3D Visualization – Ontology Visualization for Mapping

David Brokenshire & Pat Lougheed

School of Interactive Arts and Technology

Simon Fraser University

Abstract

In this paper we describe the VOM system we have created for Visualization of Ontology Mapping. It attempts to enable manual ontology mapping by using information visualization techniques to display large ontologies in a clear and understandable way. The main technique we employ is a three-dimensional tree based visualization that allows for the selection of multiple class, exclusion of classes, and saving the merged classes to an OWL ontology. Our visualization attempts to follow the principles of visualization.

Introduction

In this paper we present a principled method of visualizing ontologies for mapping. Ontology mapping is a major area of research and is a resource intensive task. Improved visualizations can support knowledge engineers in the mapping process, making it faster and less error prone. We begin with a review of the key concepts, ontologies, ontology mapping, and ontology visualization and cover existing techniques for ontology and hierarchy mapping in some detail. We the present the rational and design principles behind our visualization technique and the algorithms in detail. Then we present the results of our work in detail including screenshots and an explanation of the interface. Finally we draw our conclusions and propose future work.

Ontologies

Ontologies are a form of knowledge representation which originated with frame based systems in Artificial Intelligence. They are frequently defined as a "formal specification of a conceptualization" (Gruber, 1993). Roughly speaking, they are a formal method for recording the vocabulary of a domain of discourse along with the class structure of the vocabulary (this is a taxonomy) plus arbitrary relationships between the classes. Ontologies also use a category of logic called Description Logics (DL) to describe and reason about the ontology and infer implicit knowledge. Ontologies can be 'instantiated' with specific instances and values of the classes they define.

Ontologies range in size from very small (a few to a few dozen classes and properties) up to the truly massive (10,000 to 1,000,000+ classes and relationships). Ontologies typically have many more instances than they have classes.

Ontologies can be represented in many forms and languages. Currently the major language for representing ontologies is the Web Ontology Language (OWL), a W3C recommendation (McGuinness & Harmelen (Eds.), 2004). OWL grew out of the DAML+OIL initiative, a previous ontology representation language. OWL is specified in RDF, the Resource Description Framework, which is also an XML language. Though there are many other languages, we restrict ourselves to OWL for this project.

Ontology Mapping

Ontologies are growing in popularity thanks in large part to the rise of the Semantic Web. The Semantic Web proposes that online knowledge should be semantically marked up for agents to use intelligently, in contrast to the current state of affairs where web information is only marked up graphically for humans.

Thanks to the semantic web ontologies are now often shared between different groups using the ontology for various purposes. Since these ontologies may be created independently without any standardization, multiple ontologies may be created to represent the same information. Bridging the gap between two or more ontologies that represent over-

lapping bodies of knowledge is called ontology mapping, alignment, and translation.

Researchers are working on semi-automatic or automatic techniques for mapping and translating ontologies (Kalfoglou & Schorlemmer, 2005). Ontology mapping is a hard, as yet unsolved, problem though progress is being made. Mapping software is generally semi-automatic, meaning it can only partially map an ontology before it requires verification and aid from an expert knowledge engineer. As a result the knowledge engineer needs an environment in which to view and contribute to the editing process (Ehrig & Staab, 2004).

Ontology Visualization

Ontology visualization is the practice of representing an ontology visually so it can be more easily understood and manipulated. This is very important given the extreme size and complexity ontologies can reach. Without an adequate visualization they can become very difficult to work with.

Goals. Goals of ontology visualization vary by the task. Some common goals are:

- Navigation
- Holistic understanding
- Editing
- Mapping
- Verification

(Ernst, Storey, & Allen, To Appear)

Visualization Techniques. In this section we seek to present the major variations on ontology visualization technique along with software packages that implement them. Some of these techniques are from the more general field of hierarchy visualization but can be applied directly to ontology visualization.

Hyperbolic visualization uses the properties of hyperbolic geometry to lay out ontologies. The simplest technique is to project the model on the inside of a sphere (Munzner &

Burchard, 1995). As it is using an additional dimension, the 3D representation allows more elements to be fit into the same limited screen space.

