We are IntechOpen, the world's leading publisher of Open Access books
Built by scientists, for scientists

6,900
Open access books available

186,000
International authors and editors

200M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

CLARIVATE ANALYTICS
BOOK CITATION INDEX
INDEXED

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index
in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Decision Support Tools for Ontological Engineering

Simon Suigen Guo, Christine W. Chan and Robert Harrison
Energy Informatics Laboratory,
Faculty of Engineering and Applied Science
University of Regina
Canada

1. Introduction

Development cost of knowledge-based systems is often high because knowledge bases are often constructed from scratch in a distributed and heterogeneous environment involving different locations, system types, knowledge representation mechanisms and stakeholders. Hence, it is difficult to share and re-use existing knowledge base components.

Sharing and re-using knowledge can help reduce costs and make it possible to create systems capable of more powerful problem solving. The Semantic Web (SW) is the next evolutionary step for the World Wide Web (Web). The SW aims to provide semantics to data on the Web, enabling computers to more easily share and perform problem solving on the data (Berners-Lee et al. 2001). SW technology can be used to share and re-use knowledge between KBSs in a distributed and heterogeneous environment.

Semantic data networks on the Web provide the bases of the SW, enabling knowledge that is distributed over the Web to be easily shared and processed by a machine. A semantic data structure is required for representing data or knowledge in a shareable format, and an ontology is designed to fulfill this objective. According to the World Wide Web Consortium, an ontology is "a specification of a conceptualization" (Gruber, 1993); ontologies are the vocabularies of the SW that define the concepts and relationships used to describe an area of interest. An ontology represents the definitions of concepts, axioms and facts which discribe real world phenomenon of interest. As suggested in (Gruber, 1993), the declarative formalisms and the set of objects represented as domain knowledge in a knowledge-based system (KBS) can be described by an ontology. Hence, an ontology can also become the basis for building knowledge-based systems. Ontologies implemented in XML[1] based languages, such as RDF[2] and OWL[3], enable different KBS development groups or different Semantic Web applications to share and re-use their knowledge and data.

Knowledge Acquisition (KA) is an important step for building knowledge-based systems (KBS). In the process, software tools are often used for helping knowledge engineers construct ontologies and knowledge bases more efficiently and effectively. The objective of

---

[1] Extensible Markup Language, http://www.w3.org/XML/

[2] Resource Description Framework (RDF), http://www.w3.org/RDF/

[3] OWL Web Ontology Language, http://www.w3.org/TR/owl-features/

our research is to develop a suite of tools that support knowledge visualization and management for the Semantic Web. This suite of tools is created based on the theoretical framework of the Inferential Modeling Technique, which provides support for both static and dynamic knowledge modeling. The tools can be used to support the process of KA during the building of a KBS.

Existing software tools for knowledge engineering support will be critically examined and observations drawn in the assessment will provide the basis for design of the tools we propose for supporting knowledge visualization and management.

This chapter is organized as follows. Section 2 presents background literature relevant to our research. Section 3 describes the development of tools for static and dynamic knowledge visualization and ontology management. Section 4 provides some discussion of the tools. Section 5 provides some conclusions and discussions on directions for future work.

2. Background literature

2.1 Ontology tools

Software tools support knowledge engineers in the difficult task of knowledge acquisition and ontology construction. From a survey of research work on ontology tools, we found the existing tools do not provide adequate support in five areas, which include (Harrison & Chan, 2005): (1) support for an ontological engineering methodology, (2) support for dynamic knowledge modeling, (3) support for dynamic knowledge testing, (4) support for ontology visualization, and (5) support for ontology management. They are discussed in detail as follows:

2.1.1 Lack of support for an ontological engineering methodology

Most of the examined tools were not developed based on an ontological engineering methodology, which can help expedite the ontological engineering process. The methodology provides a sequence of steps, procedure or guidelines that help developers in the process of constructing an ontology. Some examples of research works on tool support for ontological engineering methodologies include OntoEdit which supports On-To-Knowledge (Fensel et al., 2002), WebODE which supports METHONTOLOGY (WebODE), and Knowledge Modeling System (KMS) which supports the Inferential Modeling Technique (IMT) (Chan, 2004).

2.1.2 Lack of support for modeling dynamic knowledge

While a primary function of a tool that supports ontology construction is its ability to model the problem domain. Most tools only model static domain knowledge about classes and attributes but ignore dynamic knowledge, which refers to the knowledge about objectives, tasks and activities (Chan, 2004). A brief survey of the different ontological engineering support tools reveal there are diverse methods employed. For example, the method used by Protégé (Protégé), Ontolingua (Ontolingua), and OntoBuilder (OntoBuilder) for modeling ontologies is to use a hierarchical tree listing all the concepts, and input fields are provided to capture characteristics of concepts. A graphical method of modeling ontologies is employed in tools such as KAON (KANO) and Protégé OWLViz Plug-in (Protégé OWLViz), in which the representational method involves nodes to represent concepts and edges to represent relationships between concepts.

A different approach is adopted in the Unified Modeling Language (UML) tools, which are commonly used to visually describe and communicate the design of software. Due to the similarities between ontologies and object-oriented design, UML class diagrams can be used to model ontology classes and their properties, and relationships between classes (Cranefield et al., 2005). However, UML's expressiveness for modeling ontologies is limited, and Standard UML cannot express more advanced ontologies that contain descriptive logic (Djuric et al., 2005). Due to the limitations of UML, there is on-going work to develop a graphical notation for ontologies, called the Ontology Definition Metamodel (ODM) (Djuric et al., 2005).

The existing tools described can model static knowledge to varying degrees, but they often lack the facilities for modeling dynamic knowledge. For example, it is awkward to link a task to an objective in Protégé using the existing input fields. This is a weakness that our research addresses.

2.1.3 Lack of support for ontology testing

The examined tools that support ontology construction do not include facilities for supporting ontology testing. Analogous to software testing which identifies defects during development and results in a more stable software application being constructed, ontology testing can help the knowledge engineer develop a more complete model of the domain. In the ontological engineering field, ontology testing is also called ontology evaluation. According to (Gomez-Perez et al., 2005), ontology evaluation should be performed on the following items:

* Every individual definition and axiom
* Collections of definitions and axioms stated explicitly in the ontology
* Definitions imported from other ontologies
* Definitions that can be inferred from other definitions

Existing ontology testing systems, such as the OWL Unit Test Framework (OWL Unit Test Framework) and Chimaera's (McGuiness et al., 2000) test suite, evaluate the correctness and completeness of ontologies. Chimaera performs tests in the following four areas: (1) Missing argument names, documentation, constraints, etc., (2) syntactic analysis, e.g. occurrence of words, possible acronym expansion, (3) taxonomic analysis, e.g. unnecessary super classes and types, and (4) semantic evaluation, e.g. slot/type mismatch, class definition cycle, domain/range mismatch (McGuiness et al., 2000). Such testing tools are sufficient for testing static knowledge, but are not suitable for testing the interactions between behaviour and objects.

Our research work aims to fill this gap in research by addressing the difficult issue of testing behaviour or dynamic knowledge. Our approach attempts to combine unit testing techniques with the adoption of test cases in Test Driven Development (TDD) (Janzen & Saiedian, 2005). This is a useful hybrid approach for addressing the complex interactions of task behaviour and objects. The general intuition adopted from the TDD approach of testing is that it should be "done early and done often". In TDD, a test case is written first and then the actual module is written. Writing test cases first can be beneficial because instead of thinking of test cases as "how do I break something", writing test cases first make you think "what do I want the program to do". In other words, writing test cases first make you think about what functionality and objects are required. Ontology development could also benefit from this kind of approach.

