\Desktop\camera.pprj, OWL / RDF Files)
File Edit Project OWL Reasoning Code Tools Window Help
Metadata() OWLClasses Properties Individuals Forms
SUBCLASS EXPLORER
For Project: camera
Asserted Hierarchy
owl:Thing
camera:Money
camera:PurchaseableItem
camera:Body
camera:Camera
camera:Digital
camera:Large-Format
camera:SLR
camera:Lens
camera:Range
camera:Viewer
camera:Window
CLASS EDITOR for camera:Body (instance of owl:Class)
For Class: http://www.xfront.com/owl/ontologies/camera/#Body
Inferred View
Property
Value
rdfs:comment
camera:PurchaseableItem
Logic View
Properties View

Figure 3: Protégé 3.4 screenshot

From the existing tools Protégé 3.4 is chosen because it enables the construction of domain ontologies, customized data entry forms to enter data. Protégé allows the definition of classes, class hierarchies, variables, variable-value restrictions, and the relationships between classes and the properties of these relationships.

4.2 ISAVIZ

IsaViz is a visual environment for browsing and authoring RDF models as graphs. This tool is offered by W3C Consortium. IsaViz [7] was developed by Emmanuel Pietriga.The first version was developed in collaboration with Xerox Research Centre Europe which also contributed with XVTM, the ancestor of ZVTM (Zoomable Visual Transformation Machine) upon which IsaViz is built. As of October 2004, further developments are handled by INRIA Futurs project In Situ.

