Ontology Engineering Environments

Riichiro Mizoguchi and Kouji Kozaki

The Institute of Scientific and Industrial Research, Osaka University, Osaka, Japan, miz@ei.sanken.osaka-u.ac.jp, kozaki@ei.sanken.osaka-u.ac.jp

Summary. In this chapter we discuss trends of ontology engineering environments and their characteristics through comparison between some tools. After a summarization of the recent trends of them, the authors enumerate factors which characterize those environments. Then we take up OntoEdit, Hozo, WebODE, SWOOP and Protégé, and compare them according to the factors.

1 Introduction

In order to discuss ontology engineering environments, we first need to clarify what we mean by ontology engineering. Ontology engineering is a successor of knowledge engineering which has been considered as a key technology for building knowledge-intensive systems. Although knowledge engineering has contributed to eliciting expertise, organizing it into a computational structure, and building knowledge bases, AI researchers have noticed the necessity of a more robust and theoretically sound engineering which enables knowledge sharing/reuse and formulation of the problem solving process itself. Knowledge engineering technology has thus developed into "ontology engineering" where "ontology" is the key concept to investigate.

There is another story concerning the importance of ontology engineering. It is the Semantic Web. The Semantic Web strongly requires semantic interoperability among metadata which are made using semantic tags defined in different ontologies. The issue here is to build good ontologies to come up with meaningful sets of tags which are made interoperable by ontology alignment.

Although the importance of ontology is well-understood, it is also known that building a good ontology is a hard task. This is why there have been developed some methodologies for ontology development [Chapter 6, 9] and have been built a number of ontology representation and editing tools.

This chapter discusses factors of an ontology engineering environment thorough comparison of some tools. The purpose is not to rank them but to discuss characteristics of them intended to give a guideline for users to choose an appropriate tool for their purpose. While over a hundred tools are developed to date, because of the space limitation, this chapter takes up OntoEdit [22, 23], Hozo [12, 13], WebODE [1], SWOOP [10, 11] and Protégé [17] which cover a wide range of ontology development process rather than being single-purpose tools which are covered elsewhere. After discussing the recent trends of ontology engineering tools, the authors compare some of them.

2 Trends of Ontology Engineering Environment

In the 1990s, several ontology engineering environments, such as Ontolingua Server, WebOnto, Ontosaurus, have been developed as the advancement of ontology engineering. Reference [3] surveys features of six ontology development tools at that time and found all tools did not have common ontology representation language and they were implemented based on their own ontological theories and representation models.

In the 2000s, OIL, DAML and DAML+OIL, which are the predecessors of OWL [Chapter 4], were published, and ontology engineering tools for those languages were developed. The representatives of them are OilEd, OntoEdit, Protégé and so on. After RDF(S) [Chapter 3], and OWL were published, these tools supported them as well as many other tools did. In Ontology Tools Survey[1] on XML.com, 52 tools were listed at November 06, 2002, and 93 tools were listed at September 14, 2004. At the present, the authors could find about 150 ontology development tools on the web[2] (Table 1). This shows a rapid increase of ontology engineering environments. According to the observation of these tools, the authors summarize the trends of ontology engineering environments as follows:

Domain-specific environments: In several domains, such as the Semantic Web, bioinformatics, medical science, agent technology, and software development, ontology development tools specialized to each domain are developed. For instance, OBO-Edit and DAG-Edit are ontology editors for Gene Ontology (GO) in bioinformatics, CliniClue is an ontology browsing tool for SNOMED CT in the medical domain, and Zeus is an agent development tool kit including an ontology editing tool.

Integrated environment for ontology development and use: Several tools are developed as an integrated environment which supports all processes for ontology construction to use them for development of ontology-based applications. Such environments provide users with an ontology editor, an ontology management tool, API for ontologies and so on. For instance, IODT (IBM Integrated Ontology Development Toolkit) developed by IBM includes an Eclipse-based ontology-engineering environment and OWL

[1] http://www.xml.com/pub/a/2004/07/14/onto.html

[2] Some of them are listed in the web sites such as ESW Wiki SemanticWebTools (http://esw.w3.org/topic/SemanticWebTools), Ontology Tool Survey and so on.

Table 1. List of ontology engineering environments (portion)

@@name of tools	@@web site
Domain specific environments	
OBO-Edit	http://geneontology.sourceforge.net/
DAG-Edit	http://amigo.geneontology.org/dev/
CliniClue	http://www.cliniclue.com/
Zeus	http://labs.bt.com/projects/agents/zeus/
ArgoUML	http://argouml.tigris.org/
COE	http://cmap.ihmc.us/coe/
CoGui	http://www.lirmm.fr/cogui/
Cypher	http://www.monrai.com/products/cypher
Integrated environment for ontology development and use	
KAON2	http://kaon2.semanticweb.org/
IODT	http://www.alphaworks.ibm.com/tech/semanticstk
WSMO Studio	http://www.wsmostudio.org/
Supporting system for ontology development based on various techniques	
OntoBilder(OntoX, etc)	http://iew3.technion.ac.il/OntoBuilder/
OntoGen	http://ontogen.ijs.si/
DODDLE-OWL	http://doddle-owl.sourceforge.net/
commercial tools	
OntoStudio	http://www.ontoprise.de/
IODE	http://www.ontologyworks.com/
TopBraid	http://www.topbraidcomposer.com/

The detailed list is available at http://www.hozo.jp/OntoTools/

ontology storage with an inference system based on RDBMS. WSMO Studio is Eclipse-based integrated environments to edit the Semantic Web service for WSMO[3] (Web Service Modeling Ontology). It can be used with other tools for WSMO such as a reasoner, a validator, API for web services and so on.