OntoRama is a visualization for RDF graphs which uses a hyperbolic visualization. OntoRama also introduces the idea of cloned nodes, where nodes are visualized more than once to reduce the number of crossed lines for relationships and aid readability (Eklund, Roberts, & Green, 2002). This follows the principle of keeping the number of lines smaller than the number of classes when visualizing a graph or tree.

OntoSphere is a ontology visualization which augments the 3D hyperbolic view, combining it with a sophisticated tree view to display information clearly in 3D (Bosca, Bonino, & Pellegrino, 2005). It also uses colour, shape, and size do denote further information.

TreeMaps are a visualization technique which attempts to take a tree and convert it into a series of nested rectangles. Jambalaya is a treemap based visualization plugin for Protege that allows for viewing of an ontology at arbitrary depth levels, and navigation between them. It is based on the SHriMP model developed for navigating codebases, and shows classes, instances, and properties as boxes inside one another (Storey et al., 2001).

Crop Circles is a relatively new technique for visualizing complex hierarchies by encircling children with progressively larger circles on a plane. They are in effect a circular form of tree map. It can also display connections between ontologies via edges between the circular nodes. (Parsia, Wang, & Golbeck, 2005)

Graph based visualization is the most common type of ontology visualization as ontologies are naturally either trees or Directed Acyclic Graphs(DAG). OntoViz uses the GraphViz library to create a detailed static graph of the ontology. This is one of the earliest approaches to visualizing ontologies and is still reasonably effective for editing and navigation (Sintek, 2005).

Spring models, also known as Force Directed Placement (FDP) or gravity based models are a subtype of graph model which dynamically arranges and rearranges a model

based on spring-like connections and forces between elements. For example elements may be pulled closer to their parent nodes and pushed away by their siblings. TGVizTab uses the TouchGraph library to create a spring model graph of the ontology (Alani, 2003).

Self Organizing Maps (SOM) are a type of Neural Network (NN) which allocates learns a mapping of high-dimensionality data into a lower dimensionality for viewing. Tu et al present a hybrid SOM and FDP technique for getting a high level overview of an ontology (Tu et al., 2005).

3D Cone Trees are an approach originally suggested by Robertson (Robertson, Mackinlay, & Card, 1991). In 3D cone trees a tree is constructed based on the is-a relationships in the ontology (superclass/subclass relationships). Subclasses are then arranged in circles on levels. Levels in the tree corresponds to visual depth. In its initially proposed for it was limited to roughly 1000 nodes. However additional techniques such as those below, can be used to increase this number significantly (Carriere, 1995).

Filtering/Focusing/Neighborhood view Limiting the number of classes and relationships in the ontology visualization at a given time is considered essential in almost every ontology visualization scheme which attempts to deal with large ontologies. Filtering typically involves showing only a subset of the ontology which matches some criteria selected by a user. Filtering can involve search, limiting property types, or other relevant parameters.

A neighborhood view is a specific type of filtering which only shows the classes/relationships within a certain 'neighborhood' or distance surrounding the primary area of interest. Focusing, also known as zooming or fish-eye view, involves enlarging the area of interest and shrinking the rest and/or moving it to the periphery so the user can focus their attention on the relevant information more easily. Focusing is good for inspecting things in detail locally, but the user can easily forget if they have to refocus repeatedly when doing a task which is not local.

Mapping Specific The previous discussion has covered a broad spectrum of visualizations which are available. We have not however, covered things specifically from an ontology mapping perspective largely due to a lack of work in this area. Ontology mapping requires that the user, typically a knowledge engineer, be able to see the ontologies to be mapped simultaneously in order to define or refine the mapping. The must be able to see the relevant details and structure of both ontologies and compare them visually and accurate with each other.

Typically a semi-automatic mapping technique executes and the knowledge engineer aids the process and then refines, verifies, and corrects the results. Klein et al provide a system to map RDF ontologies to deal with versioning on the web. Their approach is essentially a visual diff tool for the actual RDF/XML of the ontology (Klein, Fensel, Kiryakov, & Ognyanov, 2002). Mapping is often done with the aid of ontology editors, instead of with more elaborate visualizations.