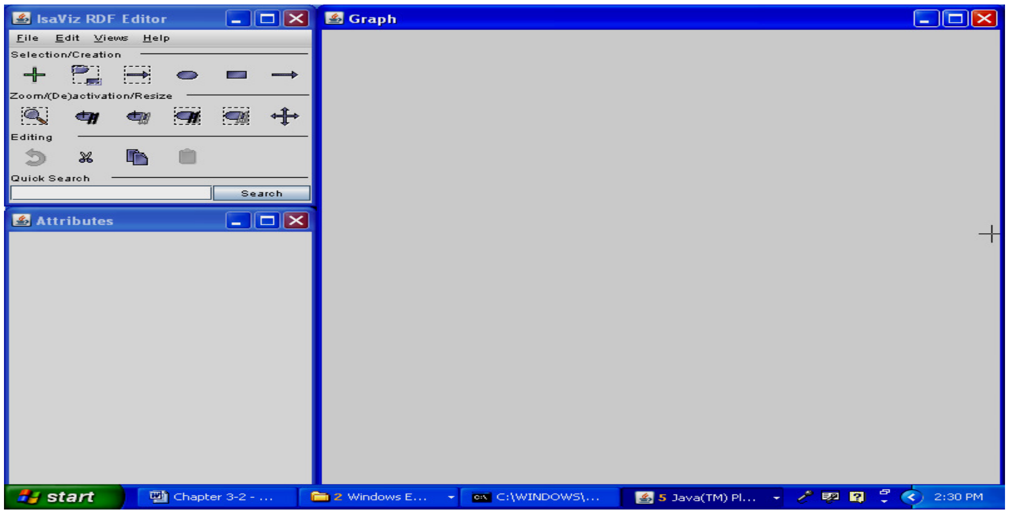


Figure 4: Screen shot of IsaViz Tool

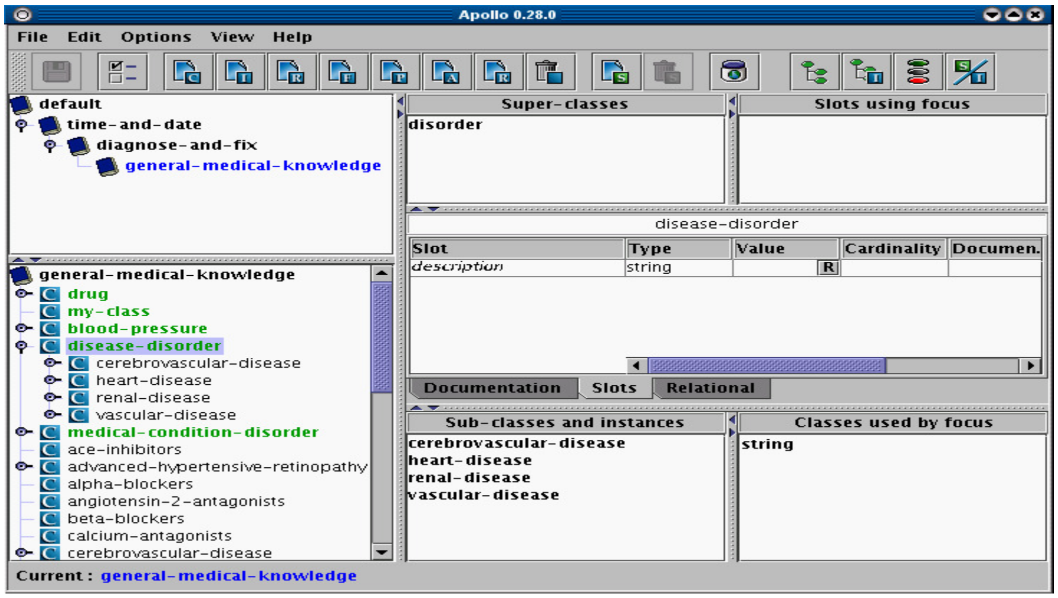
IsaViz also includes software developed by HP Labs (Jena 2 Semantic Web Toolkit), the Apache Software Foundation (Xerces Java 2), and makes use of the GraphViz library developed by AT&T Research. IsaViz does not follow or include any methodology for building an ontology. IsaViz imports RDF/XML and N-Triples, and exports RDF/XML [13], N-Triples, Portable Network Graphics (PNG) and Scalable Vector Graphics (SVG). Therefore, it is possible to import ontologies to other editors, for instance, Protégé or OilEd. The IsaViz environment is composed of four main windows: the IsaViz RDF Editor window, the Graph window, the Definition window and the Attribute window.

4.3 APOLLO

Apollo [8] is a user-friendly knowledge modeling application. The modeling is based around the basic primitives, such as classes, instances, functions, relations etc. Internal model is build as a frame system according to the internal model of the OKBC protocol.

Apollo's class system is modeled according to the OKBC. The knowledge base consists of ontology's that are hierarchically organized. Ontology can inherit other ontology's and then use classes of inherited ontology's as its own. Every ontology inherits at least one ontology – a default ontology, which contains all primitive classes: Boolean, integer, float, string, list etc. Class contains slots of two types: non template and template slots.

Apollo currently does not support non template class slots. For each class is possible to create a number of instances. An instance inherits all slots of the class. Each slot has a set of facets.



Super-classes: disorder

disease-disorder

Slot	Type	Value	Cardinality	Documen.
description	string	R		

Sub-classes and instances: cerebrovascular-disease, heart-disease, renal-disease, vascular-disease

Classes used by focus: string

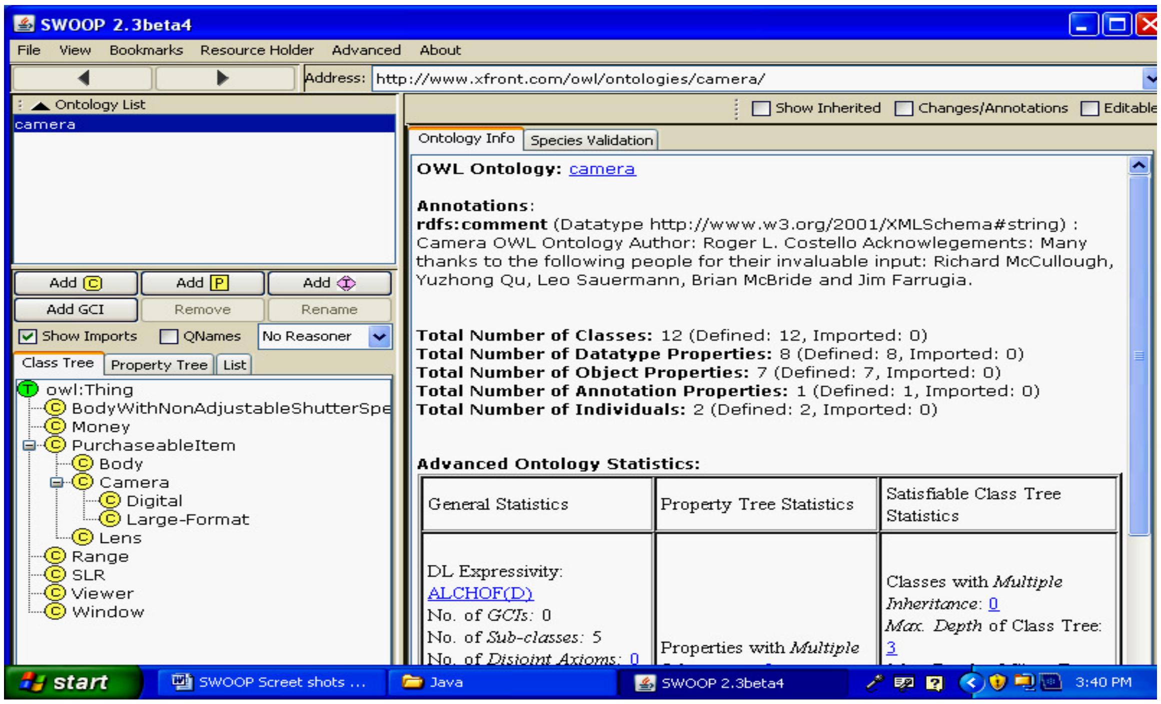
Figure 5: Main window with loaded ontology in Apollo

