Ontology Visualization Methods—A Survey

AKRIVI KATIFORI and CONSTANTIN HALATSIS

University of Athens

and

GEORGE LEPOURAS, COSTAS VASSILAKIS, and EUGENIA GIANNOPOULOU

University of Peloponnese

Ontologies, as sets of concepts and their interrelations in a specific domain, have proven to be a useful tool in the areas of digital libraries, the semantic web, and personalized information management. As a result, there is a growing need for effective ontology visualization for design, management and browsing. There exist several ontology visualization methods and also a number of techniques used in other contexts that could be adapted for ontology representation. The purpose of this article is to present these techniques and categorize their characteristics and features in order to assist method selection and promote future research in the area of ontology visualization.

Categories and Subject Descriptors: H.3.3 [**Information Search and Retrieval**]; H.5.2 [**Information Interfaces and Presentation**]: User Interfaces—*Graphical user interfaces (GUI)*; I.3.6 [**Computer Graphics**]: Methodology and Techniques—*Interaction techniques*

General Terms: Design

Additional Key Words and Phrases: Ontology, visualization method, human-computer interaction

ACM Reference Format:

Katifori, A., Halatsis, C., Lepouras, G., Vassilakis, C., and Giannopoulou, E. 2007. Ontology visualization methods—A survey. *ACM Comput. Surv. 39*, 4, Article 10 (October 2007), 43 pages DOI = 10.1145/1287620.1287621 http://doi.acm.org/10.1145/1287620.1287621

1. INTRODUCTION

Recently, the continuing progress in network technologies and data storage has made possible the digitization and dissemination of huge amounts of documents, making it more and more difficult for the user to successfully search and retrieve information

This work was supported in part by the Greek Secretariat for Research and Development under the PENED 2003 framework.

Authors' Addresses: A. Katifori and C. Halatsis, Department of Informatics and Telecommunications, University of Athens, Panepistemioupolos, Llissia, Athens, 157 84, Greece; email: vivi@mm.di.noa.gr; G. Lepouras, C. Vassilakis, and E. Giannopoulou, Department of Computer Science and Technology, University of Peloponnese, End of Karaiskaki, 22100, Tripolis, Greece.

both in the Web and in a digital document collection, personal or otherwise. The need for more effective information retrieval has lead to the creation of the semantic web and personalized information management notions, areas of study that take advantage of the semantic context of documents to facilitate their management. In many of the proposed solutions in this field, it is common to take advantage of an ontology. A term initially borrowed from philosophy, it is now used to denote a set of concepts and their interrelations in a specific domain. Consequently, the need for effective ontology visualization for design, management, and browsing has arisen.

Visualization of ontologies is not an easy task. An ontology is something more than a hierarchy of concepts. It is enriched with role relations among concepts and each concept has various attributes related to it. Furthermore, each concept most probably has instances attached to it, which could range from one or two to thousands. Therefore, it is not simple to create a visualization that will effectively display all this information and at the same time allow the user to easily perform various operations on the ontology.

In the field of ontology visualization, there are several works, mostly in 2D. Apart from the systems that propose visualizations especially tailored for ontologies, there are a number of other techniques used in other contexts such as graph or file system visualization, that could be adapted to display ontologies.

The purpose of this article is to present these techniques and categorize their characteristics and features in relation with a set of requirements compiled for an ontology visualization tool. Such an overview of techniques may be useful for choosing an ontology visualization for a specific application, taking into account both functional (e.g., navigation capabilities) and nonfunctional (e.g., ontology size) requirements as well as tasks that are related to the specific application.

The following sections provide an ontology definition, a detailed description of the techniques, followed by a discussion of their characteristics, and the conclusions.

2. ONTOLOGY DEFINITION

According to Gruber [1993], an ontology is an explicit specification of a conceptualization. The term "conceptualization" is defined as an abstract, simplified view of the world, which needs to be represented for some purpose. It contains the objects, concepts, and other entities that are presumed to exist in some area of interest, and the relations that hold among them. The term "ontology" is borrowed from philosophy, where an ontology is a systematic account of existence. For knowledge-based systems what "exists" is exactly that which can be (and has been) represented.

Therefore, as defined in Noy and McGuiness [2001], an ontology is a formal explicit description of concepts, or classes in a domain of discourse. Properties—or slots—of each class describe various features and attributes of the class, and restrictions on slots (called facets or role descriptions) state conditions that must always hold to guarantee the semantic integrity of the ontology. Each slot has a type and could have a restricted number of allowed values. Allowed classes for slots of type Instance are often called a **range** of a slot. An ontology along with a set of individual instances of classes constitutes a knowledge base.

A more mathematical definition can be the following [Amann and Fundulaki 1999].

An ontology is a triple O = (C, S, isa) where:

(1) $C = \{c_1, c_2, \ldots, c_m\}$ is a set of classes, where each class c_i refers to a set of real world objects (class instances),

(2) $S = \{s_1, s_2, \ldots, s_n\}$ is a set of slots, where each slot s_i could refer to:
   a. a property of a class: a value of a simple type such as Integer, String or Date
   b. a binary typed role: the representation of a relation between classes.

(3) $isa = \{isa_1, isa_2, \ldots, isa_p\}$ is a set of inheritance relationships defined between classes. Inheritance relationships carry subset semantics and define a partial order over classes, organizing classes into one or more tree structures.

In order to accommodate the individual instances, this definition can be extended with a fourth element $I = \{i_1, i_2, \ldots, i_q\}$, where each i_w is an instance of some class $c_x \in$ C. The instance includes a concrete value for every slot s_y associated with c_x or its ancestors (as defined by the *isa* set).

3. RELATED WORK

There are several works that review visualization techniques. They are not focused on ontologies, but attempt a more holistic view of techniques for visualizing many different types of data or documents. In Keim [2002], for example, apart from the categorization according to the type of data they support (e.g., text documents, images, processes, file system objects), techniques are divided into graphs, landscapes, dense pixel displays, and packed displays, from the visualization point of view, and in interactive projection, filtering, zooming, distortion, linking, and brushing from the interaction and distortion point of view. Young [1996] focuses mostly on 3D and distinguishes three general categories: mappings from the data domain to the visualization space (surface plots, cityscapes, etc.), information presentation techniques (perspective walls, cone trees, etc. and dynamic information visualization techniques (fish-eye views, self organizing graphs, etc.).

The Shneiderman [1996] framework categorizes visualization methods based on two criteria, the data-type of the objects to be represented in the interface (linear, planar, volumetric, temporal, multidimensional, tree, network, workspace) and the task typology (overview, zoom, filter, details-on-demand, relate, history, extract).

In another survey for 3D visualizations [Wiss and Carr, 1998] methods are examined from a cognitive point of view. Attention, abstraction and affordances are the cognitive aspects examined. Furthermore, designs are distinguished in node-link style designs (Cone Tree, Hyperbolic Space, etc.), Raised Surface Designs (Perspective Wall, Document Lens, etc.), Information Landscapes (FSN, Bead, Web Forager), and other designs (Web Book, Information Cube, etc.). In Herman et al. [2000], graph visualization techniques are presented and categorized from the graph drawing point of view. The Tao et al [2004] review approaches the issue of visualization from the point of view of Bioinformatics, including techniques for the presentation of the GO ontology [Gene Ontology Consortium www.go.org].

As there exist a number of ontology visualizations that are being used either in the context of ontology management tools or as information retrieval aids in applications that employ ontologies, some information on ontology visualization may be found in the ontology management tool surveys that can be retrieved from the Protégé Web pages [Protégé Project http://protege: Stanford.edu]. Ernst and Storey [2003] present the preliminary results of a survey using questionnaires related to ontology editing tools and ontology visualization.

However, up to this point, there are not many comparative evaluations concerning the effectiveness of ontology visualization methods for different tasks and with different user groups. One example of such an evaluation focused on ontology visualization evaluation in the context of a historical archive is Katifori et al. [2006a]. Its results have been taken into account for the discussion sections. Other evaluations like Kobsa [2004], which is focused on the presentation of hierarchies in file browsers, and Wiss et al. [1998], which evaluates three 3D visualization methods, have also been taken into account.