Ontology Editors. Ontology editors typically provide a hierarchical tree layout of the classes in the ontology, as well as some GUI-form based method of viewing the relationships, properties, and instances. This tends to be insufficient for seeing the information in the ontology in a useful way. It also only provides a local view of the ontology, specific to that class and its immediate relationships. Protege is the most popular ontology editor currently. It offers a plugin interface which allows visualizations and other components to be built into its interface, combining rich editing abilities with visualization (Protege-Project, 2004).

Our Approach

In this section we describe the approach we took to creating our tool, our goals, the principles of visualization we attempted to follow, and the algorithms we implemented.

Goal

The goal of the overall project is to simplify ontology merging through a visualization which allows the user to see similarities and differences between two ontologies quickly, and

allows them to see the result of stating that understanding by beginning to map the two ontologies visually. The goal of the algorithm is to lay out the classes in 3D space so that they are clearly visible and so that similarities and differences between two ontologies are also easy to visualize.

Principles

We are following several principles in the creation of our algorithms. Firstly that similar structures should look similar and different structures should look different. This is important to the knowledge engineer being able to see at a glance what is an is not similar about ontologies to be mapped. The simple approach we took to this was that the structure should be laid out according to the hierarchy in a clearly visible fashion. Implementing this principle fully would require a semi-automatic mapping technique which could identify semantic similarity and difference in more detail.

The second principle we followed was that visualizing the same ontology twice should result in the same visualization. This is important to the knowledge engineer being able to learn what to expect. If the visualization can change without the meaning changing the user might erroneously assume that their is a change in meaning.

We then follow several of the perceptual principles outline (Ware, 2004). Specifically we use filtering to keep the number of links as low as possible, we visually categorize the three ontologies using colors selected according to perceptual guidelines, and we make use of animation of changes can be used to offload processing onto the perceptual system. All changes in the system are animated simply so that the user can follow the results and not lose the structure. Finally we allow the user to move their point of view around and through the structure, which allows them to more easily see and understand the structure of the 3D graph.

Algorithms

As this was our first attempt at producing a ontology mapping visualization we decided to take an approach whereby we would create progressively more complex visual-

izations. As a result we produced four different visualization algorithms, though we only picture the final one in the paper. All of our algorithms implement the principle of repeatability, the same ontology will look the same if loaded multiple times. The algorithms have similar structures looking similar to a limited extent, however we have not yet been successful in ensuring that different structures look different.

2D Grid. The 2D grid is the simplest algorithm. It lays out all of the classes uniformly on points of a two dimensions grid. Since the class nodes are represented by spheres the grid is not exactly 2D, but all of the spheres reside on the same plane.

3D Grid. The 3D grid is an extension of the 2D grid from a single plane to a cube. The ontology classes are uniformly laid out on grid points on and within the cube.

Circle. The circle layout is a simple layout which arranges all the ontology classes around a circle with the classes uniformly spaced in arbitrary order.

Multiple 3D Tree. The 3D tree visualization is our most ambitious visualization by far. This visualization is very similar to the 3D cone tree visualization created by (Robertson et al., 1991) which we described in the introduction, only we are displaying three ontologies instead of one.

Each ontology tree is laid out simply from the top down. The highest point on the y axis represents the 'top' of the ontology, which in our case is always owl:Thing. Each subclass relationship moves down a level in the tree. Each group of subclasses of a class are arranged on a circle on the XZ plane, and those circles are then themselves arranged in a larger circle on the XZ plane to avoid overlapping.

The primary sub/superclass relationships are always depicted via lines running from the superclass to the subclass. Subclass relationships always move down the tree, and additionally to denote the directionality of the relationship the the lines are narrowest at the superclass and broaden towards the subclass.