2.1.4 Lack of support for ontology visualization

Most ontology editing tools lack visualization support. Often ontology visualization tools use 2 dimensional on 2D graphics for representing concepts and relationships among concepts. 2D graphics are often employed for representing static knowledge in hierarchical or frame like structures. However, 2D graphics are not adequate for representing complex and related information because there is insufficient space on a bi-dimensional plane, where complex and related visual objects tend to squeeze together or even overlap with each other. In an effort to accommodate a model of complex related objects in a 2D plane, existing visualization tools either collapse lower priority visual objects into high level objects, or implement different representations that offer multi-perspectives on an organization of knowledge elements. While these approaches solve the problem of related objects being overcrowded in a shared space, it sacrifices clarity of the display.

A popular ontology editor is Protégé, which has some degree of visualization capabilities in that it generates a tree view of classes. The Protégé's ontology visualization applications are implemented in its plug-ins; some examples of these ontology visualization plug-ins include Jambalaya (Wu & Storey, 2000) and OntoSphere (Bosca et al., 2005), which are discussed as follows.

Jambalaya is an ontology visualization tool plug-in in Protégé. It provides several viewing perspectives for the ontology model, thereby enhancing user browsing, exploring and interacting with a 2D ontology visualization. Jambalaya only visualizes the static knowledge of classes and instances of an application ontology, it does not support dynamic knowledge visualization. Furthermore, since Jambalaya is based on 2D graphics, the space it supports is insufficient for rendering complex knowledge. In its representation, text labels and symbols tend to overlap when the domain ontology is represented as a hierarchy involving many levels of concepts. This deficiency means it is difficult for users to view and understand the concepts and the relationships among concepts when the domain ontology is complex.

OntoSphere is also a Protégé plug-in. By employing 3D graphics for visualization, OntoSphere extends the volume of space available for visualizing overcrowded concepts. The 3-dimentional or 3D view is natural for humans. And its main advantage is that it allows users to manipulate the visualized knowledge elements of the application ontology by means of the actions of zooming, rotating and translating. Through physical manipulation of the concepts, the user can better understand a complex ontology. For this purpose, OntoSphere provides four scenes so that the user can observe a visualized application ontology from multiple perspectives.

However, the weakness of OntoSphere include (1) it was not developed based on any ontological engineering methodology, and (2) it does not support visualization of dynamic knowledge. Although the employment of 3D graphics enlarges the space available for rendering images, the problem of overlapping concepts and labels still exists when the application ontology is complex.

2.1.5 Lack of support for ontology management

Currently ontology development within the SW is not managed in a systematic manner. Anyone can create, reuse, and/or extend concepts in a distributed and heterogeneous environment, and different versions of an ontology can be created, which results in backwards compatibility problems. A system for ontology management is needed in order to document, track, and distribute ontologies in such an environment. Existing ontology

management frameworks provide inadequate support for replicating an ontology within the public domain and detecting when a public domain ontology has been tampered with. In addition, weaknesses of existing ontology management systems include vulnerability to malicious ontology authors, little consideration for intellectual property rights, and lack of support for ontology versioning.

2.1.6 Specific research objectives

In view of the five weaknesses discussed above, the overall objective of our research work is to construct a suite of software tools for modeling, visualizing and managing knowledge. The tools are designed so that they: (1) are derived from a theoretical basis of a knowledge modeling technique, (2) can sufficiently express the diverse types of knowledge required to model an ontology of an industrial domain, which include both static and dynamic knowledge, (3) can support visualization of both static and dynamic knowledge, (4) can support ontology management, and (5) can support testing of the developed knowledge model. In this chapter, the tools of Onto3DViz and Distributed Framework for Knowledge Evolution (DFKE) (Obst, 2006) are discussed which present new approaches for supporting ontology visualization and management. Dyna is a tool that has been developed for tackling dynamic knowledge modeling and ontology testing, and it is presented in detail in (Harrison & Chan, 2009). The two tools that will be discussed here have been developed based on the knowledge modeling method of Inferential Modeling Technique (IMT). Since the tools are based on the weak theory provided by the IMT, this knowledge modeling technique is presented next.

2.2 Inferential modeling technique

The Inferential Modeling Technique is a knowledge engineering method that supports developing a domain ontology consisting of both static and dynamic knowledge of the problem domain. Static knowledge consists of concepts, attributes, individuals and the relationships among them; dynamic knowledge includes objectives, tasks, and relationships among objectives and tasks. Static and dynamic knowledge are intertwined in that a task is a process that manipulates static knowledge to achieve an objective. The details of this modeling technique can be found in (Chan, 2004).

3. Conclusion design and implementation of the tools

The two tools that were designed and developed to address the weaknesses of ontology visualization and management include Onto3DViz and DFKE (Obst, 2006). Onto3DViz is an ontology visualization tool developed based on the IMT, which supports visualization of both static and dynamic knowledge models. Onto3DViz also supports knowledge sharing and re-use, by requiring ontology documents represented in OWL and XML as the input. DFKE is an ontology management tool which incorporates a monotonic versioning system with a peer-to-peer distributed knowledge management system. This tool was designed and a prototype of the system was developed.

3.1 Design of Onto3DViz

Onto3DViz can be seen as an extension of Dyna (Harrison & Chan, 2009) developed at the Energy Informatics Laboratory. Dyna is a dynamic knowledge modeler developed based on

the IMT, which can model the dynamic knowledge of an industrial domain. Dyna can be used in conjunction with Protégé, which models the static knowledge. When the two tools are used together, they can represent the static and dynamic knowledge of most industrial domains.
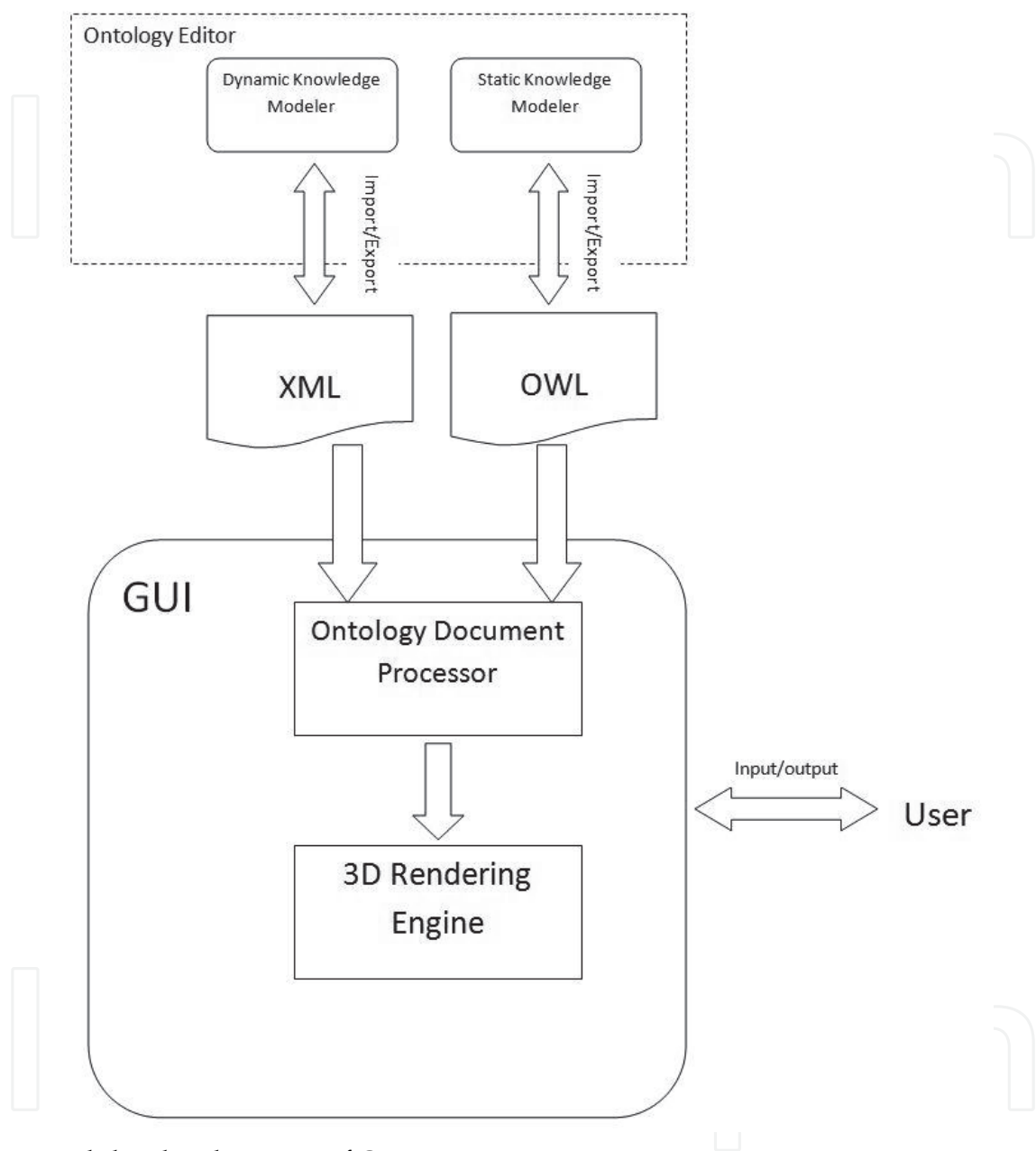