Table I. Equivalence of Document or File Categorization and Ontology Features

File system objects	Categorized documents	Ontology
Folder	Category	Entity (class or instance)
Folder/subfolder relationship	Category/subcategory relationship	isa-relationship
Tree view	Categorization	Taxonomy
File	Document	Instance
File properties	Document properties	Slots

This article is an attempt to summarize existing literature related to ontology visualization, provide comprehensive cataloguing of existing method characteristics as well as record their strong points and weaknesses in relation with user tasks.

4. VISUALIZATION TECHNIQUES GROUPING

The visualization techniques[1] presented in the following sections were either specifically created to display ontologies or were designed for other uses related to a tree or graph representation; for example for the visualization of a file system or a document categorization. Methods not created specifically for ontologies have been included because the focus of this work is not the presentation of all existing ontology management tools, but rather of existing ontology visualizations. To this end, selected visualization techniques from relevant areas could provide ideas and insight into the research on ontology visualization.

However, methods designed for other purposes probably need some modifications in order to be used for the visualization of ontologies. For a method to be eligible for the visualization of an ontology, it has to support the presentation of ontology ingredients; classes (or entity types), relations, instances, and properties (or slots). For example, a straightforward equivalence among file system objects, categorized documents, and ontologies is illustrated in the following table.

The methods can be grouped according to different characteristics of the presentation, interaction technique, functionality supported, or visualization dimensions. For the needs of this survey the methods were grouped in the following categories, representing their visualization type:

1. Indented list,
2. Node–link and tree,
3. Zoomable,
4. Space-filling,
5. Focus + context or distortion,
6. 3D Information landscapes.

Methods grouped in one of these categories may have elements of the other categories, for example, some space-filling techniques may also be zoomable. In these cases the predominant functionality features have been used for the categorization of the method. The effects of possible additional features on the performance of the visualization is presented in the respective discussion section.

This grouping was chosen as a starting point because each of these general categories of visualizations has characteristics that lead to different advantages and weak points. There is a need to investigate how those relate to the special requirements of an ontology visualization tool in relation to the tasks a user would like to perform with an ontology visualization tool.

[1]Visualization methods published until July 2006 have been considered.

The methods grouped in these six general categories were further categorized according to the number of space dimensions they employ: 2D or 3D. 2D methods use the screen space as a plane and do not use any notion of depth. 3D methods exploit the third dimension either to create visualizations that are closer to real world metaphors or to improve usage of space and/or usability. More specifically, these methods allow the user to manipulate—rotate and move—3D objects and/or to navigate inside the 3D space. 2 ½D is a term applied to 2D visualizations that use a perspective view in order to create a sense of 3D without allowing movement or manipulation in the third dimension. Methods of this category are presented with the 2D ones in this work.

This second-level grouping was chosen due to the specific needs that characterize the 3D visualizations that are also reflected in the interaction techniques employed, and functionality that can be catered for, target user group characteristics, and even system requirements. 3D visualization in general requires increased system resources in order for navigation and viewing to be smooth and without delays and, as a result, is probably not suitable for Web use. Furthermore, the 3D methods presented here employ more complex navigation methods and may be a little frustrating and disorienting for a novice user. This issue will be discussed in more detail in Section 12.

The following sections present the visualization techniques classified according to this two-level categorization scheme. Each section provides a brief overview of the methods pertaining to a specific category, followed by a summarization of the method's characteristics. The characteristics that are considered in these summarizations are presented in the following paragraphs.

As described in Section 3 an ontology is composed of several elements. These elements should be displayed in a way that the user could discern the information provided effortlessly, and are the following:

Classes. The visualization method should display all the ontology classes, at once or at the request of the user, providing at least their name, in an intelligible manner.

Instances. The instances are the actual data associated with the ontology and in most cases what the user is actually interested in. However, representing them as nodes connected to a class is not always effective because of their great number, so other alternatives should be used, like presenting the instances of a selected class as a list within a separate window.

Taxonomy (Isa relations). The presentation of the taxonomy on which the ontology is based is essential for understanding the inheritance relations between classes. The system should at least provide a holistic view of this taxonomy, in a hierarchical representation. Partial views, allowing the user to focus on a portion of the taxonomy, are also a desirable feature.

Multiple inheritance. The cases where a class has more than one parent are not easy to represent in combination with an effective representation of the taxonomy. It is desirable for the visualization to indicate nodes with multiple parents and provide efficient means to view all direct ancestors of a node. It should be noted here that many of the presented ontology visualizations support multiple inheritance by replicating child nodes under all their parents. Hierarchical visualizations that currently do not support this feature could be adapted to support it.

Role relations. Role relations are essential, but like the multiple inheritance links, not easy to represent. Apart from the link that should be visible, a label with the link name (effectively, the role type) should also be displayed (possibly with the option to hide it, to avoid display cluttering). Multiple inheritance and role relations are two types of links that transform the ontology from a hierarchy to a graph, a structure inherently more difficult to represent than a tree.

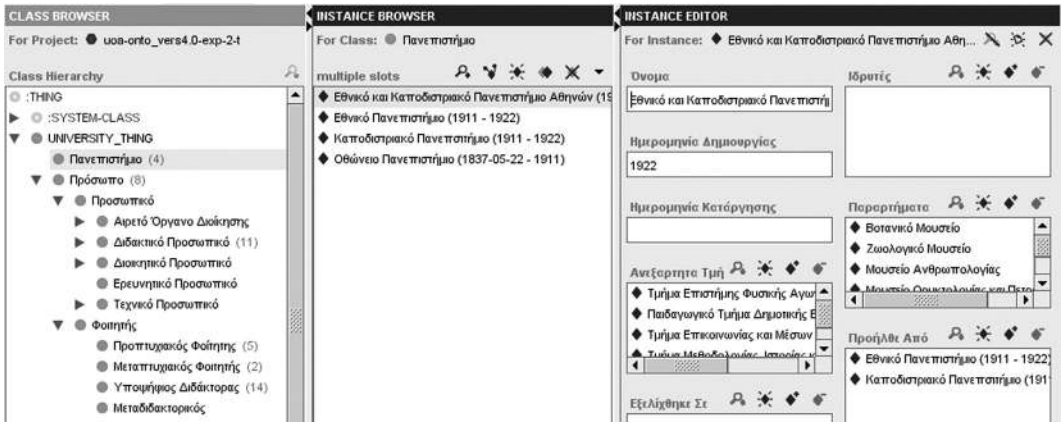

Fig. 1. The Protégé class browser.

Properties. The properties associated with an entity are also very important and a complete visualization should include their representation, either on the main ontology visualization or within separate space.

Apart from these ontology presentation characteristics, two more are added. These are keyword search and software availability. Although these characteristics are not directly relevant to the ontology visualization itself, but rather to the tool that contains it, they may be informative in case the reader would like to use the method, improve it, or add it in an existing application.

A key issue to be taken into account when evaluating the efficiency of an ontology visualization method is that of the specific user tasks that the visualization method is expected to support. Section 13 presents a detailed categorization of tasks, based on the top level task analysis proposed by Shneiderman [1996], along with a commentary on the suitability of each presented method in relation with these tasks. This analysis proposes overview, zoom, filter, details on demand, relate, history, and extract as general tasks that may be preformed with the visualization tool.

In the rest of this document, Sections 5-10 present the six visualization method categories. For each category, a brief description is given, followed by a short presentation of individual methods of the pertinent category; each section is concluded with a table summarizing the characteristics of the methods presented therein. In these tables, names of the methods that were designed especially for ontologies are denoted with an asterisk (*).

Subsequently, Section 11 presents issues related to visualization of evolution and time in the context of ontologies, while Sections 12–17 discuss advantages and disadvantages of method categories and characteristics, with regards to different criteria. Finally, Section 18 concludes the article and outlines future work.

5. INDENTED LIST

Most of the ontology visualization systems, like Protégé [Noy et al. 2000], OntoEdit [Sure et al. 2002], Kaon [Kaon, http:// kaon.semanticweb.org] and OntoRama [Eklund et al. 2002], along with their main visualization technique, offer a Windows Explorer-like tree view of the ontology. In this view, the taxonomy of the ontology (as dictated by the *isa* inheritance relationships) is represented as a tree (Figure 1). The features provided for Protégé Class Browser in Table II are common for the other implementations

Table II. Indented List Visualization Characteristics Summary. The Asterisk (*) Indicates that the Method has been Used for Ontology Visualization

Method	Classes and Instances	Taxonomy	Multiple Inheritance	Role Relations	Properties	Keyword Search	Software Availability
Protégé Class Browser (*)	Classes are presented as nodes in an indented, expandable and retractable tree. Instances are displayed in a separate window.	Child classes are placed under their parent and indented to the right	Child nodes are placed under both parents.	No. Supported through the properties window only.	Properties are displayed in a separate window	Available only for the already visible nodes in the class and instance windows	Open–Source, available at [Protégé]

in Kaon, OntoEdit and Ontorama, although they offer a more comprehensive search feature than Protégé Class Browser.