Since all owl ontologies are rooted at owl:thing we show the thing node only once in our visualization, in effect pre-mapping it. Then the other ontologies inherit from it. One
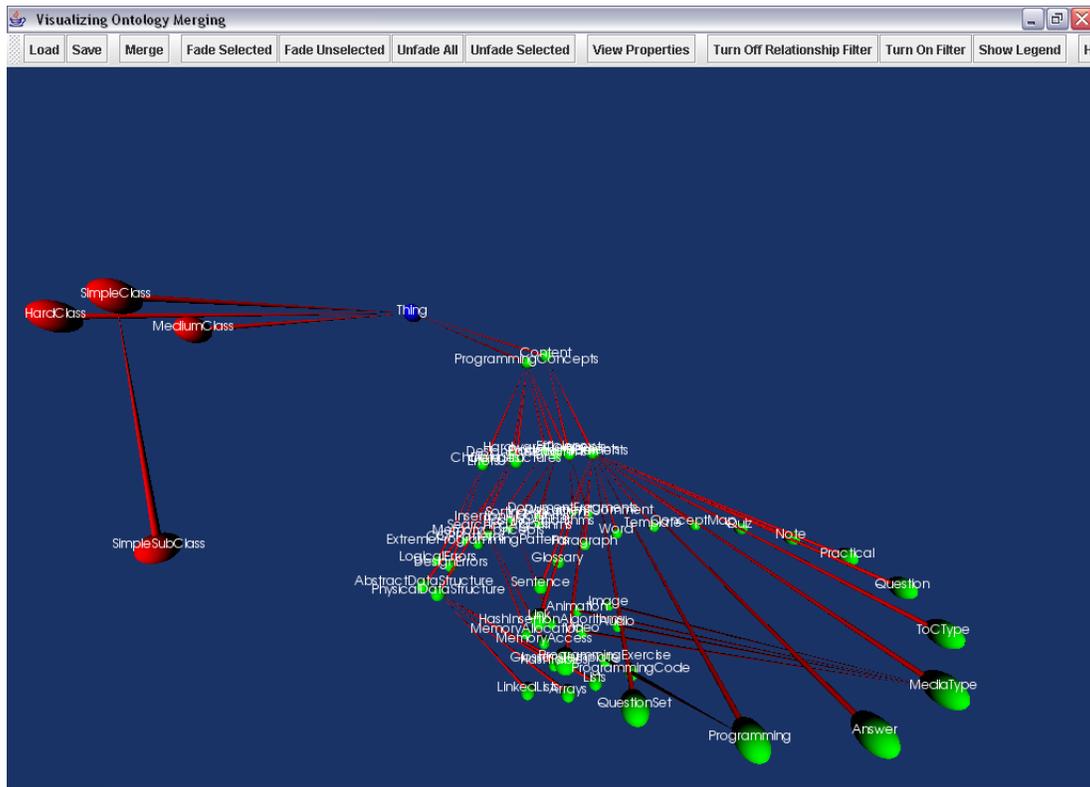
Figure 1. The initial screen with two ontologies.

loaded ontology is displayed on the negative side of the x-axis, the other on the positive side of the x-axis, with the mapped ontology centered in between. As the user maps classes into the mapped ontology they move to their new location in the middle layout in an animated fashion, and change colour to indicate their presence in the new merged ontology.

Results

Our resulting program allows for the merging of ontology classes, viewing of class properties, filtering of relationships, and the fading of classes and relationships not important to the user.