Fig. 1. High-level architecture of Onto3DViz

While Dyna (Harrison & Chan, 2009) can support ontology modeling, its capability for supporting ontology visualization is limited. Onto3DViz addresses this limitation and has been developed for visualizing an application ontology in 3D graphics. It is written in the Java™ language and its 3D visualization engine is implemented in Java 3D™. Onto3DViz differs from other ontology visualization tools in two ways: (1) it is a visualization tool developed based on the IMT (Chan, 2004), and (2) it supports visualization of both static and dynamic knowledge models specified according to the IMT. As well, Onto3DViz requires supports knowledge sharing and re-use. Unlike Dyna which has been implemented as a
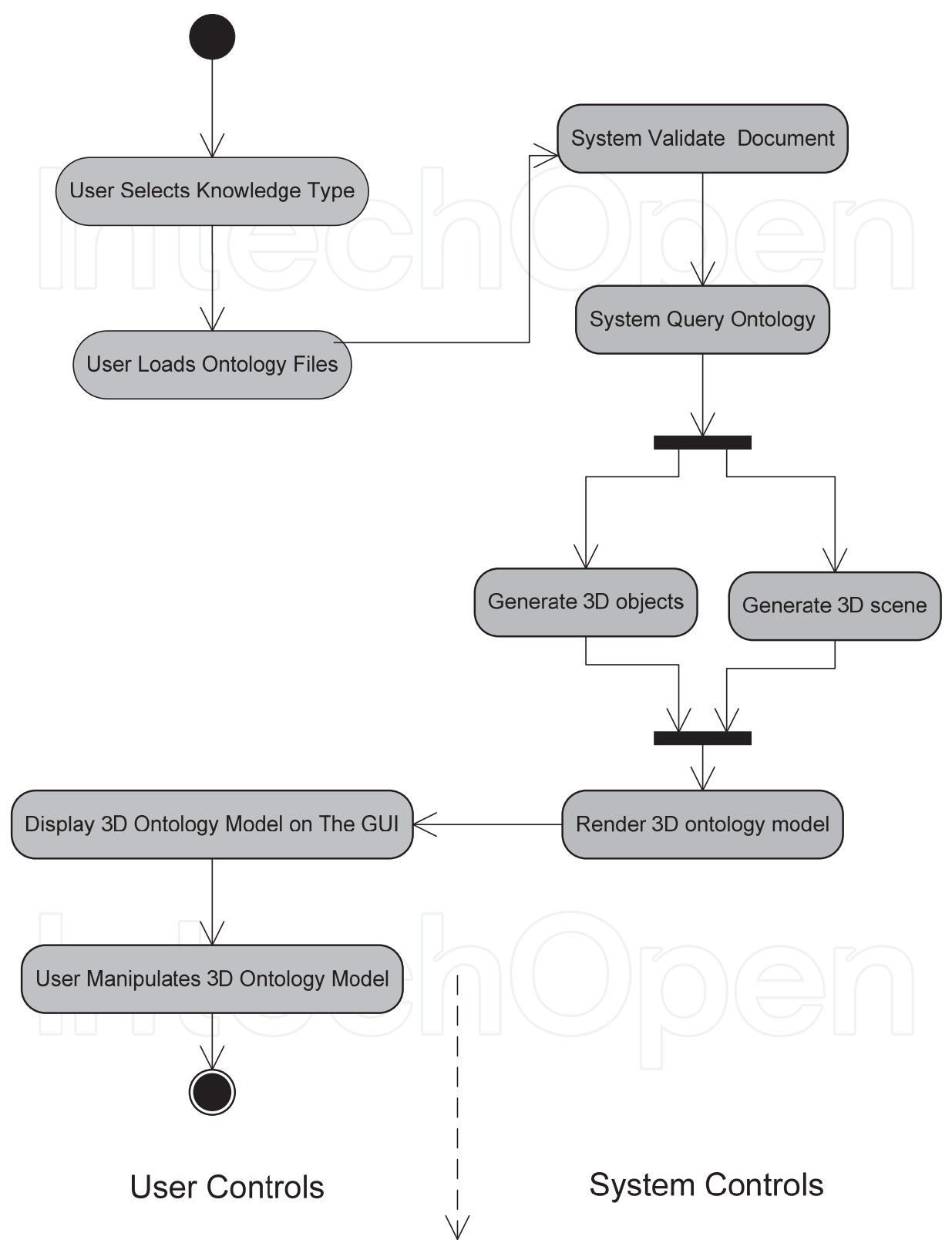
Fig. 2. Activity Diagram of Onto3DViz

Protégé plug-in, Onto3DViz is developed as an independent software application. It requires as input application ontologies stored in the OWL format, which can be developed based on IMT and by other tools or techniques. The ontology documents can be shared and re-used by other systems.

The high-level architecture of Onto3DViz is shown in figure 1. With the ontology editor, the user creates static knowledge models which consist of classes, properties, and relations, and dynamic knowledge models which consist of tasks, objectives, and behaviours of the tasks and links to the class objects. Then the files consisting of the XML and OWL represented ontologies are loaded into Onto3DViz, which generates the 3D visualized ontology model on the computer screen as an output. The user can then adjust the 3D model using the computer input devices of mouse and keyboard for exploring and navigating the visualized model.

The procedure of using Onto3DViz for ontology visualization involves nine steps. This process is also shown in the activity diagram in Figure 2.