Supporting system for ontology development based on various techniques: Many researchers propose methods to support ontology development based on various techniques such as Natural Language Processing, Machine Learning [Chapter 11], and Search Engine. For instance, OntoBilder supports ontology development by extracting terms form web pages, OntoGen is a semi-automatic ontology construction system based on machine learning and text mining algorithms, and DODDLE-OWL supports construction of domain ontology by extracting valuable information from existing lexical databases or ontologies such as Word-Net[4]. GINO (a guided input natural language ontology editor) uses controlled natural language to edit and query ontologies.

[3] http://www.wsmo.org/

[4] http://wordnet.princeton.edu/

Increase of commercial tools: Recently, commercial tools for ontology development are increasing. About 30 tools among 150 which the authors found are commercial software. Most of them support large scale construction for development of enterprise system. OntoStudio powered by Ontoprise is a successor of OntoEdit which was developed in the early days of the Semantic Web research. It supports RDF(S), OWL as ontology language and F-Logic for the rule processing. And it can connect to databases, file-systems, applications and web-serves thorough many connecters. Ontology Works provides integrated environments for ontology construction and uses such as modeling tools, databases server and information integration software. Their central technology is the Integrated Ontology Development Environment (IODE$^{\mathrm{TM}}$). It supports construction and management of high-fidelity domain ontologies. TopBraid Composer$^{\mathrm{TM}}$ is eclipse-based platform for developing web ontologies and the Semantic Web applications. It supports the Semantic Web standards and other components for applications such as Geography and Location Mapping, Ontology-Driven Forms, UML-like Class Diagrams and so on.

3 Factors of an Ontology Engineering Environment

A comprehensive evaluation of ontology engineering tools is found in [3, 6] in which the major focus is put on static characteristics of tools. The evaluation in this chapter is done focusing on dynamic aspects of the tools. We consider that a lifecycle of ontology engineering process consists of ontology development phase, ontology use phase and ontology refinement and evaluation phase. We concentrate on characteristics of the ontology engineering process supported by the five environments. Let us enumerate factors by which an environment should be characterized for each phase.

Ontology development phase The first key task of ontology engineering is ontology construction. It includes constructions of class hierarchies, describing definitions of classes, defining relations between classes, and so on. Ontology engineering environments should support the process with the following characteristics.

Development methodology Though an ontology development requires a sophisticated development methodology, a methodology itself is not sufficient. Developers need an integrated environment which helps them build an ontology in every phase of the building process. In other words, a computer system should navigate developers in the ontology building process according to a methodology.

Collaborative development Building an ontology is often done with collaboration of multiple developers who need help in orchestration of the collaborative activities.

Compliance with an ontological theory (Theory-awareness) An ontology is not just a set of concepts but at least a "well-organized" set of concepts. An environment is expected to guide users to a well-organized ontology which largely depends on the environment's discipline of what an ontology should be rather than an ad hoc classification of concepts or a frame representation. This is why an environment needs to be compliant with a sophisticated theory of ontology.

Ontology use phase Ontology use is the other key task of ontology engineering. Users need also effective support in how to share ontology with others, how to use/reuse an ontology and how to build an instance model based on an ontology.

Compliance with WWW standard There are many languages standardized by W3C: XML, RDF(S), DAML+OIL and OWL, etc. The environment is required to be compliant with these.

Ontology/Model(instance) server Ontologies and instance models should be available through internet.

Ontology evaluation and refinement phase To construct a well-organized ontology, evaluation and refinement of the developed ontology are repeated many times. An environment should support the process.

Evaluation methodology Many theories and methods for ontology evaluation are discussed [Chapter 13]. Tools should support them.

Inference service An inference engine is used to check the consistency of ontologies/instances.

Refinement mechanism It is important to manage version of ontologies and its change histories for maintenance of the consistency of ontologies. Debugging mechanisms and suggestion for modification are also useful for refinement of ontologies.

Software level issues

Usability GUI as well as functionality is essential to the usability of the environment.

Architecture of the environment An environment should be designed in an advanced and sophisticated architecture to make it usable.

Extensibility It is good if users easily extend the environment.

4 OntoEdit

OntoEdit [22, 23], professional version, is an ontology engineering environment to support the development and maintenance of ontologies. Ontology development process in OntoEdit is based on their own methodology, On-To-Knowledge [Chapter 6] which is originally based on Common KADS methodology and consists of major three steps such as requirement specification, refinement and evaluation processes. The requirement specification consists of description of the domain and the goal of the ontology, design guidelines, available knowledge sources, potential users and use cases, and applications

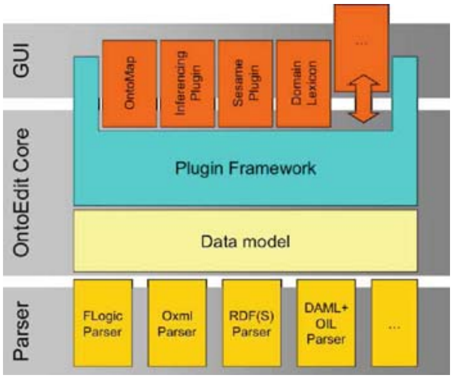

Fig. 1. Architecture of OntoEdit

supported by the ontology. The output of this phase is refined into a formal description in the next phase. Refinement is done usually collaboratively. In the evaluation phase, competency questions are used to evaluate if the ontology built can answer these questions.

Figure 1 shows the architecture of OntoEdit consisting of three layers: GUI, OntoEdit Core and Parser. It employs the plug-in architecture to make it easily extensible and customizable by the users. It is compliant with XML family standards in import and export the ontology. At the present, the technologies of OntoEdit are inherited to OntoStudio as a commercial tool. It has new features such as connectors to many kinds of resources, integrated rule management, mapping view between different ontologies and so on.