4.4 SWOOP

SWOOP [9] is a Web-based OWL ontology editor and browser [4]. SWOOP contains OWL validation and offers various OWL presentation syntax views. It has reasoning support and provides a multiple ontology environment. Ontologies can be compared, edited and merged. Different ontologies can be compared against their Description Logic-based definitions, associated properties and instances.

SWOOP's interface has hyperlinked capabilities so that navigation can be simple and easy. SWOOP does not follow a methodology for ontology construction.Users can reuse external ontological data [4].

This is possible either by purely linking to the external entity, or importing the entire external ontology. It is not possible to do partial imports of OWL. There are several ways to achieve this, such as a brute-force syntactic scheme to copy/paste relevant parts (axioms) of the external ontology, or a more elegant solution that involves partitioning the external ontology while preserving its semantics and then reusing (importing) only the specific partition as desired.



OWL Ontology: camera

Annotations:
rdfs:comment (Datatype http://www.w3.org/2001/XMLSchema#string) : Camera OWL Ontology Author: Roger L. Costello Acknowlegements: Many thanks to the following people for their invaluable input: Richard McCullough, Yuzhong Qu, Leo Sauermann, Brian McBride and Jim Farrugia.

Total Number of Classes: 12 (Defined: 12, Imported: 0)
Total Number of Datatype Properties: 8 (Defined: 8, Imported: 0)
Total Number of Object Properties: 7 (Defined: 7, Imported: 0)
Total Number of Annotation Properties: 1 (Defined: 1, Imported: 0)
Total Number of Individuals: 2 (Defined: 2, Imported: 0)

Advanced Ontology Statistics:

General Statistics	Property Tree Statistics	Satisfiable Class Tree Statistics
DL Expressivity: ALCHOF(D) No. of *GCIs*: 0 No. of *Sub-classes*: 5 No. of *Disjoint Axioms*: 0	Properties with *Multiple*	Classes with *Multiple Inheritance*: 0 *Max. Depth* of Class Tree: 3

Figure 6: Screenshot of SWOOP with loaded Camera.owl

It is possible to search concepts across multiple ontologies. SWOOP makes use of an ontology search algorithm, that combines keywords with DL-based in order to find related concepts. This search is made along all the ontologies stored in the SWOOP knowledge base.

5. EXPERIMENTAL RESULTS AND ANALYSIS

The framework that we have set for analyzing all the tools in chapter 3 are closely examined for the ontology development for same example Camera (camera,owl)and then we compared all the tools against the evaluation framework.

The result for comparison of tools are shown in the form of Tables which are categorized on the basis of 1)Tool architecture;in which Extensibility and ontology storage are closely examined.2) Tool's interoperability;in that Import Format,Export Format and Merging features are discussed.3) Tool's inference services which include Inference Engine,Exception Handling and Consistency Checking. 4) Tools' usability that discussed Collaboration with other tools, Ontology Library and Visualizaion. 5) Overview of Tools' versioning and collaborative work support on the basis of Versioning and collaboration.An important aspect when analyzing a tool is its tool architecture (Table 5.1). We have included information about extensibility and storage of the ontologies (databases, ACII files, etc.). From this perspective, most of the tools are moving towards extensible architectures. Storage in databases is still a weak point of ontology tools, since just a few of them i.e. Protégé 3.4 use databases for storing ontologies.

Interoperability (Table 5.2) with other ontology development tools, merging tools, information systems and databases, as well as translations to and from some ontology languages, is another important feature in order to integrate ontologies in applications. Most of the new tools export and import to ad-hoc XML and other markup languages. However, there is not a comparative study about the quality of all these translators. Moreover, there are no empirical results about the possibility of exchanging ontologies between different tools and about the loose of knowledge in the translation processes.

Before selecting a tool, it is also important to know which inference services are attached to it (Table 5.3). This includes: built-in and other inference engines, consistency checking mechanisms and exception handling, among others.Protégé-3.4 performs inference using PAL. Finally, none of the tools provide exception-handling mechanisms.