1. The user selects the type of knowledge to be visualized, it can be either static knowledge only, or both static and dynamic knowledge.
2. The user selects an ontology document for loading into Onto3DViz.
3. Onto3DViz validates the OWL or XML file. If there is error, Onto3DViz will generate an error message to the user.
4. Onto3DViz extracts the ontology model using the OWL or XML parser and pass the retrieved information to the 3D graphic engine.
5. Onto3DViz generates the 3D ontology models.
6. The 3D graphic engine generates visual objects and visual effects.
7. The GUI displays the ontology 3D visualization model on the screen.
8. The user can adjust the viewing of the 3D visualization model by using the computer keyboard or mouse.
9. Onto3DViz provides adjusted views according to the user's preferences.

The modeling process assumes the user to have no programming knowledge. The tool is designed so that when the user loads a compatible and valid ontology document to Onto3DViz, a visualized ontology model will be constructed and displayed in 3D on the computer screen.

3.1.1 Knowledge extraction and interpolation

After an application ontology has been constructed using a tool like Dyna (Harrison & Chan, 2009) both the static and dynamic knowledge elements are stored in the XML files, which can be shared and re-used by other systems. Instead of being connected to an ontology editor that generates an application ontology, Onto3DViz is a separate stand-alone system that uses OWL and XML documents as inputs for 3D ontology visualization. The benefit of this approach is that Onto3DViz can produce a 3D ontology visualization as long as it receives a valid ontology document from another system or via the network. Since the formats of OWL and XML are well standardized and recognizable, the only requirement for using Onto3DViz is that a valid ontology document is available to the system in either format. Onto3DViz processes a valid OWL or XML formatted ontology file as follows:

1. Import Ontology
    a. If the ontology includes only static knowledge, the user provides a correct path of the ontology file to Onto3DViz so that it can import it.

    b. If the ontology includes both static and dynamic knowledge, the user provides the paths to both the static and dynamic knowledge files to Onto3DViz so that it can import the ontology from both paths.

2. Extraction and Interpolation of Knowledge
    a. Onto3DViz extracts and interpolates the OWL file(s) by applying a Protégé-OWL API. The Protégé-OWL API originally is one of the plug-in packages from the Protégé-OWL editor. This API is efficient in extracting and interpolating knowledge from an OWL file.
    b. Onto3DViz parses an XML-formatted file that stores dynamic knowledge by using Document Object Model (DOM[4]).

3. After the knowledge has been extracted from the ontology, it is stored into the Java objects that are custom designed for storing visual data in the computer memory, and which are used by Onto3DViz for creating the 3D model.

3.1.2 Visual object creation and display

The objective of Onto3DViz is to visually represent knowledge elements specified in the IMT. The tool employs three techniques for accomplishing this objective; they are (1) representation of concepts by the shape of visual objects, (2) size scaling of visual objects, and (3) positioning of visual objects. These techniques enable Onto3DViz to represent an ontology with minimal literal description while enhancing visual effects of the representation. The three techniques are discussed in detail as follows.

3.1.3 Shape of visual objects

Onto3DViz uses the following symbols to represent the following concepts:

- Class : Sphere
- Objective : Cylinder
- Instance : Box
- Task : Cone
- Relationship between concepts : Line

3.1.4 Size scaling of visual objects

For the purpose of showing hierarchical structure among concepts of an ontology, Onto3DViz decreases the size of the visual objects that occupy the lower levels in the hierarchy. For example, if the class of Thing is the parent class of all classes, then it is scaled to be the largest sized object. A subclass of the Thing class is a class, and the visual object of this class is scaled to be 70 percent of the size of the Thing class. Similarly, subsequent children classes are scaled to be 70 percent of the associated parent class. Hence the classes in the default hierarchical view of an application ontology represents the visual objects that are decreasing in size from top to bottom. By employing this technique, the hierarchical relationships as reflected in successive parent-and-child relationships are clearly represented among concepts.

---

[4] Document Object Model, http://www.w3.org/DOM/

3.1.5 Positioning of visual objects

Every 3D system requires a set of 3D coordinates to define its position and orientation. The 3D coordinate system used in Onto3DViz is defined as shown in figure 3.

This definition of the 3D coordinates in Onto3DViz is commonly adopted in most 3D computer graphic languages. The X axis is a horizontal line on a flat piece of paper, going from left to right, and the numbering goes from negative on the left to positive on the right. The Y axis is a vertical line on a piece of paper, and the numbering goes from negative in the bottom to positive at the top. The Z axis is perpendicular or orthogonal to the piece of paper, and the numbering follows the direction that is pointing out from the surface of the piece of paper. The intersection of these three axes is the origin. Onto3DViz allocates visual objects into three X-Y planes along the Z axis. They are:

- Class plane: this is the first X-Y plane located at the positive region of the Z axis as shown in figure 4a. The visual objects of classes are placed on this plane.
- Instance plane: this is the second X-Y plane located at the origin of the Z axis as shown in figure 4b. The visual objects of instances are located on this plane. If there is no instance being created in the ontology, this plane is not visible in the 3D model.
- Objective and task plane: This X-Y plane is perpendicular to and located in the negative region of the Z axis as shown in figure 4c. This plane contains the dynamic knowledge elements. The objectives are placed at the top region of this vertical plane, and the tasks associated with the objectives are placed in the bottom region of this plane. If there is not any dynamic knowledge element in the ontology, this plane is not visible in the 3D model.

The technique of organizing the placement of knowledge concepts at multiple levels in the three dimensional space enables categorization according to the types of the knowledge concepts, but also solves the problem of concepts being overcrowded in a shared space.
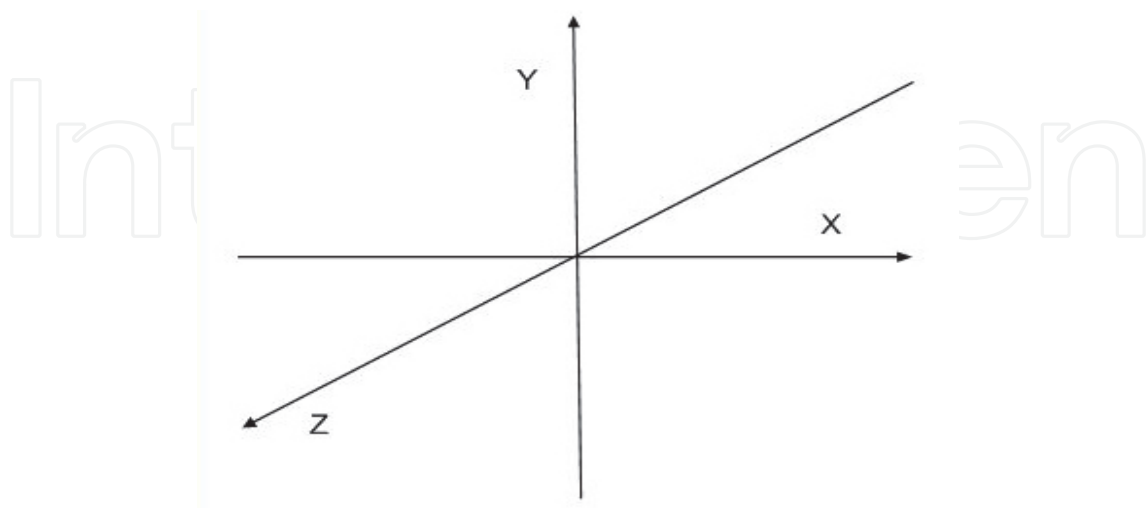