4.1 Ontology Development Phase

Requirement Specification Phase

Two tools, OntoKick and Mind2Onto, are prepared for supporting this phase of ontology capture. OntoKick is designed for computer engineers who are familiar with software development process and tries to build relevant structures for building informal ontology description by obtaining competency questions proposed in [8] which the resulting ontology and ontology-based applications have to answer. Examples of competency questions made by OntoKick include “which research groups exist at the institute?”, “which teaching courses are offered by the insti-tute?”, etc. Mind2Onto is a graphical tool for capturing informal relations between concepts. It is easy to use because it has a good visual interface and allows loose identification of relations between concepts. However, it is necessary to convert the map into a more formal organization to generate an ontology.

4.2 Ontology Evaluation and Refinement Phase

Refinement Phase [23]

This phase is for developers to use the editor to refine the ontological structure and the definition of concepts and relations. Like most of other tools, OntoEdit employs the client/server architecture where ontologies are managed in a server and multiple clients access and modify it. A sophisticated transaction control is introduced to enable concurrent development of an ontology in a collaborative manner. Because OntoEdit allows multiple users to edit the same class in an ontology at the same time, it needs a powerful lock mechanism of each class and devises Strict two Phase Locking protocol: S2PL to support arbitrary nested transactions.

Evaluation Phase

The key process in this phase is use of competency questions obtained in the first phase to see if the designed ontology satisfies the requirements. To do this, OntoEdit provides users with a function to form a set of instances and axioms used as a test set for evaluating the ontology against the competency questions. It also provides users with debugging tools for ease of identify and correct incorrect part of the ontology. It maintains the dependency between competency questions and concepts derived from them to facilitate the debugging process. This allows users to trace back to the origins of each concept. Another unique feature of this phase is that collaborative evaluation is also supported by introducing the name space so that the inference engine can process each of test sets given by multiple users. Further, it enables local evaluation corresponding to respective test sets followed by global evaluation using the combined test. Like WebODE, OntoEdit supports OntoClean [Chapter 9] methodology to build a better *is-a* hierarchy.

Inference

OntoEdit employs Ontobroker [2] and F-Logic[Chapter 2] as its inference engine. It is used to process axioms in the refinement and evaluation phases. Especially, it plays an important role in the evaluation phase because it processes competency questions to the ontology to prove that it satisfies them. It exploits the strength of F-logic in that it can express arbitrary powerful rules which quantify over the set of classes which Description logics cannot.

5 Hozo

Hozo[5] is an ontology engineering environment based on fundamental ontological theories [12, 13]. It is composed of "Ontology Editor", "Onto-Studio," "Ontology Server" and "Ontology Manager." One of the most remarkable features of Hozo is that it can deal with Role based on a sophisticated ontological theory of Role [16].

When an ontology and its instance model seriously reflects the real world, users have to be careful not to confuse the Role such as teacher, mother, fuel, etc. with other basic concepts (natural type) such as human, water, oil, etc. Let us take an example: <*teacher is-a human*>. Assume John is a teacher of a school. Given the usual semantics of *is-a*, since John is an instance of teacher then he is also an instance of human at the same time. When he quits being a teacher, he cannot be an instance of teacher so that you need to delete the instance-of link between John and teacher. However, you have to restore an instance-of link between John and human, otherwise John dies. This problem would be difficult for a model with no idea of roles to represent changes in the roles played by John (e.g., teacher, husband, patient) according to contexts or aspects.

In Hozo, three different classes are introduced to deal with the concept of role appropriately.

Role-concept A concept representing a role dependent on a context (e.g., teacher role)

Basic concept A concept which does not need other concepts for being defined (e.g., human)

Role holder An entity of a basic concept which is holding the role (e.g., teacher)

A basic concept is used as the class constraint which indicates potential players who can play the role (role concepts). Then an instance that satisfies the class constraint plays the role and becomes a role holder. Hozo supports to define such a role concept as well as a basic concept.

5.1 Ontology Development Phase

Like other editors, Ontology Editor in Hozo provides users with a graphical interface through which they can browse and modify ontologies by simple mouse operations. How to deal with "role concept" and "relation" on the basis of fundamental consideration is discussed in [12]. This interface consists of the following four parts (Fig. 2):

1. *Navigation pane* provides several functionalities for browsing the ontology such as displaying the ontology in a hierarchical structure according to

[5] http://www.hozo.jp/

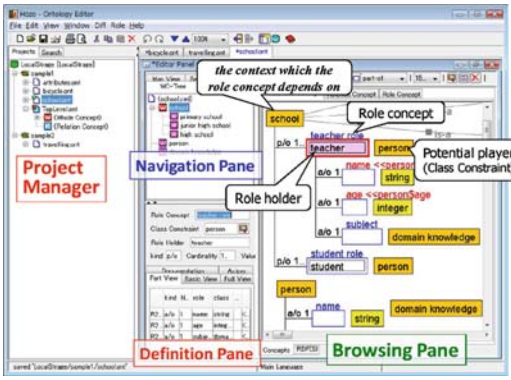

Fig. 2. GUI of ontology editor

only *is-a* relation between concepts, showing thumbnail of the ontology and searching for concepts.
2. *Browsing pane* displays the concept graphically, and the user can select concepts which he/she wants to edit.
3. *Definition pane* allows users to define and modify the selected concept in the browsing pane or in the *is-a* hierarchy browser.
4. *Project Manager* supports distributed development of ontologies.