Related to the usability of tools (Table 5.4), Protégé 3.4 has the most advanced features related to the cooperative and collaborative construction of ontologies. In general, more features are required in existing tools to ensure the successful collaborative building of ontologies. Finally, other usability aspects related to help system, edition & visualization, etc., should be improved in most of the tools.

.

Feature	**Apollo**	**IsaViz**	**Protégé 3.4**	**SWOOP**
Extensibility	No	No	Via plug-ins	No
Ontology Storage	Files	Files	Files & DBMS	Files

Table 5.1: Tools' Architecture

Feature	**Apollo**	**IsaViz**	**Protégé 3.4**	**SWOOP**
Import Format	OCML	XSLT, RDF (S), OIL, DAML+OIL , OWL	XML, RDF (S), XML Schema and OWL	RDF (S), OIL, DAML,
Export Format	OCML	XSLT, RDF (S), OIL, DAML+OIL , OWL	XML, RDF (S), XML Schema, Java, html	RDF (S), OIL, DAML,
Merging	No	No	Via ANCHOR-PROMPT plug-in	No

Table 5.2: Tools' interoperability

Feature	Apollo	IsaViz	Protégé 3.4	SWOOP
Inference Engine	No	Yes	With PAL	No
Exception Handling	No	No	No	Yes
Consistency Checking	Yes	Via type inheritance and detection of cycles in hierarchies	Via plug ins like FACT and PAL	Only checks writing mistakes

Table 5.3: Tools' inference services

Feature	Apollo	IsaViz	Protégé 3.4	SWOOP
Collaboration with other tools	No	No	No	No
Ontology Library	Yes	No	Yes	No
Visualization	No	Via plug-ins like Graph Viz	No	No

Table 5.4: Tools' usability

Feature	Apollo	IsaViz	Protégé 3.4	SWOOP
Versioning	Not supported	Not supported	supported	Not supported
Collaboration	Not supported	Not supported	Not fully supported	Not fully supported

Table 5.5: Overview of Tools' versioning and collaborative work support

5. Conclusion And Further Research

For Ontology development, effective tools are a central requirement. Fortunately, software tools are already available to achieve most of the required activities of ontology development allowing us to focus specifically on the innovate requirements of ontology development within the model. Projects often involve solutions using numerous ontologies (Wine.rdf, food.owl, Companies.rdf etc.) from external sources. Sometimes there is also the need to use existing and newly developed in-house ontologies(i.e. camera.owl). For this reason it is important that the editing tools for ontology construction promote interoperability.

Ontology Editors prove an asset in the development of ontologies. The need is to identify a suitable editor for a particular domain. A theoretical attempt has been made to analyze and make a comparative analysis of the various ontology editors available and their role in ontology building and maintenance. It can be further extended to choose and make use of an ontology editor for a particular domain ontology creation.

To conclude, there are open source ontology tools (Protégé 3.4), there are ontology tools that demand learning/knowing a specific language (SWOOP) and there are ontology tools that are more graphic (IsaViz). Other tools are Web-based application (Apollo and SWOOP) or follow a methodology (Protégé 3.4 and SWOOP). Some tools only support common edition and browsing functionalities. Other tools provide ontology documentation, ontology import/export for different formats, graphical view of ontologies, ontology libraries and attached inference engines.

It is quite clear Ontology development is an ad-hoc approach. Among several viable alternatives, one need to find which one would work better for the projected task that can easily and effectively be maintained and expressed. Though foundation of ontology is logic but it is a model of reality and the concepts in the ontology must reflect this reality.. We have described a tool-assisted method for building the basis for ontologies adopted from domain analysis.

References

[1] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. Scientific American, May 2001

[2] N.F. Noy and D.L. McGuinness, “Ontology Development 101: A Guide to Creating Your First Ontology”, Technical Report KSL-01-05, Stanford Knowledge Systems Laboratory, March 2001.

[3] T. Gruber, “Ontology”, Encyclopedia of Database Systems, Springer, 2008.

[4] L. Stojanovic and B. Motik, “Ontology evolution within ontology editors” in Conference on the Evaluation of Ontology-based Tools, September 2002

[5] Gruber R. Formal Ontology in Conceptual Analysis and Knowledge Representation. Chapter "Towards principles for the design of ontologies used for knowledge sharing” in Conceptual Analysis and Knowledge Representation.1993.