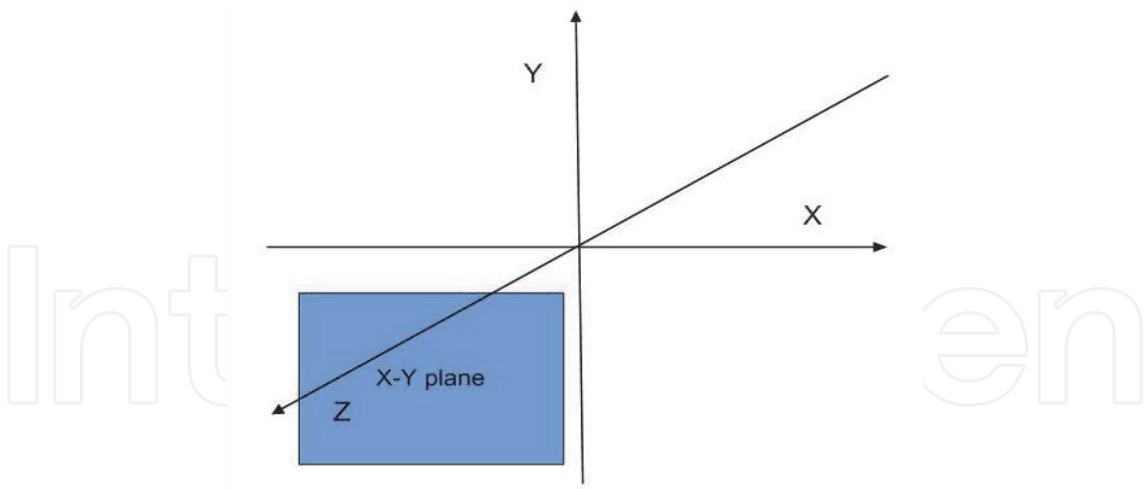Fig. 3. The 3D coordinate system of Onto3DViz
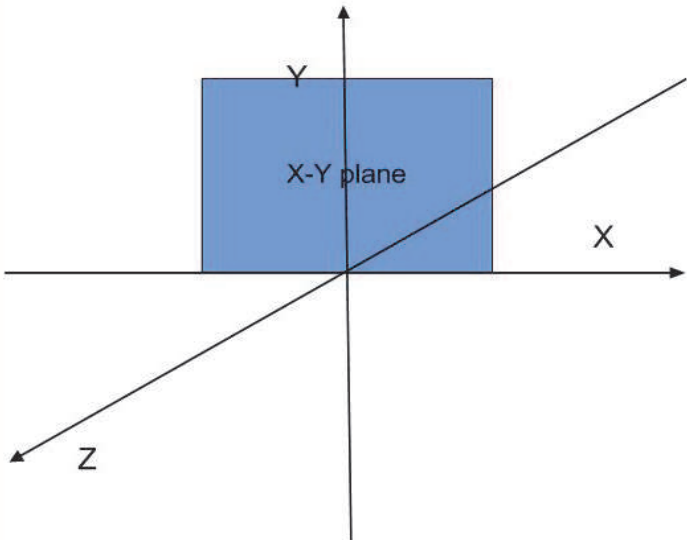
Fig. 4a. the X-Y plane stores classes

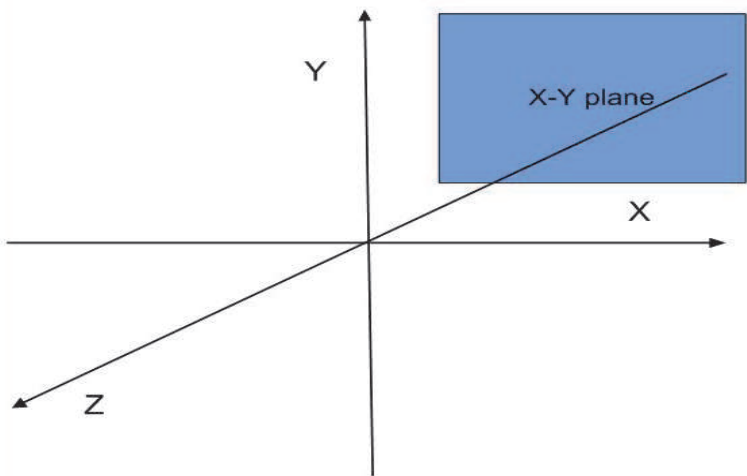Fig. 4b. the X-Y plane stores instances

Fig. 4c. the X-Y plane stores objectives and tasks

3.1.6 User interaction

Interactions between the 3D model generated by Onto3DViz and the user are realized using the physical devices of the computer mouse and the keyboard. Both the mouse and keyboard can be used for controlling operation of the 3D model in Onto3DViz as follows.

1. Mouse:
   - Rotation of the model : the user can drag the mouse while pressing the left button
   - Translation of the model : the user can drag the mouse while pressing the right button
   - Zoom into the model : the user can drag the mouse while pressing the middle button (or holding the Alt key on the keyboard)
2. Keyboard:
   - Rotation
     - Rotate to the left : use ← Key
     - Rotate to the right : use → key
     - Rotate upward : use PageUp key
     - Rotate downward : use PageDown key
   - Translation
     - Translate along Z axis toward the front : use ↑ key
     - Translate along Z axis toward the back : use ↓ key
     - Translate along X axis toward the left : use ← key
     - Translate along X axis toward the right: use → key
     - Translate along Y axis upwards : use Alt-PageUp keys
     - Translate along Y axis downwards : use Alt-PageDown keys
   - Zoom
     - Zoom out : use – key
     - Zoom in : use + key
     - Reset to default viewing position : use = key

By combining these user control actions, the users can manipulate the model and obtain multiple perspectives of the 3D model of an application ontology.

3.2 Design of the ontology management system

To address the objective of knowledge management, a tool called Distributed Framework for Knowledge Evolution (DFKE) (Obst, 2006) has been developed. The high level architecture of DFKE is shown in figure 5.

The DFKE interacts with an ontology editor via the ClientServices. ClientServices provides functions for: (1) representing knowledge in a generic knowledge representation, and (2) importing and exporting knowledge models from and to XML. DFKE uses a peer-to-peer (P2P) network architecture. The "Feeder Node" is a node in this P2P network. The DFKE stores the knowledge projects in a knowledge repository or database, which is attached to a "Feeder Node". Network communication between the Client Services and the FeederNode (and Knowledge Repository) is achieved using the Common Object Request Broker Architecture (CORBA)[5]. DFKE also handles security, by encrypting network communication and digitally signing objects. There is also a Project Browser for assisting the user in finding projects on the P2P ontology network.