Collaborative Development

Collaborative development of an ontology is supported in Hozo [21]. At the primitive level, the ontology server stores ontologies under version management and access control by lock/unlock mechanism. It allows users to sharing ontologies and to avoid conflict of modification by different users. Furthermore, Hozo allows users to divide an ontology into several components and manages the dependency between them to enable the concurrent development of the whole ontology. In the concurrent development, one of key issues is the maintenance of consistency among inter-dependent component ontologies. Hozo provides users with a module to maintain consistencies of the dependencies among ontologies. When a component ontology is updated, the system checks the change by comparing the modified ontology and its old version. Hozo shows users a list of changes with possible countermeasures for coping with each of the changes. These countermeasures are devised through our investigation on conceptual dependencies of ontologies and the change type of imported concepts.

5.2 Ontology Use Phase

Functionality and GUI of Hozo's instance editor is the same as the one for ontology. The consistency of all the instances with the ontology is automatically guaranteed, since a user is given valid classes and their slot value restrictions by the editor when he/she creates an instance. Inference mechanism of Hozo is not very sophisticated. Axioms are defined for each class but it works as a semantic constraint checker like WebODE. Hozo has an experience in modeling of a real-scale Oil-refinery plant with about 2000 instances including even pipes and their topological configuration which is consistent with the Oil-refinery plant ontology developed with domain experts [15]. The model as well as the ontology are served by the ontology server and can answer questions on the topological structure of the plant, the name of each device, etc. Any ontology can have multiple sets of instances which are independent of one another. The ontology server stores ontologies and instance models and serves them to clients through API. Ontology editor is also a client of the ontology server. The internal representation of Hozo is XML-based frame language and it generates RDF(S) and OWL code to export the ontology and instance.

6 WebODE

WebODE[6] [1] is a scalable and integrated workbench for ontology engineering and is considered as a Web evolution of ODE(Ontology Development Environment [4]). It supports building an ontology at the knowledge level, and translates it into different ontology languages. WebODE is designed on the basis of a general architecture shown in Fig. 3 and to cover most of the

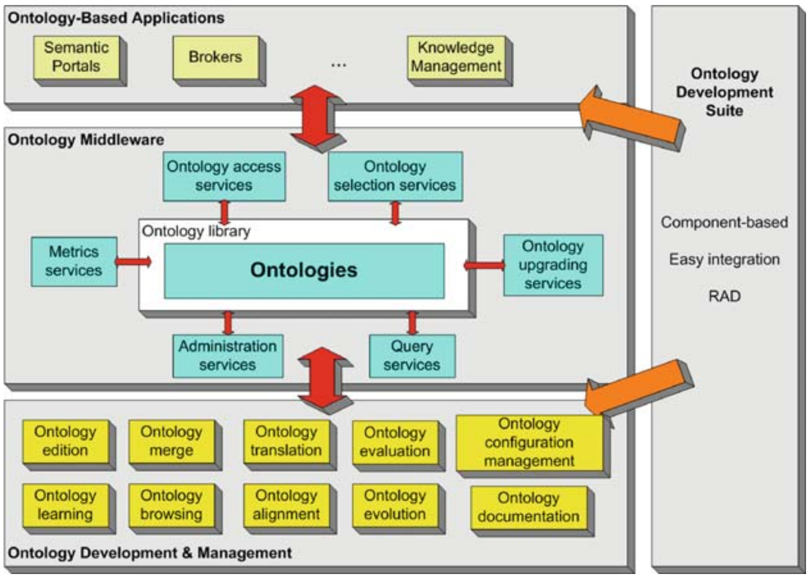

Fig. 3. Arcdhitecture of WebODE [1]

[6] http://webode.dia.fi.upm.es/WebODEWeb/index.html

processes appearing in the ontology lifecycle. WebODE is based on a client-server architecture which provides high extensibility and usability by allowing the addition of new services and the use of existing services. Ontology is stored in an SQL database to attain high performance in the case of a large ontology. It has export and import services from and into XML, and its translation services into and from various ontology specification languages such as OWL, RDF(S), OIL, DAML+OIL, UML, Prolog, X-CARIN, Jess and F-Logic. Like OntoEdit, WebODE's ontology editor allows the collaborative edition of ontologies. One of the most characteristic features of WebODE is that it is based on an ontology development methodology named METHONTOLOGY [4].

6.1 Ontology Development Phase

WebODE has ontology editing service, WAB: WebODE Axiom Builder service, inference engine service, interoperability service and ontology documentation service in this phase. The ontology editor provides users with form based and graphical user interfaces, WAB provides an easy graphical interface for defining axioms. It enables users to define an axiom by using templates given by the tool with simple mouse operations. Axioms are translated into Prolog. The inference engine is based on Prolog and OKBC protocol to make it implementation independent. Interoperability services provided by WebODE are of variety. It includes ontology access API, ontology export/import in XML-family languages, translation of classes into Java beans to enable Jess system to read them and OKBC compliance.

ODEClean [5]

Like OntoEdit, WebODE supports OntoClean methodology to build a more convincing *is-a* hierarchy. Ontology for OntoClean is composed of top level universal ontology developed by Guarino [Chapter 9], a set of meta-properties and OntoClean axioms which are translated into Prolog to be interpreted by WebODE inference engine. It is given to the ODEClean which works on the basis of it.

Collaborative Development

The collaborative editing of an ontology is supported by a mechanism that allows users to establish the type of access to the ontologies developed through the notion of groups of users. Synchronization mechanism is also introduced to enable several users to safely edit the same ontology. Ontologies are automatically documented in different formats such as HTML tables with Methontology's intermediate representations, concept taxonomies and XML.