6. NODE—LINK AND TREE

This category of techniques represents ontologies as a set of interconnected nodes, presenting the taxonomy with a top–down or left to right layout. The user is generally allowed to expand and retract nodes and their subtrees, in order to adjust the detail of the information shown and avoid display clutter.

6.1. Two Dimensional

OntoViz [Sintek 2003] is a Protégé [Protégé Project http://protege.stanford.edu] visualization plug-in using the GraphViz [GraphViz http://graphViz.org] library to create a very simple 2D graph visualization method. The ontology is presented as a 2D graph (Figure 2) with the capability for each class to present, apart from the name, its properties, and inheritance and role relations. The instances are displayed in different colors. It is possible for the user to choose which ontology features will be displayed, as well as to prune parts of the ontology from the Config Panel on the left. Right-clicking on the graph allows the user to zoom-in or zoom-out.

IsaViz [Pietriga http://www.w3.org/2001/ii/IsaViz] is a visual environment for browsing and authoring RDF ontologies represented as directed graphs. Graphs are visualized using ellipses, boxes, and arcs between them (Figure 3). The nodes are class and instance nodes and property values (ellipses and rectangles respectively), with properties represented as the edges linking these nodes.

SpaceTree [Plaisant et al. 2002] is a tree browser that builds on the conventional node-link tree diagrams by substituting branches that cannot be fully opened with a preview icon. In the current initial design this icon is an isosceles triangle, the shading of which is proportional to the total number of nodes in the subtree. Its height represents the depth, and the base the average width. Layout adjustments and orientation change are available as an option.

The TreePlus visualization [Lee et al. 2006a] focuses on supporting localized and rapid browsing and easy reading of labels. It proposed the “Plant a seed and watch it grow” metaphor which allows the user to explore the hierarchy or graph starting from

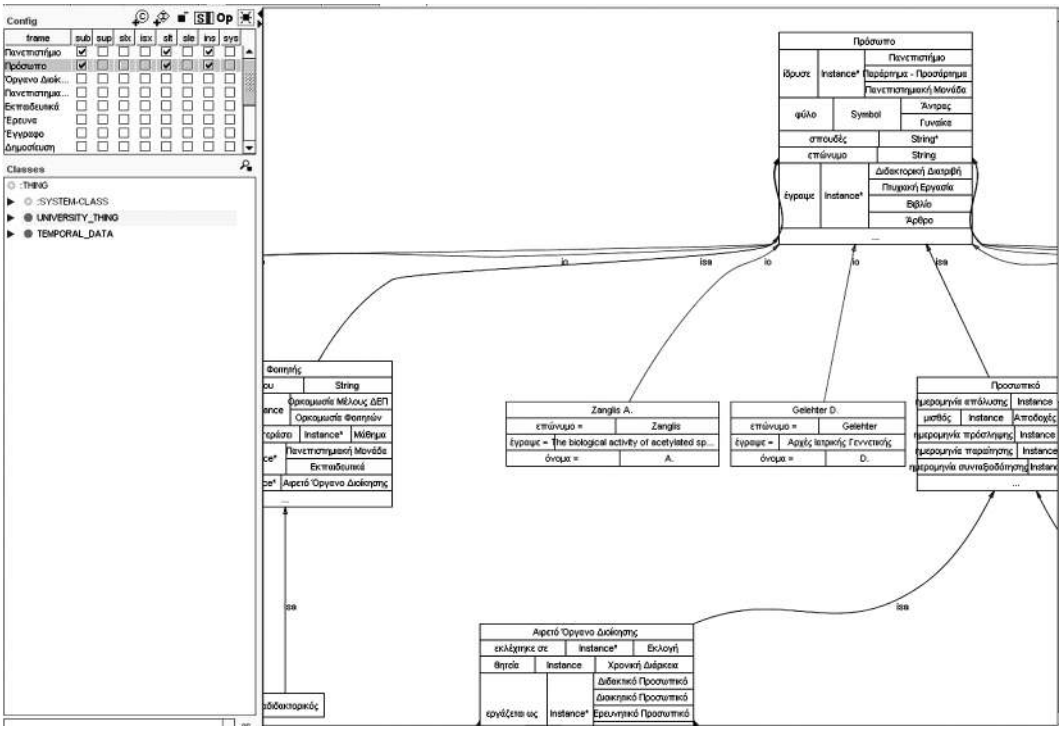

Fig. 2. Protégé OntoViz visualization.

a specific node. It uses a left-to-right tree layout in combination with expansion and retraction of nodes and node highlighting.

OntoTrack [Liebig and Noppens 2004] is a browsing and editing "in-one-view" authoring tool with a hierarchical layout. It resembles the SpaceTree visualization as it represents retracted subhierarchies with triangles of length, width, and shading that approximate depth, branches, and number of subclasses. As an extra feature, it provides an interface with an external OWL reasoner.

GoSurfer [GoSurfer http://www.gosurfer.org], [Zhong et al. 2004a, 2004b] is a data mining tool for visualizing GO [Gene Ontology Consortium http://www.go.org] associated with specific genes given as input. It uses a common, top down tree visualization and tools for comparing genes in relation to their corresponding terms in the GO ontology: comparing ontology paths.

The **GOBar** visualization [GOBar http://Katahdin.csh.org: 9331/GO], [Lee et al. 2005] is based on the GraphViz [GraphViz http://graphviz.org] library to create an ontology for visualizing GO [Gene Ontology Consortium http://www.go.org]. **GOMiner** [GOMiner http//:discover.nci.nih.gov/gominer] uses a similar top down graph to represent the GO ontology hierarchy.

6.2. Three Dimensional

A special type of a 3D graph is the 3D tree **Cone Tree** [Robertson et al. 1991], with its nodes arranged at the base of a cone and its parent at the top of the cone. That way a subtree is represented as a cone containing subcones. The cones are semitransparent, creating a visible structure and at the same time providing an outline of the

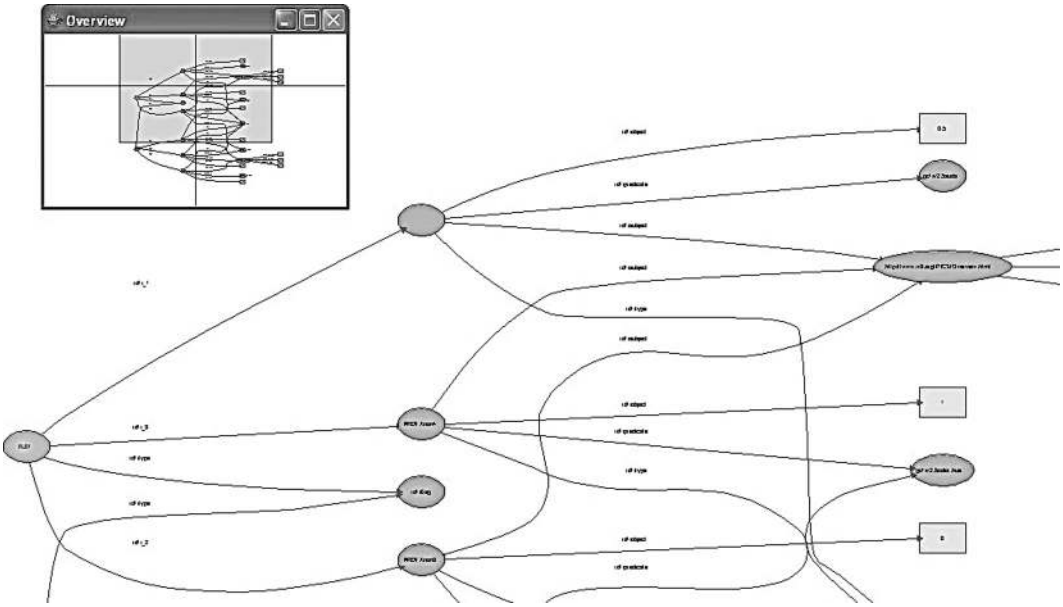

Fig. 3. IsAviz: graph with the radar view visible.

background nodes. When a node is selected, the cone to which it belongs is rotated to bring the selected node to the front. Similarly, the predecessors of the selected node are brought to the front. The speed of rotation has been set so as to allow the user to watch the transition. Cone trees may be presented horizontally or vertically. An interesting feature is the use of the tree shadow in order to provide a 2D overview of the hierarchy.