---

[5] CORBA, http://www.omg.org/gettingstarted/corbafaq.htm

One of the goals of the ontology management system was to hide as much of the internal processing of ontology management from the user as possible; the user should only be concerned with ontology modeling, not ontology management. With this goal in mind, much of the ontology management work is done automatically in the background for the user.Documentation is an important part of ontology management, which is often neglected. In creating ontologies for sharing and reuse, it is particularly important to provide information describing the author of the ontology. Author information includes the user's name, email address, organization, etc. Since author information is the same for all knowledge elements created by the user, the author information can be gathered once and then used by the system to automatically fill the input fields. The first time the user launches the Ontology Editor, a window requesting user information is displayed. This window has input fields for gathering personal identification information such as the user's name, email, web page, and organization. When a user creates any knowledge element, this gathered information is automatically attached to the knowledge element. All of this user information can be viewed in the property window of any knowledge element.
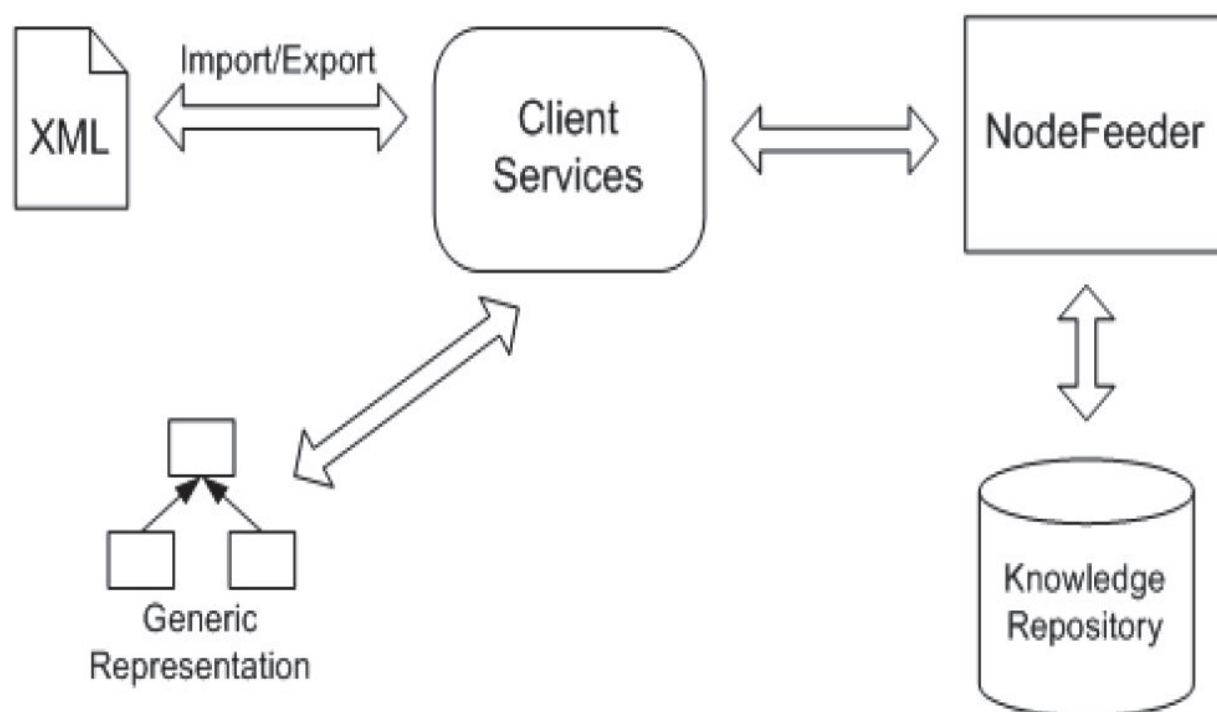

Fig. 5. High-level architecture of DFKE

The window also has an input field for a digital signature. A digital signature is "a data string which associates a message (in digital form) with some originating entity" (Maedche & Staab, 2003). The DFKE uses digital signatures to identify the author of an ontology element and to restrict access to an ontology element to only those people who should have access, thereby protecting the ontology from malicious ontology authors. To generate a digital signature for the user, the user enters a passphrase into an input field in the user information window, and clicks a button to generate the digital signature. In the background, the DFKE uses the user information and passphrase to generate the digital signature. This digital signature is used to identify each user and the knowledge elements the user creates. If another user attempts to modify a knowledge element that was created by the user, the other user will be denied.

The distributed and heterogeneous nature of the Semantic Web makes it very difficult for users to find ontologies. The Unified Ontology View (UOV) was proposed in (Harrison et al., 2005), and UOV provides a layer that hides the heterogeneous and distributed nature of the ontology network. The user is able to access the ontology network from anywhere and obtain a unified view of the ontologies in the network. Figure 9 gives an overview of the network environment which includes the servers, the ontology projects, and the client application. In this network, whether the Client is connected to server A, B, or C is irrelevant; the Client will be presented with the same view of the ontology projects in the ontology network.
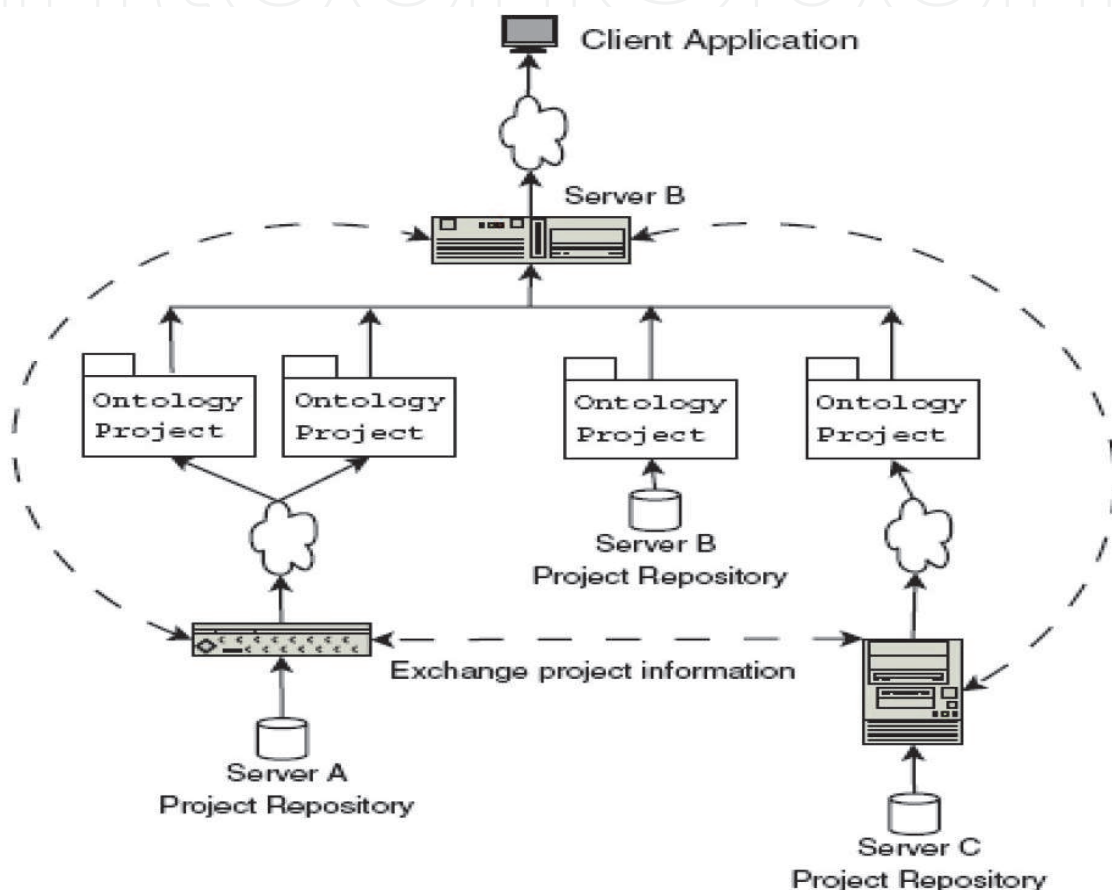

Fig. 6. Server view of ontology projects

Figure 7 shows the ontology projects from the perspective of the Client. When the Client connects to the ontology network, it only sees the ontology projects. It is possible that the ontology projects are distributed among many different types of systems, however, the Client does not need to be aware of this fact.