6.2 Ontology Use Phase

To support the use process of ontology, WebODE has several functionalities. Like Hozo, it allows users to have multiple sets of instances for an ontology by introducing instance sets depending on different scenarios, and conceptual views from the same conceptual model, which allows creating and storing different parts of the ontology, highlighting and/or customizing the visualization of the ontology for each user. WebPicker is a set of wrappers to enable users to bring classification of products in the e-Commerce world into WebODE ontology. ODEMerge is a module for merging ontologies with the help of correspondence information given by the user. Methontology and ODE have been used for building many ontologies including chemical ontology [4].

WebODE is also used for developing some semantic web frameworks such as ODE SWS [7] and ODESeW. The ODE SWS is a framework for designing semantic web services at the knowledge level. It supports development of web services based on problem solving method ontology. The ODE SeW is a semantic web application framework to develop and manage web sites as a knowledge portal. In these ways, many applications have been developed using WebODE as workbench for ontology engineering.

7 SWOOP

SWOOP[7] [10,11] is an ontology browser and editor designed wholly for OWL, while many other tools (e.g., Hozo, WebODE and Protégé) support OWL as an extended feature. The architecture is based on the Model-View-Controller paradigm. SwoopModel component stores OWL ontologies loaded by a reasoner and other information related to them. They are visualized by renderers in multiple views. Controller is based on the plug-in architecture. Although the development of SWOOP is done by Mindswap project but has been terminated on August in 2006, its source code is available at URL.

7.1 Ontology Development Phase

A key feature of its design rationale is to realize user interface like the standard web browser. It consists of an address bar, history buttons (back, next), a navigation sidebar, bookmarks and so on (Fig. 4). In this GUI, URIs play a central role for understanding and constructing OWL ontologies. The users can load an OWL ontology by entering its URL in the address bar. If the ontology is importing other ontologies by owl:import property, SWOOP also loads the imported ontologies automatically. The loaded multiple ontologies are listed on the top of the navigation sidebar and their class/property hierarchies are shown at the bottom. The contents of selected ontology/entity (class,

[7] http://www.mindswap.org/2004/SWOOP/

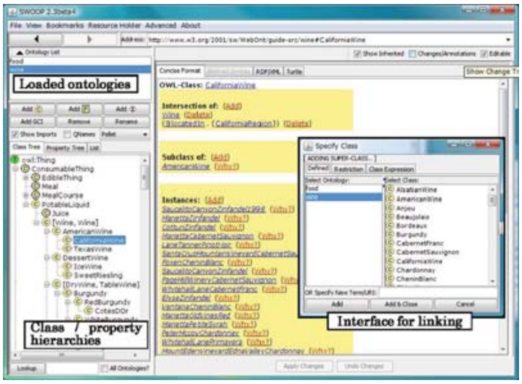

Fig. 4. Graphical User Interface of SWOOP

property and instance) are displayed on the center pane in the webpage-like format. In the pane, relationships between entities are represented by hyperlinks. It enables users to navigate the OWL ontology just like another web page. For linking entities in different ontologies, SWOOP provides a single common interface. It is displayed by clicking "Add" hyperlink in the center pane and shows the list of ontologies along with entities defined in them. Users can edit the ontology by selecting the entity to link. Through the editing process, external ontologies are modified as a local version and maintained separately. SWOOP also supports various presentation syntaxes for OWL such as RDF/XML, OWL Abstract Syntax and Turtle. Users can browse and edit[8] ontologies in these syntaxes.

Collaborative Development

For collaborative ontology development, SWOOP supports collaborative annotation and version control. The collaborative annotation is based on the standard W3C Annotea protocols. Users can share annotations about change of ontologies and discussions through a public Annotea server. The version control supports undo/redo with logging of changes and save of checkpoints. While the change logs can be used to track back the changes, the checkpoints are used as a snapshot of ontology at particular time.

7.2 Ontology Evaluation and Refinement Phase

SWOOP contains two reasoners: RDFS-like and Pellet. The former is a lightweight reasoner for RDFS, and the latter is a powerful reasoner for OWL-DL. Pellet is based on the tableaux algorithms and can be used to check inconsistencies of definition in ontologies. SWOOP provides functions

[8] Inline editing in RDF/XML and Turtle is supported by SWOOP ver.2.3.

for ontology debugging and repair using the description logic reasoner [10]. The former explains the result of reasoning to the user in a meaningful and readable manner, and the latter gives a guideline to repair the inconsistencies of ontologies.

8 Protégé

Protégé[9] [17] whose architecture is shown in Fig. 5 is strong in the use phase of ontology: Use for knowledge acquisition, merging and alignment of existing ontologies, and plug-in new functional modules to augment its usability. It has been used for many years for knowledge acquisition of domain knowledge and for domain ontology building in recent years. Its main features include:

1. Extensible knowledge model to enable users to redefine the representational primitives
2. A customizable output file format to adapt any formal language
3. A customizable user interface
4. Powerful plug-in architecture to enable integration with other applications