The program loads two default ontologies and displays them. Other ontologies can be used by pressing the *Load* button and entering the ontology locations (currently, only URLs are supported).
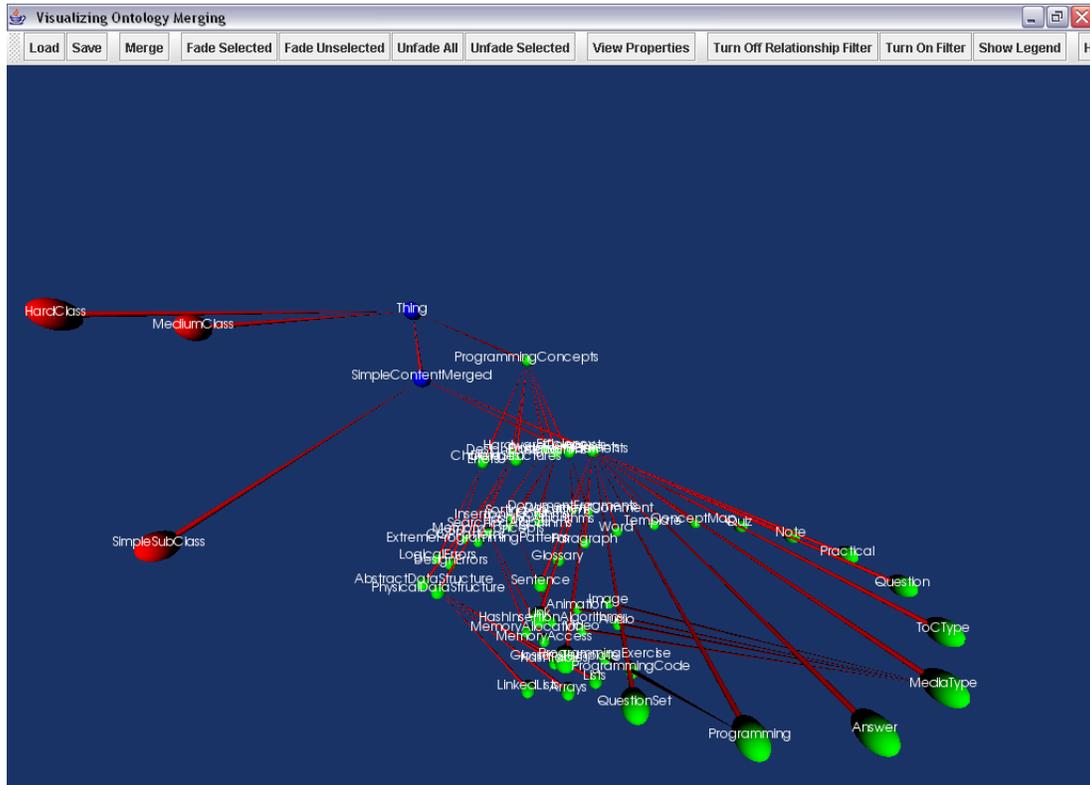
Figure 2. Two ontologies with a merged class.

The ontology can then be merged by selecting one class from each ontology and pressing the *Merge* button. The program will animate the two classes merging, and alter the colour to make the new class stand out. The merged ontology can them be saved by pressing the *Save* button.

Classes can be faded by selecting some objects (or no objects) and pressing the *Fade Selected* or *Fade Unselected* buttons. Relationships will fade if both of it's endpoints are faded as well. *Unfade All* and *Unfade Selected* will unfade all or some classes, and the appropriate relationships.

All relationships other than subclass/superclass relationships are initially filtered out. These relationships can be progressively turned on for viewing by pressing the *Turn Off Relationship Filtering* button and selecting a relationship to view. A maximum of 4 relationships can be viewed at a time. Specific relationships can be turned off by pressing the

Figure 3. Two ontologies with an unfiltered relationship and faded classes.

Turn On Filtering button.

Relationship colours are chosen from four at runtime, depending on the order relationships are unfiltered; relationships are also unlabeled. As such, a legend is required, and this can be viewed by pressing the *Show Legend* button.

Finally, a help button gives a shift help screen on world navigation. The world is fully navigable, and supports zoom, pan and rotate operations.

The graphics themselves are created with the help of the Visualization Toolkit (VTK) Library. Due to the fact that the Java bindings are no created by default, a custom compile of the libraries were required. Additionally, the interactor style that allows for the selection of multiple objects is a custom extension to VTK, and due to various ways the Java bindings are created, was programmed in the original library's C++. This required two separate compiles - one for VTK, and one for the project built in Java itself. Stock compiles of VTK

will not work with the system.

Conclusion

Limited work has been done in ontology visualization for mapping. However, related work has been done in hierarchy visualization which can be fruitfully applied to ontology visualization. Ontology mapping is a ripe area for visualization because it is a complex task with a large number of variables to track which has not been automated successfully to date. The 3D tree visualization is promising, but requires filtering to be able to deal with large ontologies. The visualizations we chose have the virtue of consistency between uses, which allows the user to visually learn the structure. Our work was limited by our inability to implement measures of semantic distance for located the components within the tree in time.

Future Work

There are a number of ways this project could be expanded. First, more robust and full feature ontology processing, such as the ability to merge properties, the ability to map properties within merged classes, and the ability to view instances. The display of text could be greatly improved; while the current solution is the best that could be found within VTK's framework, there are other possibilities that may prove fruitful with a little more development. Auto-rotation of trees. As well, by incorporating semi-automatic ontology mapping techniques, we may be able to make some of the work done via the tool automated.