The ontology management system implemented the idea of a UOV by using a peer-to-peer (P2P) architecture, as shown in Figure 8. Knowledge projects created with the Ontology Editor are stored in a database or knowledge repository, which is connected to a Feeder node. The Feeder node that the Ontology Editor connects to may be local or remote. Feeder nodes can share knowledge projects with each other. The Feeder nodes discover other feeder nodes by contacting a central Hub node. Ontology projects are automatically added to the network when they are created with the Project Wizard. Once a project is added to a node on the network, it can be accessed from any node in the network.
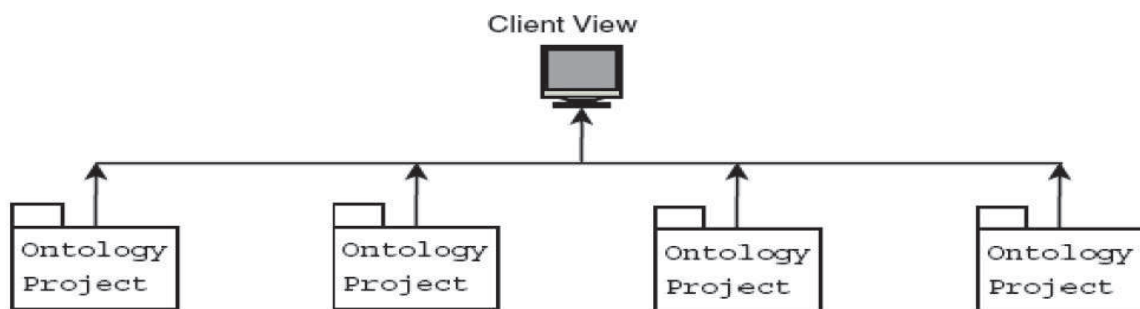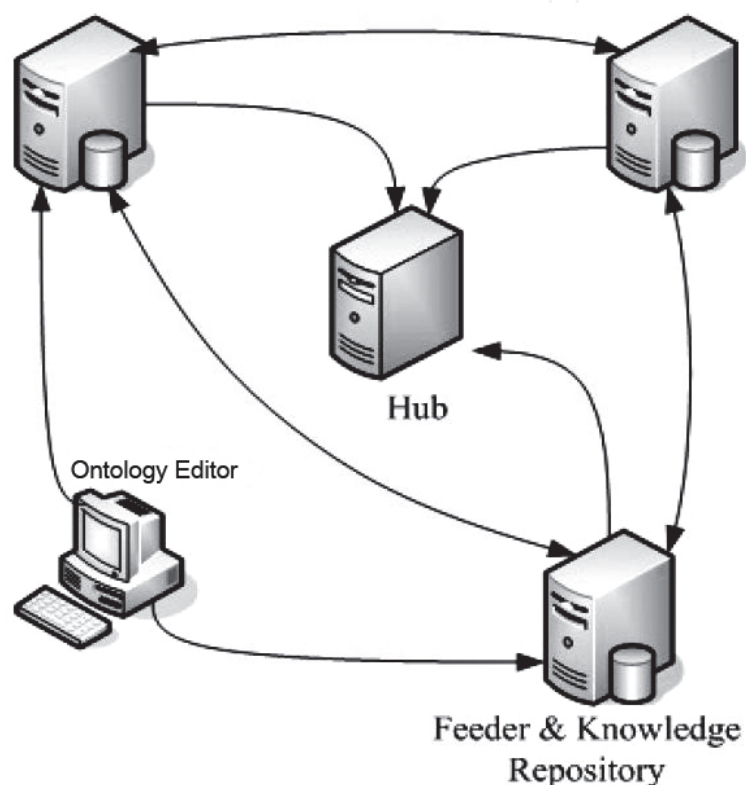
Fig. 7. Unified ontology view

Fig. 8. P2P architecture

Projects can be used outside of the network by saving them in an XML file.

To support knowledge evolution, DFKE uses a generic ontology representation that was developed for representing conceptualizations and refinements of an ontology (Obst & Chan, 2005). Conceptualizations involve to be major changes to an ontology, while refinements reflect minor changes. The major/minor versioning scheme proposed by (Klein & Fensel, 2001) is used to keep track of the versions of the ontology as conceptualizations and refinements are added. Removing a conceptualization can have serious problems on other systems that depend on the conceptualization. Therefore, conceptualizations cannot be removed. The DFKE uses the concept of monotonicity to guarantee that existing concepts are not removed. The Class Editor helps support monotonicity by denying users the ability to remove knowledge elements that have been marked as monotonic by DFKE.

4. Discussions

Applications of the two tools have been conducted and presented in (Guo & Chan, 2010) and (Obst, 2009). Some strengths and weaknesses of the tools have been observed from the application experiences.

Onto3DViz is able to successfully render the complex domain concepts and relationships among concepts in a 3D model, thereby creating a visualized knowledge model that IMT formulated. The user can easily examine and manipulate the 3D model by performing the operations of zooming, rotating, and translating using the computer mouse or keyboard. The 3D model clearly shows the hierarchical structure of the static knowledge; it can also support representing the tasks associated with each object and arranging the tasks in the correct order. However, some weaknesses were also observed in the current version of Onto3DViz. First, when the visual objects are too close to each other, the objects and labels become overlapped, which makes it difficult for users to clearly see details of the model. Secondly, Onto3DViz does not currently support a search function. If the user wants to retrieve a particular concept, he or she has to manually examine the 3D model so as to find the corresponding visual object. This can be difficult if the ontology is complex or consists of a large number of concepts. Thirdly, if an application ontology consists of too many hierarchical levels, the lower level nodes will be small and difficult to see. This happens because the scaling technique implemented in Onto3DViz (described in section 3.1.4) reduces the size of the visual nodes in the lower levels.

The biggest limitation of the DFKE is the lack of integration with an ontology editor, such as Protégé and Dyna. This limitation means the user would need to conduct ontology modeling and ontology management as two different tasks and on two separate software tools. This is not convenient and DFKE in its current version does not adequately support the ontology authoring process.

5. Conclusions and future works

The objective of this research is to develop a suite of ontological engineering tools which supports (1) static and dynamic knowledge visualization, and (2) management of an application ontology. The motivation for this objective is derived from an analysis of existing tools which reveals their lack of support for modeling and visualizing dynamic knowledge. Among the few tools that support visualization, their capabilities for visualizing a large volume of information is limited due to the constraints of the 2D graphical space. Another weakness observed from our analysis is that existing ontology management frameworks provide inadequate support for replication and evolution of ontology, and they do not support detecting when a public domain ontology has been possibly tampered with. To address these limitations, Onto3DViz has been developed to support 3D visualization of an ontology model and the DFKE was designed for ontology management.

However, Onto3DViz and the monotonic DFKE have some weaknesses which will be tackled in the future. The current version of Onto3DViz can be enhanced by adding a collision avoidance system to address the problem of overlapping concepts. The collision avoidance system would verify that the visual objects in a 3D model do not collide. And if a collision is detected, it will automatically reassign the location of the visual objects and adjust the space between them. To address the difficulty of finding concepts in a model, a feature that supports automatic identification of a required concept can be added to Onto3DViz. To prevent overly tiny visual nodes from being generated, the next version of

Onto3DViz will set a cut off level for applying the scaling technique. That is, the system will include a pre-determined cut-off or optimal level for applying the scaling technique, so that the descendant nodes will remain the same size after the cut off level is reached. Moreover, assigning colors to the visual objects can also assist the user in indentifying different types of concepts. For instance, the visual objects for classes can be blue and visual objects for instances can be yellow. The different colors can help to distinguish the different knowledge types in a visualized ontology. Onto3DViz can also be implemented as a Protégé-OWL editor plug-in, so that ontology visualization and editing can be done in real time. As well, more user controls can be implemented which will enhance user-friendliness of the tool.