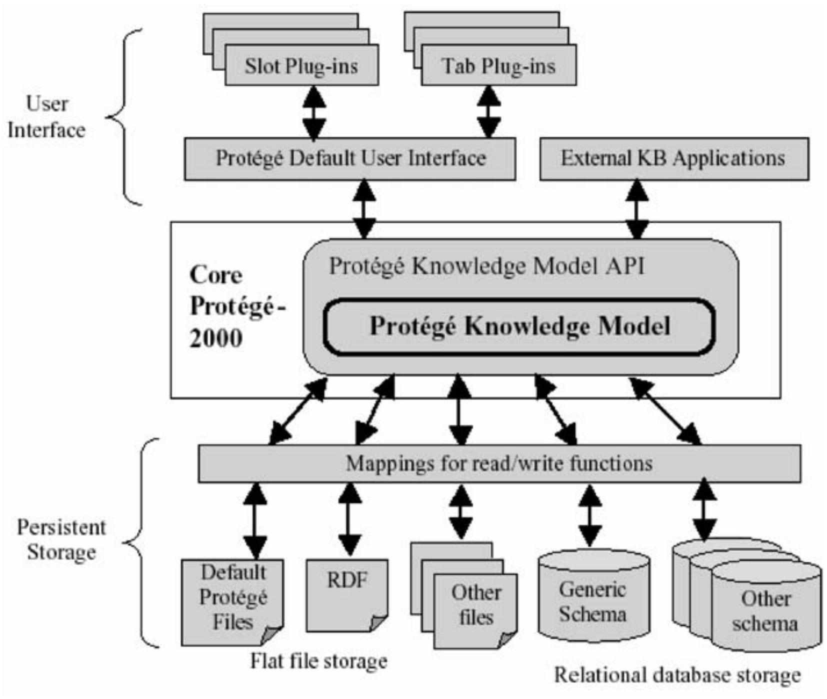

Fig. 5. Architecture of Protégé [17]

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

These features make Protégé a meta-tool for domain model building, since a user can easily adapt it to his/her own instance acquisition tool together with the customized interface. It is highly extensible thanks to its very sophisticated plug-in architecture and a Java-based API for development of knowledge based applications. Unlike the other three, Protégé assumes local installation rather than use through internet using client/server architecture. Its knowledge model is based on frame similar to other environments. Especially, the fact that Protégé generates its output in many ontology languages and its powerful customizability make it easy for users to change it to an editor of a specific language. For instance, the definition of a class of RDFS is defined as a subclass of standard class of Protégé. This "meta-tuning" can be easily done thanks to Protégé's declarative definition of all the meta-classes which play a role of a template of a class.

8.1 Ontology Development Phase

The system provides two main ways for ontology development such as Protégé-Frames and Protégé-OWL. The former supports frame-based knowledge model which is compatible to OKBC. And the latter is extensions of them using the Protégé OWL plug-in [14] for supporting OWL. It also supports to edit SWRL rules. It support owl:imports mechanism by ontology repository manager which manages URLs of imported ontologies.

Protégé has a semi-automatic tool for ontology merging and alignment named PROMPT [18]. It performs some tasks automatically and guides the user in per-forming other tasks. It also detects possible inconsistencies in the ontology, which result from the user's actions, and suggests ways to remedy them. For ontology evolution in collaborative environments [19], Protégé provides two functionalities: Change-management which stores a list of change with annotations and shows history of the change to the user, and Client-Server mode which support synchronous ontology editing by multiple users.

9 Comparison and Discussion

The five environments are compared according to the factors presented above. Table 2 summarizes the comparison.

9.1 Ontology Development Phase

Development Methodology

Philosophy of supporting ontology development is partly based on viewing an ontology as a software product. Common features of OntoEdit and WebODE

Table 2. Comparison of the five environments

	OntoEdit	Hozo	WebODE	SWOOP	Protégé
Ontology development phase					
Methodological support	On-To-Knowledge	No	METHON TOLOGY	No	No
Collaboration support	Partly	Yes	Partly	Yes	Yes
Ontological theory	Ontoclean	Role theory	Ontoclean	Implicit	Implicit
Ontology use phase					
Standards compliance	RDF(S), F-Logic	RDF(S), OWL (export only)	RDF(S),OWL F-Logic	RDF(S), OWL	RDF(S), OWL, SWRL
Ontology/ model server	High	High	High	Middle (Web server)	Middle
Ontology evaluation and refinement phase					
Evaluation methodology	OntoClean	No	OntoClean	Debugging by reasoner	No
Inference service	OntoBroker	Constraint checking	Prolog, Jess	RDFS-like, Pellet	FaCT, Jess, F-Logic...
Refinement support	Debugging tool	Change checking	ODEClean	Debugging tool	PROMPT
Software level issue					
Friendly GUI	GUI based editing	Graphically editing	GUI based editing	In line editting	GUI based editing
Architecture	Client /server	Client/ server	Client/ server	Standalone	Standa-lone
Extensibility	Plug-in	API	API/Plug-in	Plug-in	API/Plug-in

include management of the well-known steps in software development process, that is, requirement specification, conceptual design, implementation and evaluation. OntoEdit is based on the On-To-Knowledge methodology, and WebODE is based on the METHONTOLOGY. Others have no such a methodology.

Collaboration

Collaboration occurs in two different ways: (1) Construction a single ontology by different developers and (2) Construction several modularized ontologies in parallel by different developers. In the case of (1), because multiple persons might modify the same class at the same time, transaction control is one of the main issues in supporting collaboration. OntoEdit, WebODE, SWOOP and Protégé take this approach. While OntoEdit and WebODE only have some mechanisms for access management to ontologies, SWOOP and Protégé can manage histories of change with annotation for supporting collaborative construction.

On the other hand, Hozo mainly takes (2). Its main issue is to take care of the dependencies between the modularized ontologies because each developer constructs some modules under his responsibility. When building a large ontology, (2) is very useful because it allows users the concurrent development of an ontology like usual software development. To make the latter approach feasible, however, the system does need to provide developers with relevant information of changes done in other ontologies developed by others which might influence on the ontology they are developing. Hozo is designed to cope with all the possible situations developers encounter by analyzing possible patterns of influences propagated to each modularized ontology in a different module according to the type of the change. Although both approaches look different, they are complementary. The former can be incorporated in the latter. In fact, Hozo supports the former as well so that users can share a component ontology.