Carriere and Kazman [1995], proposed an enhanced version of the Cone Tree, **fsviz**, with several features such as dynamic queries, coalescing of distant nodes into a single graphical representation, node size, and frequency of usage queries.

The **Reconfigurable Disk Tree** [Jeong and Pang 1998] is an extension to the Cone Tree that allows the user to change the height of each subtree cone in order to improve the visibility of the nodes. The base of the cone, which contains the nodes, may become larger or smaller, according to the number of nodes it contains. As a result, the user may arrange the subtrees so as to make better use of the available space.

The **Tree Viewer** [Kleiberg et al. 2001] visualizes trees in the form of a real-world tree. The hierarchy root is the tree stem and its children are branches (multiple sub-hierarchies of a node branch off one by one). Terminal nodes are "bulbs" at the end of the branches and instances are disc-shaped "fruits" on top of the bulbs. Instances and classes at the same level are displayed in the same color. Users can move and rotate the tree and zoom in and out. They can also change the colors of the tree, leaves, branches, and the background, and customize the general appearance of the tree.

OntoSphere [Bosca et al. 2005] proposes a node-link tree type visualization that uses three different ontology views in order to provide overview and details according to the user's needs. The RootFocus Scene (Figure 4a) presents a sphere bearing and on its surface a collection of the upper level classes represented as small spheres. It does not visualize the taxonomy, but the direct role relations between classes. Color and size coding is used to denote the existence of subtrees and their size. The user may right-click on a class to display the RootFocus View of its children. The TreeFocus Scene (Figure 4b), displayed when left-clicking on a class, shows the selected class

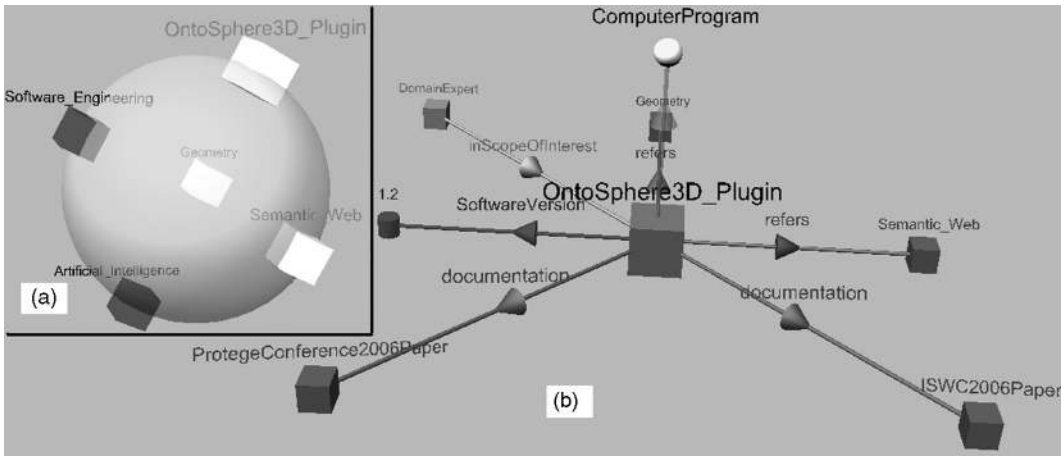

Fig. 4. OntoSphere visualization (a) Root Focus view (b) TreeFocus view.

with its subtree. Only three levels down from the selected node are shown expanded. ConceptFocus Scene depicts all the information about the selected class, like ancestors, children, and semantic relations.

Table III summarizes the characteristics of the node-link and tree visualizations.

7. ZOOMABLE VISUALIZATIONS

This category contains all the methods that present the nodes in the lower levels of the hierarchy nested inside their parents, and with smaller size than that of their parents. These techniques allow the user to zoom-in to the child nodes in order to enlarge them, making them the current viewing level.

7.1. Two Dimensional

Grokker [Rivadeneira and Bederson 2003], [Grokker http://www.groxis.com] is a system for the display of knowledge maps. It offers graphical representation of information like the results of a search engine or a file search in general. The clustering mechanism presents the documents as a series of nested Venn diagrams (Figure 5). Users may navigate in the hierarchy by clicking on a circle. When a circle is selected, it is magnified with the use of animation, making its contents visible. Circles filled with color suggest that they include lower levels of the hierarchy. Transparent circles suggest that they are the lower level of the hierarchy. From the lower level of the hierarchy, users may select documents to view their contents on a larger window.

Jambalaya [Storey et al. 2001] is a visualization plug-in for the Protégé ontology tool [Noy et al. 2000, Protégé Project http://protege.stanford.edu] that uses the SHriMP (Simple Hierarchical Multi-Perspective) [Wu and Storey 2000] 2D visualization technique. SHriMP uses a nested graph view (Figure 6) and the concept of nested interchangeable views. It provides a set of tools including several node presentation styles, configuration of display properties and different overview styles.

CropCircles [Parsia et al. 2005; Wang and Parsia 2006] is an ontology visualization that represents the class hierarchy tree as a set of concentric circles (Figure 7). Nodes are given the appropriate space in order to guarantee enclosure of all the subtrees. If there is only one child, it is placed as a concentric circle to its parents, otherwise the child-circles are placed inside the parent node from the largest to the smallest. The user

Table III. Node-Link Tree Visualization Characteristics. The Asterisk (*) Indicates that the Method has been Used for Ontology Visualization. "No$^+$" Under Multiple Inheritance Means that the Tool Currently does not Support Multiple Inheritance through Node Replication, but could be Extended to Accommodate Such Support

Method	Classes and Instances	Class Hierarchy	Multiple Inheritance	Role relations	Properties	Keyword Search	Software availability
OntoViz (*)	Rectangle nodes with different color for classes and instances	The child nodes are placed under the parent ones and linked with an "isa" link	The child node is linked with all the parents	They are represented with labeled links	Properties are displayed on the node	No	Open Source, available as a Protégé [Protégé Project] plug-in
GOBar (*)	Nodes are presented as ellipsis or rectangles	The child nodes are placed under the parent ones	The child node is linked with all the parents	No	Properties are displayed in a separate window when the cursor is placed on the node	Yes. Matching nodes are highlighted. Filtering of nodes is also possible.	Freely available as a web – based tool [GOBar]
IsAviz (*)	Classes and instances are represented as labeled ellipses	Nodes are linked to their parents. An overview (Radar View) is provided apart from the focused view	The child node is linked with both the parents	They are represented with labeled links	Property values are displayed as rectangle nodes linked to the instance with a link labeled with the name of the property or in a separate window	Yes	Open source, available in Pietriga. Possibility to create plug-ins
SpaceTree	Tree nodes are rectangles containing a label	The child nodes are placed under the parent ones, some subtrees may be substituted by their preview icon	No$^+$	No	No	Yes. Matching nodes are highlighted. Dynamic Queries are also supported, providing node filtering	Available under license at SpaceTree
TreePlus	Tree nodes are rectangles containing a label	Nodes are linked to their parents.	The child node is linked with both the parents	They are represented with labeled links	No	Yes	
OntoTrack (*)	Tree nodes are rectangles containing a label	The child nodes are placed under the parent ones, some subtrees may be substituted by their preview icon	There are links to all the node parents	No	Properties are presented in hierarchies in another view with the option to render them as a transparent read-only layer with the class hierarchy	Yes. Matching nodes are highlighted.	Available under non commercial license at OntoTrack
GOSurfer (*)	Represented as tree nodes. Selected nodes are marked with numbers with their labels	Nodes are placed under their parent nodes.	No$^+$	No	Properties are displayed in a separate window	No. Filtering is performed before the display of the tree.	Freely available at GoSufer