[6] http://protege.stanford.edu

[7] http://www.w3.org/2001/11/IsaViz/Overview.html

[8] http://apollo.open.ac.uk/index.html

[9] Aditya Kalyanpur, Bijan Parsia, Evren Sirin, Bernardo Cuenca Grau, James A. Hendler: "Swoop: A Web Ontology Editing Browser". J. Web Sem. 4(2): 144-153 (2006).

[10] C.W. Holsapple and K.D. Joshi, "A collaborative approach to ontology design". Commun. ACM 45 (2002) pp 42–47

[11] Web Ontology Language (OWL), Available online: http://www.w3.org/2004/OWL/

[12] S. Karim, A. M. Tjoa, Towards the Use of Ontologies for Improving User Interaction for People with Special Needs, in: Computers Helping People with Special Needs, vol. 4061/2006, Springer Berlin / Heidelberg, 2006, pp. 77–84.

[13] S. Decker, S. Melnik, F. Van Harmelen, D. Fensel, M. Klein, J. Broekstra,M. Erdmann, I. Horrocks. The Semantic Web: The Roles of XML and RDF, IEEE Internet Computing, 2000, 4(5): 63-74

[14] http://www.w3.org/TR/2007/WD-xmlschema11-1-20070830/

[15] http://www.w3.org/TR/2004/REC-owl-guide-20040210/

[16] M.F. Lopez and A.G. Perez, "Overview and Analysis of Methodologies for Building Ontologies". Knowledge Engineering Review, 17(2), 129-156. (2002)

[17] IEEE Standard for Developing Software Life Cycle Processes. IEEE Computer Society. New York (USA). April 26, 1996.

[18] OWL Web Ontology Language Reference, http://www.w3.org/TR/2003/PR-owl-ref-20031215/

[19] OWL Web Ontology Language Reference, http://www.w3.org/TR/2004/REC-owl-ref-20040210/

[20] S. Karim, A. M. Tjoa, Towards the Use of Ontologies for Improving User Interaction for People with Special Needs, in: Computers Helping People with Special Needs, vol. 4061/2006, Springer Berlin / Heidelberg, 2006, pp. 77–84.

[21] M. Uschold and M. King, "Towards a Methodology for Building Ontologies". Workshop on Basic Ontological Issues in Knowledge Sharing (1995).

Bhaskar Kapoor is working as Lecturer in Maharaja Agarsen Institute of Technology, NewDelhi,He has done ME(CTA) from Delhi College of Engineering. He is Lifetime Member of ISTE.He Teaches Courses for B.Tech,BCA and MCA. His research areas of interest include Computer Graphics Vision and Multimedia , Human Computer Interaction,Sementic web Applications and ontologies.He has presented & published papers in national and international Journals and conferences.

Savita Sharma working as a lecturer in Maharaja Agrasen institute of technology, She has done M.E. (CTA) from Delhi College of Engineering, Bawana Road, Delhi. Her Area of Interest is Software Engineering, Object Oriented Programming, Semantic Web and Semantic Web Applications. She has presented and published Various Research papers in national and international conferences.

APPENDIX A

Feature			Cyc	Uschold &King's	OTK	DILIGENT	ROD
Project Management Processes	**Project Initiation**		Not Proposed	Not Proposed	Described	From OTK	Not Proposed
	Control		Not Proposed	Not Proposed	Described	From OTK	Not Proposed
	Quality Management		Not Proposed	Not Proposed	Described	From OTK	Not Proposed
Ontology Development Oriented Activities	**Pre-development processes**	**Environment Study**	Not Proposed	Not Proposed	Proposed	Described	Described
		Feasibility Study	Not Proposed	Not Proposed	Described	Described	Described
	Development Processes	**Requirements**	Not Proposed	Proposed	Described in detail	Proposed	Proposed
		Design	Not Proposed	Not Proposed	Described	Proposed	Proposed
		Implementation	Proposed	Proposed	Described	Proposed	Proposed
	Post-development Processes	**Installation**	Not Proposed	Not Proposed	Proposed	Not Proposed	Not Proposed
		Maintenance	Not Proposed	Not Proposed	Proposed	Proposed	Proposed
		Retirement	Not Proposed	Not Proposed	Not Proposed	Not Proposed	
Integral Processes	**Knowledge acquisition**		Proposed	Proposed	Described	Proposed	Described
	Verification & Validation		Not Proposed	Proposed	Proposed	Proposed	Proposed
	Ontology Configuration Management		Not Proposed	Not Proposed	Proposed	Proposed	Proposed
	Documentation		Proposed	Proposed	Described	From OTK	Proposed
	Training		Not Proposed	Not Proposed	Described	From OTK	Not Proposed