Theory-Awareness

Ontology building is not easy. This is partly because a good guideline is not available which people badly need when articulating the target world and organizing a taxonomic hierarchy of concepts. An ontology engineering environment has to be helpful also in this respect. WebODE and OntoEdit support Guarino's Ontoclean method. Guarino and his group have been investigating basic theories for ontology for several years and have come up with a sophisticated methodology which identifies inappropriate organization of *is-a* hierarchy of concepts. Developers who develop an ontology based on their intuition tend to misuse of *is-a* relation and to use it in more situations than are valid, which Guarino called "*is-a* overloading." Ontoclean is based on the idea of meta-property which contributes to proper categorization of concepts at the meta-level and hence to appropriate organization of *is-a* hierarchy.

OntoEdit and WebODE way of ontology cleaning can be said that post-processing way. On the contrary, Hozo tries to incorporate the fruits of ontological theories during the development process. One of the major causes of producing an inappropriate *is-a* hierarchy from Guarino's theory is lack of the concept of Role such as teacher, mother, food, etc. which has different characteristics from so-called basic concepts like human, tree, fish, etc. Ontology editor in Hozo incorporates a way of representing the concept of Role.

9.2 Ontology Use Phase

Standards Compliance

All the five have support standards ontology languages such as RDF(S) and OWL. Hozo only can export its ontology and model in RDF(S) and OWL. Protégé also supports Semantic Web Rule Language (SWRL).

Ontology/Model(Instance) Server

Hozo and WebODE has an ontology/model server which allows agents to access the ontologies and instance models through internet. OntoEdit and Protégé have an ontology server. SWOOP does not have a specific ontology server but can download ontologies in general web servers.

9.3 Ontology Evaluation and Refinement Phase

Evaluation Methodology

OntoEdit and WebODE support OntoClean methodology to build a better *is-a* hierarchy. SWOOP provides functions for ontology debugging and repair using the tableaux based reasoning. It explains the result of reasoning with a guideline to repair the inconsistencies of ontologies. Hozo and Protégé have no evaluation methodology.

Inference Service

All the five have inference mechanisms. Hozo supports only inference for constraint checking of own language.

Refinement Support

OntoEdit and WebODE have a debugging tool based on OntoClean. Hozo has a function to Check changes and suggest countermeasures for modification by comparison of ontologies. SWOOP provides a debugging tool based on reasoning and change management and a version control mechanism with logging of changes. Protégé also supports change monument and a semi-automatic ontology alignment tool named PROMPT.

9.4 Software Level Issue

Friendly GUI

All the five have sophisticated GUI such as visualization of class hierarchies and editing tool for constraints (axioms) of classes. It makes users free from coding using a complicated language. In Hozo, visualization of an ontology is default, and users can browse and edit it graphically. SWOOP supports in-line/GUI based editing functions and visualization of ontologies as just like a web page. Others provides mainly GUI based editing functions with some graphical visualization tool of ontologies. For instance, Protégé supports several ontology visualization pulig-ins such as OWL-Viz and Jambalaya.

Architecture and Extensibility

While WebODE and Hozo employ standardized API to the main ontology base, OntoEdit and SWOOP supports a plug-in architecture. Protégé provides both of API and plug-in. Both enable a module can easily added or deleted to make the environment extensible. WebODE, OntoEdit and Hozo are web-based, while Protégé is basically not. But Protégé has another mode to support server-client architecture.

10 Other Environments

In this section, other environments are summarized. We discuss four tools which have characteristic functionalities to aid user's ontology development.

10.1 OntoGen

OntoGen[10] is a system for semi-automatic ontology construction developed under SEKT project. The system has functionalities for keywords extraction form text data and suggestion using text mining and machine learning techniques. It helps users construct overview of ontologies from text documents. Some features of this tool are as follows:

Keywords extraction: In the system, two keyword extraction methods, the concept's centroid model and SVM linear model, are implemented. The extracted keywords by both methods are shown with related information about the number of related documents, average inner-cluster similarity measure, and so on.

Concept suggestion: The system suggests sub-concepts of the selected concept to the user based on two different approaches: an unsupervised approach and a supervised one. In the unsupervised approach, OntoGen supports four clustering methods: k-means, LSI, PH k-means, and a categorization according to the labels in the input data. The user can select one of the methods, and supervises the parameters for the method. Then the system suggests sub-concepts using the selected method with the parameters. The supervised approach is based on SVM active learning method. In the approach, the user enters a query the active learning system then the system asks if a particular document (instance) belongs to the selected concept and the user answers yes or no. After repetition of this learning process the system outputs most important keywords with information about positively classified into the concept.

Document management: The system manages documents related to concepts. When a new concept is added to ontology, it automatically assigns documents to it according to the similarity between documents. The system

[10] http://ontogen.ijs.si/

also has a functionality to detect if documents related to a concept belongs to its super concept. The user can know inconsistency between ontologies through the result.

10.2 CampTools Ontology Editor

CampTools ontology editor (COE) [9] is a tool for collaborative ontology development and reuse based on CampTools. CmapTools is software to construct, navigate and navigate a Concept map which is a knowledge representation model to display knowledge as a two-dimensional network of labeled nodes and links. COE supports editing, storing, and sharing Concept maps and ontologies, and the users can search for concepts and properties in them. The system also provides a cluster-based vicinity concepts view. In the view, the system shows concepts which relevant to selected concept based on multi-viewpoint clustering analysis (MVP-CA) software developed by Pragati, Inc..

10.3 OntoBilder

OntoBuilder [20] is a tool for extraction and matching of ontologies from web sources. The system extracts HTML form elements of web pages and relationships among them. A set of terms (vocabulary) associated with the extracted elements are regarded as an ontology in the system. The main feature of OntoBuilder is functionalities for ontology matching. It supports several matching algorithm such as term matching, value matching, precedence matching and so on. And the user can add another matching algorithm as plug-in.