Continues

Table III. *(Continued)*

Method	Classes and Instances	Class Hierarchy	Multiple Inheritance	Role relations	Properties	Keyword Search	Software availability
	listed underneath the tree structure.						
GOMiner (*)	Represented as rectangle tree nodes.	Nodes are placed under their parent nodes.	There are links to all the node parents	No	Properties are displayed as tooltips on mouse over	No	Freely available at GOMiner
Cone Tree	Represented as labeled nodes	Child nodes are placed at the circumference of the base of the cone with the parent as the cone apex	No$^+$	No	No	No	Available upon request to its authors
fsviz	Represented as 3D shapes	Child nodes are placed at the circumference of the base of the cone with the parent as the cone apex	No$^+$	No	No	Yes. Dynamic queries are supported	No
Reconfigurable Disk Tree	Represented as nodes	Child nodes are placed at the circumference of the base of the cone with the parent as the cone apex. The radius of the cone is configurable.	No$^+$	No	No	No	No
Tree Viewer	Classes are represented as branches, child-less classes as bulbs and instances as disks on top of the bulbs	Child nodes branch off their parents. Instances are placed on top of their parent classes	No$^+$	No	No	No	No
OntoSphere (*)	Classes and instances are represented as spheres	In the TreeFocus View child nodes are placed under their parent.	The child node is connected to both its parents in TreeFocus View.	In Concept-Focus View links are used to denote role relations.	No	No	Available as a Protégé plug-in in OntoSphere

may click on a circle to highlight it and see a list of its immediate children on a selection pane. The selection pane can let the user drill down the class hierarchy level-by-level and it also supports user browsing history. The user may also select which top level nodes to show in the visualization.

7.2. Three Dimensional

In **Information Cube** [Rekimoto and Green 1993] nested and semitransparent cubes are used in order to provide to the user a view of the categories further down in the hierarchy. This transparency is gradually reduced in the inner cubes because otherwise the view would become cluttered. A label is placed on the surface of each cube and the

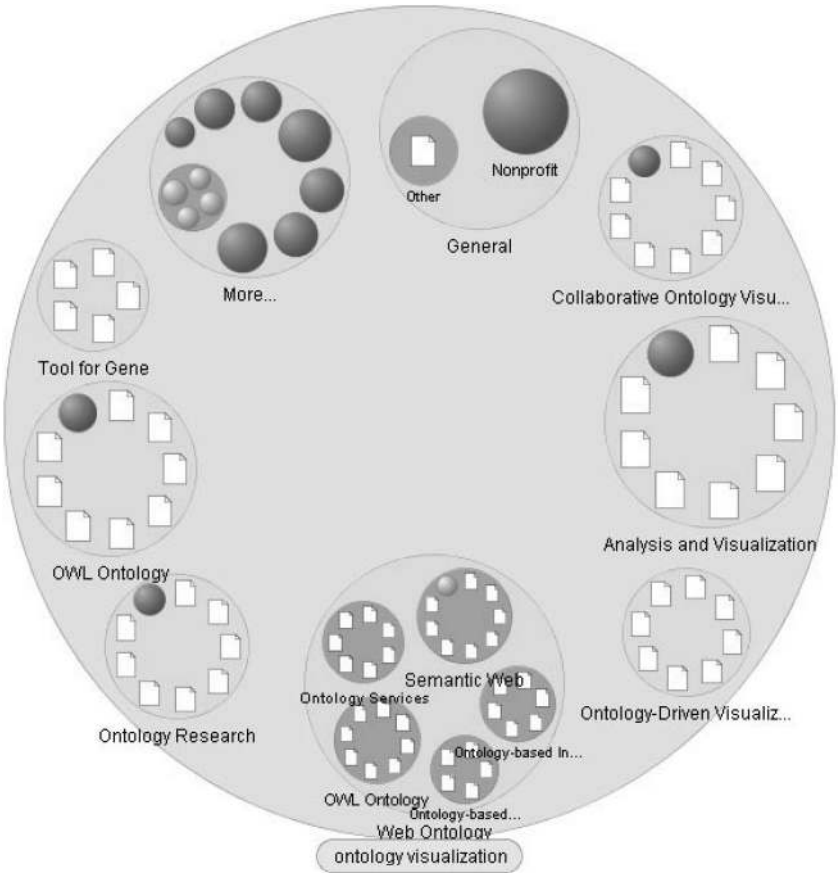

Fig. 5. Grokker. visualization of the results of a Web search on "ontology visualization."

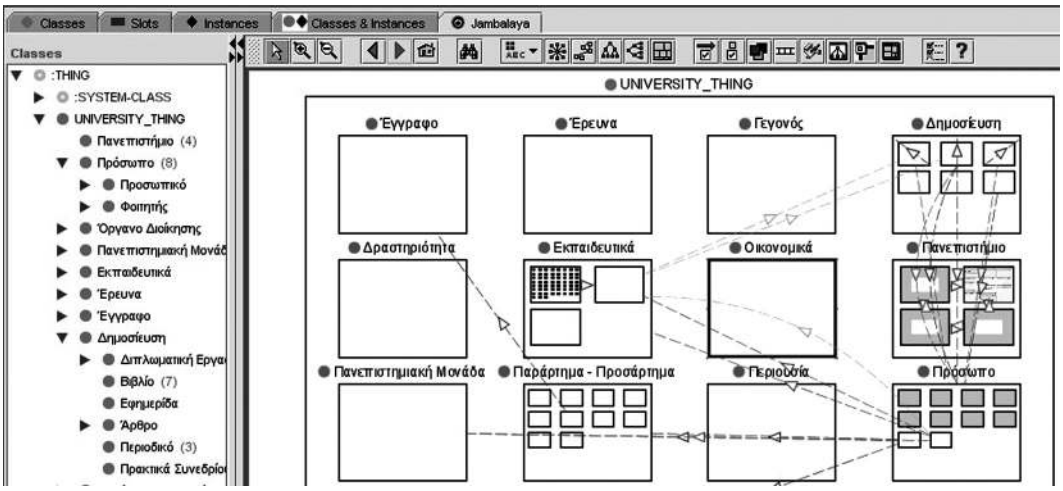

Fig. 6. The Jambalaya tab in Protégé with class browser on the left.

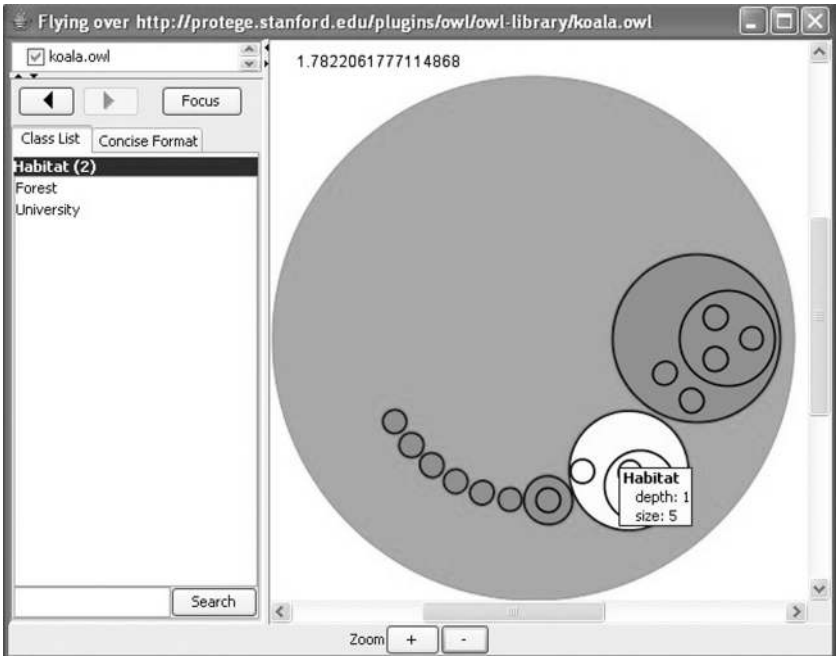

Fig. 7. TheCropCircles visualization in Swoop. The "Habitat" node is selected and its label visible on mouse over.

leaves (in the ontology case, the instances), are represented as 2D plates with their label on their surface.

In **Information Pyramids** [Andrews et al. 1997; Andrews 2002], the hierarchy is represented by pyramids that have a flattened top and are placed one on top of the other. In this case, the subcategories are placed on top of the broader category pyramids as smaller ones. If the category contains leaf nodes, they are represented as small rectangle objects placed on one side of the top of the pyramid. This layout is used recursively for all hierarchy levels.