The program should be made more robust by separating the custom VTK extensions from VTK itself, as it is built now. This would allow for using stock VTK installs with the program.

References

Alani, H. (2003). Tgviztab: An ontology visualisation extension for protg. In *Proceedings of knowledge capture (k-cap'03) workshop on visualization information in knowledge engineering.*

Bosca, A., Bonino, D., & Pellegrino, P. (2005, December). Ontosphere: more than a 3d ontology visualization tool. In *Swap 2005, the 2nd italian semantic web workshop.* CEUR Workshop Proceedings.

Carriere, R., J.; Kazman.(1995, October). Research report. interacting with huge hierarchies: beyond cone trees. In *Information visualization, 1995. proceedings.* (p. 74-81). IEEE.

Ehrig, M., & Staab, S. (2004). *Qom quick ontology mapping* (Vol. 3298).

Eklund, P., Roberts, N., & Green, S. (2002). *Ontorama: Browsing rdf ontologies using a hyperbolic-style browser.*

Ernst, N. A., Storey, M., & Allen, P. (To Appear). Cognitive support for ontology modeling. *International Journal of Human-Computer Studies.*

Gruber, T. R. (1993). A translation approach to portable ontologies [Journal Article]. *Knowledge Acquisition*(5(2)), 199-220.

Kalfoglou, Y., & Schorlemmer, M.(2005). Ontology mapping: The state of the art. In Y. Kalfoglou, M. Schorlemmer, A. Sheth, S. Staab, & M. Uschold (Eds.), *Semantic interoperability and integration.* Internationales Begegnungs- und Forschungszentrum (IBFI), Schloss Dagstuhl, Germany. (<http://drops.dagstuhl.de/opus/volltexte/2005/40> [date of citation: 2005-01-01])

Klein, M., Fensel, D., Kiryakov, A., & Ognyanov, D.(2002, October). Ontology versioning and change detection on the web. In *Knowledge engineering and knowledge management. ontologies and the semantic web : 13th international conference, ekaw 2002, siguenza, spain, october 1-4, 2002* (Vol. 2473). Springer.

McGuinness, D. L., & Harmelen (Eds.), F. van. (2004, February). *Owl web ontology language overview.* W3C Recommendation.

Munzner, T., & Burchard, P.(1995). Visualizing the structure of the world wide web in 3d hyperbolic space. In *Proceedings of the first symposium on virtual reality modeling language.*

Parsia, B., Wang, T., & Golbeck, J. (2005). Visualizing web ontologies with cropcircles. *End User Semantic Web Interaction WS @ ISWC2005.*

Protege-Project. (2004). *Protege ontology editor and knowledge acquisition system website at: http://protege.stanford.edu/.*

Robertson, G. G., Mackinlay, J. D., & Card, S. K. (1991). Cone trees: animated 3d visualizations of hierarchical information. In *Chi '91: Proceedings of the sigchi conference on human factors in computing systems* (pp. 189–194). New York, NY, USA: ACM Press.

Sintek, M. (2005). *Ontoviz tab: Visualizing protege ontologies.* Accessed online. (http://protege.stanford.edu/plugins/ontoviz/ontoviz.html)

Storey, M., Musen, M., Silva, J., Best, C., Ernst, N., Fergerson, R., et al. (2001). Jambalaya: Interactive visualization to enhance ontology authoring and knowledge acquisition in protege. In *Workshop on interactive tools for knowledge capture.*

Tu, K., Xiong, M., Zhang, L., Zhu, H., Zhang, J., & Yu, Y. (2005). *Towards imaging large-scale ontologies for quick understanding and analysis* (Vol. 3729; Y. Gil, E. Motta, V. R. Benjamins, & M. A. Musen, Eds.). Springer-Verlag GmbH.

Ware, C. (2004). *Information visualization: Perception for design* (Second Edition ed.; S. Card, J. Grudin, & J. Nielsen, Eds.). San Francisco: Morgan Kaufmann Publishers.