10.4 KAON

KAON[11]: Karlsruhe Ontology and Semantic Web framework is a sophisticated plug-in framework with API and provides services for ontology and metadata management for E-Services. Its main focus is put on the enterprise application in the semantic web age. At the present, its new version, called KAON2, is available. It supports OWL-DL, SWRL and F-logic, and provides an API for ontology, an inference engine for answering SPARQL queries, a DIG interface and so on.

11 Concluding Remarks

A lot of ontology engineering environments have been developed. Although some are powerful as a software tool, but many are passive in the sense that few guidance or suggestion is made by the environment. Theory-awareness should be enriched further to make the environment more sophisticated. Especially,

[11] http://kaon.semanticweb.org/, http://kaon2.semanticweb.org/

more effective guidelines for appropriate class and relationship identification are needed. Collaboration support becomes more and more important as ontology building requirements increases. Ontology alignment is also crucial for reusing the existing ontologies and for facilitating their interoperability. Combination of the strong functions of each environment of the five would realize a novel and better environment, which suggests that we are heading right directions to go.

References

1. Corcho O, Fernandez-Lopez M, Gómez-Pérez A, Vicente O (2002) WebODE: An Integrated Workbench for Ontology Representation, Reasoning and Exchange, Proc. of EKAW2002, Springer LNAI 2473: 138–153.
2. Decker S, Erdmann M, Fensel D, Studer R (1999) Ontobroker: Ontology Based Access to Distributed and Semi-structured Information: 351–369.
3. Duineveld A, Weiden M, Kenepa B, Benjamis R (1999) WonderTools? A Comparative Study of Ontological Engineering Tools. Proc. of KAW99.
4. Fernandez-Lopez M, Gómez-Pérez A, Pazos Sierra J (1999) Building a Chemical Ontology Using Methontology and the Ontology Design Environment, IEEE Intelligent Systems, 14(1): 37–46.
5. Fernandez-Lopez M, Gómez-Pérez A, (2002) The Integration of OntoClearn in WebODE, Proc. of the 1st Workshop on Evaluation of Ontology-based Tools (EON2002): 38–52.
6. Gómez-Pérez A, Angele J, Fernandez-Lopez, et al. (2002) A Survey on Ontology Tools. OntoWeb Deliverable 1.3, Universidad Politecnia de Madrid.
7. Gómez-Pérez A, González-Cabero R, Manuel Lama (2004) ODE SWS: A Framework for Designing and Composing Semantic Web Services, IEEE Intelligent Systems 19(4): 24–31.
8. Gruninger R, Fox M (1994) The Design and Evaluation of Ontologies for Enterprise Engineering, Proc. of Comparison of implemented ontology: 105–128.
9. Hayes P, Eskridge T C, Mehrotra M, et al. (2005) COE: Tools for Collaborative Ontology Development and Reuse, Proc. of K-CAP2005.
10. Kalyanpur A, Parsia B, Sirin E, Cuenca-Grau B, Hendler J (2005) Swoop: A 'Web' Ontology Editing Browser, Journal of Web Semantics 4(2).
11. Kalyanpur A, Parsia B, Sirin E, Hendler J (2005) Debugging Unsatisfiable Classes in OWL Ontologies, Journal of Web Semantics 3(4).
12. Kozaki K, et al. (2000) Development of an Environment for Building Ontologies Which is Based on a Fundamental Consideration of "Relationship" and "Role", Proc. of the Sixth Pacific Knowledge Acquisition Workshop (PKAW2000): 205–221.
13. Kozaki K, Kitamura Y, Ikeda M, Mizoguchi R (2002) Hozo: An Environment for Building/Using Ontologies Based on a Fundamental Consideration of "Role" and "Relationship", Proc. of EKAW2002: 213–218.
14. Knublauch H, Fergerson R W, et al. (2004) The Protégé OWL Plugin: An Open Development Environment for Semantic Web Applications, Proc. of ISWC 2004: 229–243.
15. Mizoguchi R, Kozaki K, Sano T, Kitamura Y (2000) Construction and Deployment of a Plant Ontology, Proc. of EKAW2000: 113–128.

16. Mizoguchi R, et al. (2007) A Model of Roles in Ontology Development Tool: Hozo, Journal of Ontological Analysis and Conceptual Modeling, 2(2): 159–179.
17. Musen M A, Fergerson R W, Grosso W E, Crubezy M, Eriksson H, et al. (2003) The Evolution of Protégé: An Environment for Knowledge-Based Systems Development, International Journal of Human–Computer Studies, 58(1): 89–123.
18. Noy N F, Musen M A (2000) PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment, Proc. of the Seventeenth National Conference on Artificial Intelligence (AAAI-2000): 450–455.
19. Noy N, Chugh A, Liu W, Musen M (2006) A Framework for Ontology Evolution in Collaborative Environments, Proc. of ISWC2006, LNCS 4273: 544–558.
20. Roitman H, Gal A (2006) OntoBuilder: Fully Automatic Extraction and Consolidation of Ontologies from Web Sources Using Sequence Semantics. Proc. of EDBT Workshops 2006: 573–576.
21. Sunagawa E, Kozaki K et al. (2003) An Environment for Distributed Ontology Development Based on Dependency Management, Proc. of ISWC2003: 453–468.
22. Sure Y, Staab S, Angele J (2002) OntoEdit: Guiding Ontology Development by Methodology and Inferencing, Proc. of the Confederated International Conferences CoopIS, DOA and ODBASE: 1205–1222.
23. Sure Y, Staab S, Erdmann M, et al. (2002) OntoEdit: Collaborative Ontology Development for the Semantic web, Proc. of ISWC2002: 221–235.