The icon that represents the leaves may be color- or size-coded to represent certain properties. The user may focus on the parts of the hierarchy she/he wishes and have an overview of the hierarchy, as viewed from the top.

Gopher VR [Gopher VR; Andrews et al 1997] is a visualization created for Gopher, one of the first systems to easily access multimedia documents on the Internet. The nodes are 3D objects that are placed on a plane, but each time only the objects belonging to the current level of the hierarchy are displayed. By clicking on a node, its contents are displayed. The user may focus on a node or rotate around the center by using the buttons at the bottom of the screen. By choosing "Overview" the viewpoint is automatically moved to a position above the level to provide an overview of its contents. With "Up" and "Down" the viewpoint moves away from or closer to the nodes, respectively. An interesting available navigation method is bouncing, using the middle mouse button.

Table IV summarizes the characteristics of zoomable visualizations.

8. SPACE FILLING

Space filling techniques are based on the concept of using the whole of the screen space by subdividing the space available for a node among its children. The size of each

Table IV. Zoomable Visualization Characteristics. The Asterisk (*) Indicates that the Method has been Used for Ontology Visualization. "No^{+}" Under Multiple Inheritance Means that the Tool Currently does not Support Multiple Inheritance through Node Replication, but could be Extended to Accommodate Such Support

Method	Classes and Instances	Class Hierarchy	Multiple Inheritance	Role relations	Properties	Search and Filtering	Software availability
Grokker	Represented as colored circles	Lower level nodes are represented by smaller size circles and placed inside their parent nodes	No^{+}	No	Properties are displayed on a separate window	Yes	Trial version available, commercially available as a file browser tool [Grokker]
Jambalaya (*)	Represented as rectangles inside their parent node	Lower levels are represented by smaller size rectangles and placed inside their parent nodes	Child nodes are placed under both parents.	Supported through the properties and as directed links with their label visible as a tooltip	Properties are displayed as an embedded form if the selected node is zoomed - in	Yes, with the possibility to select the type of the searched item and search between the results	Open Source, available as a Protégé plug-in
CropCircles (*)	Concentric Circles	Lower level nodes are represented by smaller size circles and placed inside their parent nodes	Child nodes are placed under both parents. When a node is selected, all its appearances in the graph are highlighted.	Links between related nodes, offered as a user option	No. Only supported by the tool (Swoop) CropCircles is currently embedded	No. Only supported by the tool (Swoop) CropCircles is currently embedded	Preliminary Java version available at CropCircles
Information Cube	Classes represented as cubes and instances as 2D plates	Lower level nodes are placed inside their parent node and represented by smaller size and more transparent	No^{+}	No	No	No	No
Information Pyramids	Classes represented as pyramids with flattened top and instances as rectangle objects	Lower levels are represented by smaller size on top of their parent node	No^{+}	No	Properties are displayed in a separate window	Yes	No
GopherVR	Nodes are labeled rectangles on a plane, arranged in a circular pattern around a pyramid	Only one hierarchy level is visible. The central pyramid gives access to the parent level	No^{+}	No	Properties are displayed in a separate window	No	Available at [GopherVR]

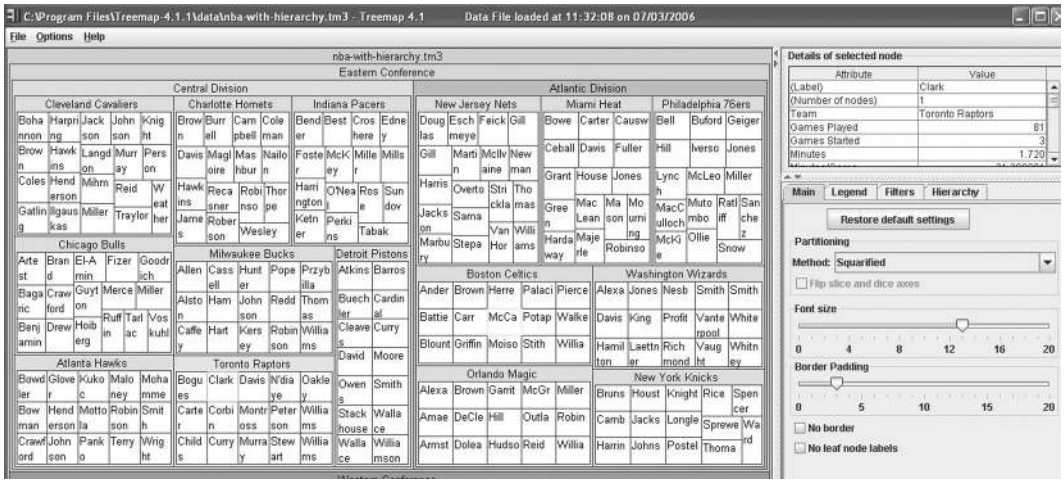

Fig. 8. Treemap with path to instance "Toronto Raptors" highlighted.

subdivision corresponds to a property of the node assigned to it—its size, number of contained nodes, and so on.

8.1. Two Dimensional

The **TreeMaps** [Shneiderman 1992] visualization method uses a 2D approach of space filling to represent hierarchies, using a rectangular area with rectangular subdivisions (Figure 8).

The Treemap technique has been proposed by Baehrecke et al. [2004] and Babaria [2004] as a tool for visualizing the GO ontology [Gene Ontology Consortium http://www.go.org]. Size and color are used to provide a mechanism to evaluate data. Treemap 4.0 has the functionality to assign labels, size, and color to different gene attributes. Moreover, the user may zoom on details by double-clicking on an area of interest so that the area selected is rapidly updated and may query data in the context of the entire GO classification.

SequoiaView [SequoiaView http://www.win.tue.nl/sequoiaview] visualizes trees in a similar manner as TreeMap. It goes beyond Treemap though, by supporting a $2\frac{1}{2}$D appearance of the segments through shading and spotlighting. It combines the Cushion Treemaps [Van Wijk and Van De Wetering 1999] shading with the Squarified Treemaps [Bruls et al. 2000], which uses rectangles with a smaller aspect ratio.

The **Information Slices** [Andrews and Heidegger 1998] technique uses one or more semicircular disks to more compactly visualize large hierarchies in 2D space. Each disk represents multiple hierarchy levels; typically in each disk 5–10 levels are represented—a, number that may be configured by the user. In deeper hierarchies, the child—nodes use subdivisions of the available space, depending on their size. Figure 9 presents a view of the system when a slice of the left disk, which corresponds to a child node, is expanded to the right.

8.2. Three Dimensional

BeamTrees [Van Ham and Van Wijk 2000] features both a space-filling Treemap-like visualization and a 3D node-link visualization. Overlapping beams are used to represent the hierarchy. Users can rotate and magnify the display, brush files and

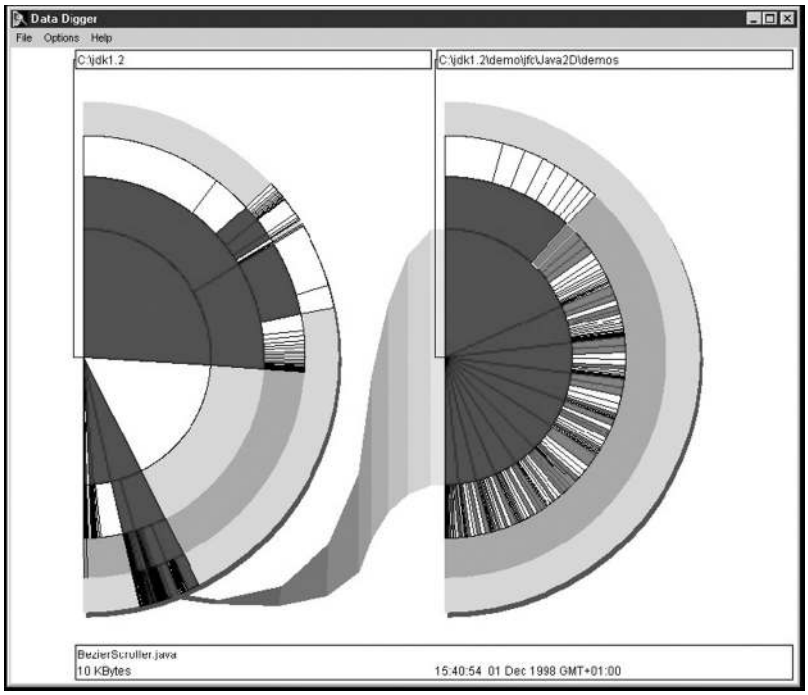

Fig. 9. Information Slices. A selected node is expanded to the right.

folders to obtain information about them, change the proportions of the visualized objects, and change the color scheme.