The DFKE has been designed and implemented as a standalone application and is not integrated with any other ontology tool as mentioned in section 4. This limits usefulness of the tool because ontology development and management have to be conducted separately. In the future, it will be useful to integrate DFKE with other ontological engineering tools, such as Protégé (Protégé), Dyna (Harrison & Chan, 2009), and Onto3DViz, so that users can conduct ontology authoring, editing, visualization and management in a unified system.

6. Acknowledgment

7. References

Berners-Lee, T., Hendler, J., & Lassila, O. (2001). The Semantic Web. *Scientific American* , 34-43.

Bosca, A., Bonino, D., & Pellegrino, P. (2005). OntoSphere: more than a 3D ontology visualization tool. *Proceedings of SWAP 2005, the 2nd Italian Semantic Web Workshop.* Trento, Italy: CEUR Workshop Proceedings.

Chan, C. (2004). From Knowledge Modeling to Ontology Construction. *Int. Journal of Software Engineering and Knowledge Engineering, 14(6)* .

Cranefield, S., Pan, J., & Purvis, M. (2005). "A UML Ontology and Derived Content Language for a Travel Booking Scenario", *Ontologies for Agents: Theory and Experiences* Basel: Birkh\auser (2005), p. 259-276.

Djuric, D., Gasevic, D. & Devedzic, V. (2005). "Ontology Modeling and MDA", In *Journal of Object Technology*, 4(1), 2005, pp. 109-128.

Fensel, D., Van Harmelen, F., Ding, Y., Klein, M., Akkermans, H., Broekstra, J,. . . Horrocks, I. (2002). "On-To-Knowledge in a Nutshell", 2002.

Gomez-Perez, A. Fernandez-Lopez, M., & Corcho, O. (2004). "Ontological Engineering: with examples from the areas of Knowledge Management, e-Commerce, and the Semantic Web", Springer, 2004.

Guo, S. & Chan, C. (2010). A tool for ontology visualization in 3D graphics: Onto3DViz. *The Proceedings of the 23rd Canadian Conference on Electrical and Computer Engineering*. Calgary, AB, May 2010.

Gruber, T. (1993). Towards Principles for the Design of Ontologies Used for Knowledge. *In Guarino & Poli (eds): Formal Ontology in Conceptual Analysis & Knowledge Representation* .

Harrison, R. & Chan, C. (2005). "Implementation of an Application Ontology: A Comparison of Two Tools", *Artificial Intelligence Applications and Innovations II: Second IFIP TC12 and WG12 Conference on Artificial Intelligence Applications and Innovations (AIAI-2005)*, Beijing, China, Sept. 7-9, 2005, pp. 131-143.

Harrison, R. Obst, D., & Chan, C. (2005). "Design of an Ontology Management Framework", In Proceedings of the 4th International Conference on Cognitive Informatics 2005 (ICCI'05), University of California, Irvine, USA, Aug. 8-10, 2005, pp 260-268.

Harrison, R. & Chan, C. (2009). "Dynamic Knowledge Modeler" *Artificial* Intelligence *for Engineering Design, Analysis and Manufacturing,* Volume 23 Issue 1, February 2009 Cambridge University Press New York, NY, USA.

Janzen, D. & Saiedian, H. (2005). "Test Driven Development: Concepts, Taxonomy, and Future Direction", IEEE Computer, September 2005.

KANO. (n.d.). Retrieved on Thursday 6 January 2011. < http://kaon.semanticweb.org/>

Obst, D. & Chan, C.W. (2005). "Towards a Framework for Ontology Evolution", *18th Annual Canadian Conference on Electrical and Computer Engineering (CCECE05)*, Saskatoon, Saskatchewan, Canada, May 1-4, 2005.

Obst, D. (2006). "Distributed Framework for Knowledge Evolution", Presentation at University of Regina Grad. Student Conf., Regina, SK, Canada, 2006.

Obst, D. (2009)., "Monotonic Versioning of Ontologies in a Peer-to-Peer Environment", Master Thesis, University of Regina, July 2009.

OntoBuilder. (n.d.). Retrieved on Thursday 6 January 2011.
          <http://iew3.technion.ac.il/OntoBuilder/>

Ontolingua. (n.d.) Retrieved on Thursday 6 January 2011.
          <http://www-ksl-svc.stanford.edu:5915/>

OWL Unit Test Framework. (n.d.). Retrieved on Thursday 6 January 2011.
          < http://www.co-ode.org/downloads/owlunittest/>

Protégé. (n.d.). Retrieved on Thursday 6 January 2011. http://protege.stanford.edu

Protégé OWLViz. (n.d.). Retrieved on Thursday 6 January 2011. <http://www.co-ode.org/downloads/owlviz/co-ode-index.php>

Klein, M. & Fensel, D. (2001). "Ontology Versioning on the Semantic Web", *Proceedings of SWWS '01, The First Semantic Web Working Symposium*, pp. 75-92, August 2001.

Maedche, A. & Staab, S. (2003). "Ontology Learning", In: Staab, S., Studer, R., eds Handbook of Ontologies in Information Systems, Springer Verlag, pp 173-190, 2003.

McGuiness, D., Fikes, R., Rice, J., & Wilder, S. (2000). "An Environment for Merging and Testing Large Ontologies". In *Proceedings of KR 2000*, pp. 485-493.

WebODE. (n.d.). Retrieve from Thursday 6 January 2011.
          <http://webode.dia.fi.upm.es/WebODEWeb/index.html>

Wu, J. & Storey, M.-A. (2000). A Multi-Perspective Software Visualization Environment. *conference of the Centre for Advanced Studies on Collaborative Research 2000*, (pp. 31-40). Mississauga, Ontario, Canada.
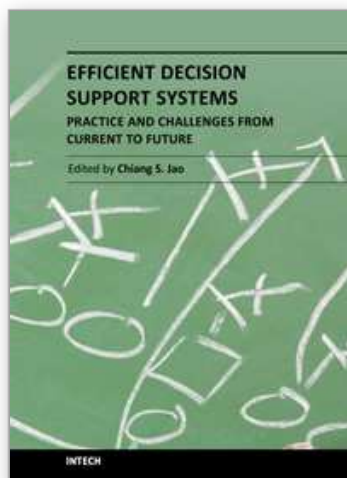
Efficient Decision Support Systems - Practice and Challenges From Current to Future

Edited by Prof. Chiang Jao

This series is directed to diverse managerial professionals who are leading the transformation of individual domains by using expert information and domain knowledge to drive decision support systems (DSSs). The series offers a broad range of subjects addressed in specific areas such as health care, business management, banking, agriculture, environmental improvement, natural resource and spatial management, aviation administration, and hybrid applications of information technology aimed to interdisciplinary issues. This book series is composed of three volumes: Volume 1 consists of general concepts and methodology of DSSs; Volume 2 consists of applications of DSSs in the biomedical domain; Volume 3 consists of hybrid applications of DSSs in multidisciplinary domains. The book is shaped upon decision support strategies in the new infrastructure that assists the readers in full use of the creative technology to manipulate input data and to transform information into useful decisions for decision makers.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Simon Suigen Guo, Christine W. Chan and Robert Harrison (2011). Decision Support Tools for Ontological Engineering, Efficient Decision Support Systems - Practice and Challenges From Current to Future, Prof. Chiang Jao (Ed.), ISBN: 978-953-307-326-2, InTech, Available from: http://www.intechopen.com/books/efficient-decision-support-systems-practice-and-challenges-from-current-to-future/decision-support-tools-for-ontological-engineering

INTECH
open science | open minds