Table V summarizes the characteristics of space-filling visualizations.

9. CONTEXT + FOCUS AND DISTORTION TECHNIQUES

This group of techniques is based on the notion of distorting the view of the presented graph in order to combine context and focus. The node on focus is usually the central one and the rest of the nodes are presented around it, reduced in size until they reach a point that they are no longer visible. Usually a hyperbolic equation is used to this end. The user has to focus on a specific node, in order to enlarge it.

9.1. Two Dimensional

In Souza et al. [2003], a **2D hyperbolic tree** is used in order to present the ontology of the Brazilian Agricultural Research Society.

The hyperbolic tree technique is based on a hyperbolic transformation. The root of the tree is initially placed in the middle of a circular area with the child nodes around it, their child nodes placed around them and so forth. Moving from the center of the tree to the circumference the distance between the tree levels is diminished so that, as a result of the hyperbolic transformation, the whole tree fits in the circular area. The outer nodes, when smaller than a pixel, are not displayed. The technique is therefore based on distortion to keep the visualization within certain limits and combine detailed presentation within the information context. Another commercially available hypertree visualization is the **StarTree** [StarTree http://www.inxight.com; Lamping and Rao 1996].

Table V. Space-Filling Visualization Characteristics. The Asterisk (*) Indicates that the Method has been Used for Ontology Visualization. "No^{+}" under Multiple Inheritance means that the Tool Currently does not Support Multiple Inheritance through Node Replication but could be Extended to Accommodate Such Support

Method	Classes and Instances	Class Hierarchy	Multiple Inheritance	Role relations	Properties	Search and Filtering	Software availability
TreeMap 4.0 (*)	Nodes are represented as colored squares of size proportional to a selected property. Labels are displayed up to a certain depth.	Lower level nodes are placed inside their parent nodes	No^{+}	No	Yes	Filtering grays out nodes or omits subhierarchies	Available commercially and as a free demo version in Treemap
SequoiaView	Nodes represented as colored squares of size proportional to a selected attribute and having a 2 1/2 D appearance	Lower level nodes are placed inside their parent nodes	No^{+}	No	No	Yes. Also filtering that omits filtered files	Available at SequoiaView
Information Slices	Represented as segments of disk rings	Nodes are placed on the same sector with their parent, on the next outer ring or on a new, cascading disk	No^{+}	No	No	No	No
BeamTrees	Class nodes are represented as circular beams and instances as beam slices	Child classes overlap their parents and instances are slices of their class beam	Beams overlap all their parents	No	No. Limited information available through tooltips	No	Publicly available at BeamTrees

OntoRama [Eklund et al. 2002; Eklund 2002; Ontorama http://www.ontorama.com] is a Java application used for browsing the structure of an ontology with a hyperbolic–type visualization. Ontorama currently does not support "forest structures," which are subhierarchies, neither directly nor indirectly connected to the root. It uses cloning of nodes that are related to more than one node, in order to avoid cases where the links become cluttered. It can support different relation types. Apart from the hyperbolic view, it also offers a windows explorer–like tree view.

The **MoireGraphs** [Jankun and Kwan 2003] visualization attempts to combine a graph topology that supports focus and context with a set of interaction techniques for graph exploration, especially for graphs having a visual content that should be displayed (e.g. images, documents etc).

This technique uses radial graphs. In these graphs the focused node appears in the center while the nodes related to it are placed around it. Every next level of nodes away from the central one corresponds to an outer concentric circle. A set of interaction methods has been added to this static visualization to support quick navigation in the graph, movement and focus on selected nodes and comparison between nodes. Some of these interaction techniques are the adjustment of focus strength, graph rotation, navigation using animation and highlighting of a specific level.

TGVizTab (TouchGraph Visualization Tab) [Alani 2003] incorporates the TouchGraph [http://www.touchgraph.com] visualization technique in the Protégé [Protégé Project http://www.protege.stanford.edu] ontology management tool. TouchGraph is an open source Java environment for the creation and navigation of network graphs, also employed by the Kaon [http://Kaon.semanticweb.org] ontology management tool. It uses a spring–layout technique where nodes repel one another, whereas the edges (links) attract them. This results in placing the semantically similar nodes close to one another. A characteristic of this technique is that it is especially interactive, as the nodes move and adjust to the user commands.

This visualization allows the user to navigate gradually making visible parts of the graph. A variable radius of visibility is used to limit the size of the graph in smaller, more manageable sizes. The user may also expand or retract nodes, hide them, and change the node on focus by double clicking on it. Furthermore, she/he has full control of the color and visibility of the links and may change the zoom level or make the graph hyperbolic.

Figure 10 presents the interface of the TGVizTab. The ontology is presented as a tree structure on the left (Class Browser). In order to create the visualization on the right, a class or instance should be selected as a starting focal point.

The **Bifocal Tree** [Ricardo et al. 2002] is a visualization technique based on the focus + context concept, but uses two foci instead of one. It displays the hierarchy as a node-edge diagram separated in two connected sub-diagrams, the focus area, which corresponds to the sub-tree with the node of interest as root and the context area, which contains the selected node parent and remaining sub-trees.

OZONE (Zoomable Ontology Navigator) [Suh and Bederson 2002] is a visual interface for searching and browsing ontological information. OZONE visualizes query conditions and provides interactive, guided browsing for DAML (DARPA Agent Markup Language) ontologies. OZONE reads ontology information and rearranges it visually with context information so that ontology information can easily be queried and browsed without knowledge of their structure. Queries can be formulated interactively and incrementally by manipulating objects on the screen.

For example, if the user wants information about people, she/he begins to form a query by selecting the "Person" class from a class list that contains all classes of the ontology. This action puts the "Person" class on the display. Since the goal of the query is to find information about people in a particular research group, the user scans the

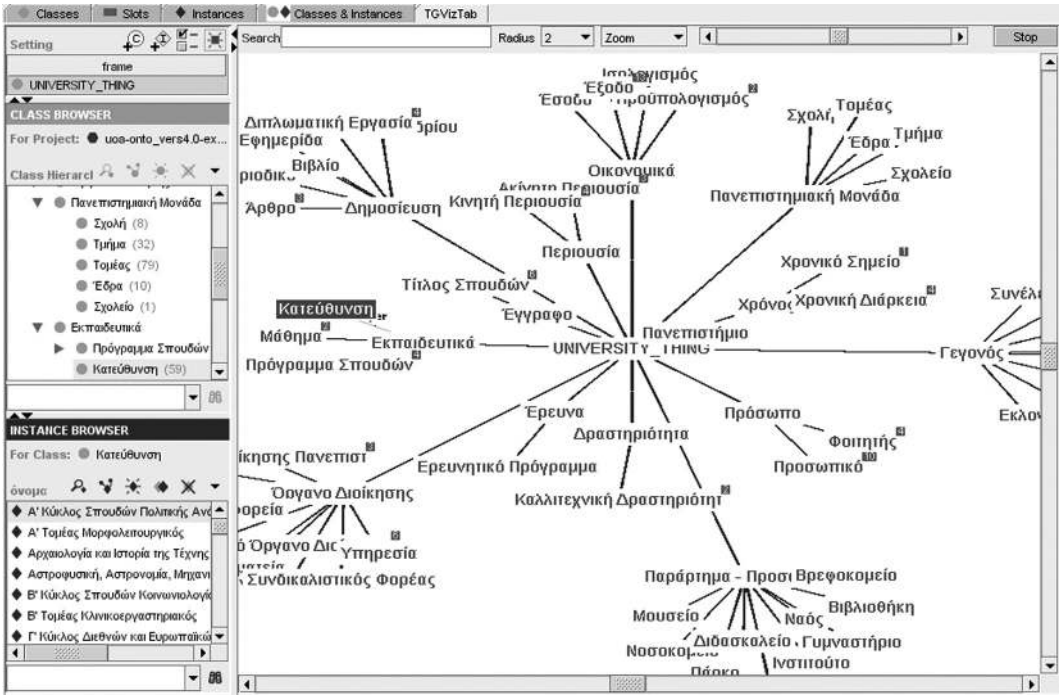

Fig. 10. Protégé TGVizTab.

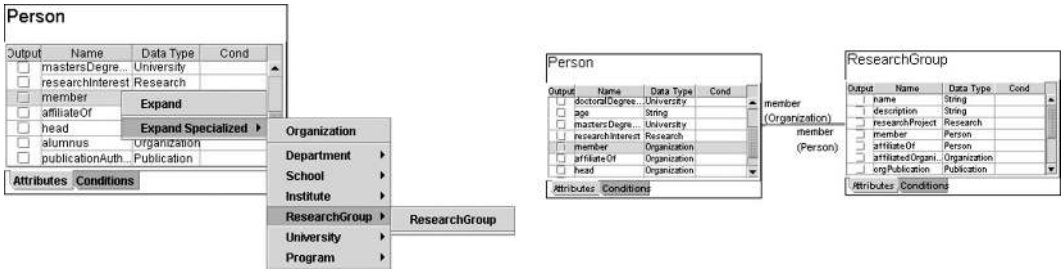

Fig. 11. Selecting a property (left), and the expanded node (right).

properties of the "Person" to find a property that relates a person with an organization (Figure 11). The user clicks the "member" property of the visual node because she/he finds that the most appropriate property to specify "is a member of" relationship. When the user clicks, a pop-up menu appears.

In OZONE, any subgraph can be grouped and transformed into a single node by choosing the 'Group' menu in the main menu after selecting nodes on the screen. The collection of nodes is zoomed out and a simple new node replaces the collection. The user can access the detailed sub nodes at any time by zooming in.

9.2. Three Dimensional

The **3D Hyperbolic Tree** [Munzner 1997, 1998] visualization was created for Web site visualization but has been used as a file browser as well. It presents a tree in the 3D

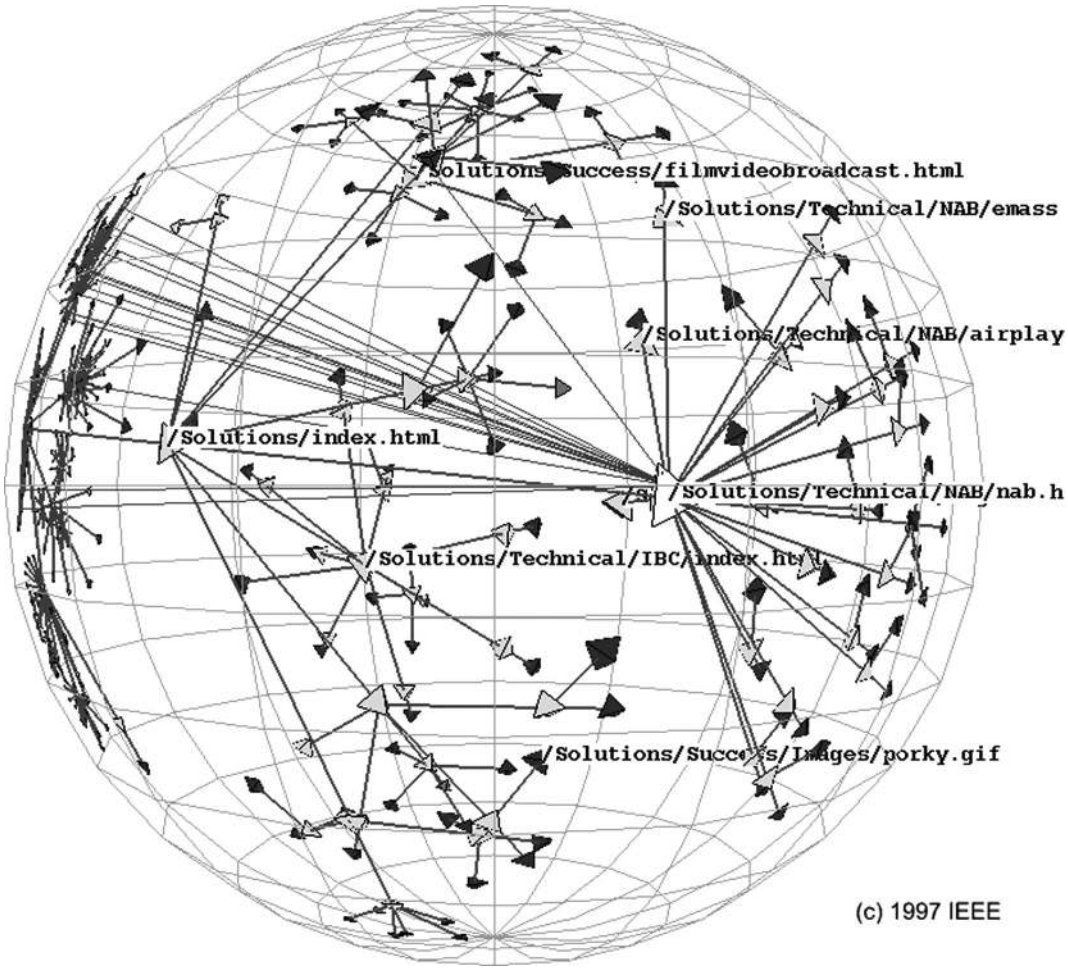

Fig. 12. Hyperbolic Tree.

hyperbolic space in order to achieve greater information density. The nodes of the tree are placed at a hemisphere of a sphere. Figure 12 presents the whole structure of a 3D hyperbolic tree. It offers animated transitions when changing the node on focus.

Table VI summarizes the results for context + focus and distortion visualizations.

10. INFORMATION LANDSCAPES

A very common metaphor used in VR environments for document management is the landscape metaphor, where documents are placed on a plane as color- and size-coded 3D objects. Two systems are presented in this category, with slightly different characteristics.

The **File System Navigator (FSN)** [Strasnick and Tesler 1996] system was created as a 3D file explorer for UNIX systems. The height of the nodes represents the number of contained files (in the case of an ontology, instances). Looking from above, the nodes form a 2-D tree, which represents the hierarchy. Selecting the column with the mouse highlights it, whereas double-clicking opens a detail view for the item on focus.

Table VI. Context + Focus and Distortion Visualization Characteristics. The asterisk (*) Indicates that the Method Has Been Used for Ontology Visualization. "No$^+$" Under Multiple Inheritance Means that the Tool Currently Does Not Support Multiple Inheritance Through Node Replication but Could Be Extended to Accommodate Such Support

Method	Classes and Instances	Class Hierarchy	Multiple Inheritance	Role relations	Properties	Keyword Search	Software availability
HyperTree Visualization (*)	Represented as labels around the central node	Lower level nodes are displayed further away from the center and connected to their parent	No$^+$	No	No	Yes	No
OntoRama (*)	Represented as labels around the central node	Lower level nodes are displayed further away from the center connected to their parent	Child nodes are placed under both parents. Their subtrees are "cloned" as well	Relation types may be selected through checkboxes and displayed on the graph	Properties are displayed in a separate window	Yes	No
StarTree	Represented as labels around the central node	Lower level nodes are displayed further away from the center connected to their parent	No$^+$	No	No	Yes	Available as part of a commercial application in StarTree
MoireGraphs	Represented as nodes around the node on focus	Lower levels are represented with smaller size and further away from the center	No$^+$	No	Properties are displayed on the node as tooltips when it is on focus	No	No
Protégé TGVizTab (*)	Classes and instances are represented as labels of different colors	Lower level nodes are displayed around their parent and connected to it with a "sub" link	There is a link from the node to both its parents	Links with labels on mouse over are used	Properties are displayed in a separate window	Yes, but only works on the part of the ontology that is already visible	Open Source, available as a Protégé [Protégé] plug-in
Ozone (*)	Rectangle nodes with class or instance information	Lower level nodes are linked to their parent	There is a link from the node to both its parents	Labeled links are used	Properties are displayed on the node	Yes. The visualization itself is visual query tool	May be available in the future in Ozone
BiFocal Tree	Represented as labeled rectangles when close to the focus area and as small circular nodes further away.	Child nodes placed under their parent and towards the circumference of the circle	No$^+$	No	No	No	No
3D Hyperbolic Tree	Represented as nodes placed on the surface of a sphere with visible labels on the ones closest to the one on focus.	Lower level nodes are displayed further away from the one on focus and with smaller size	There is a link from the node to both its parents	Nonlabeled links are used	No	No	Available for noncommercial use in 3D Hyperbolic Tree

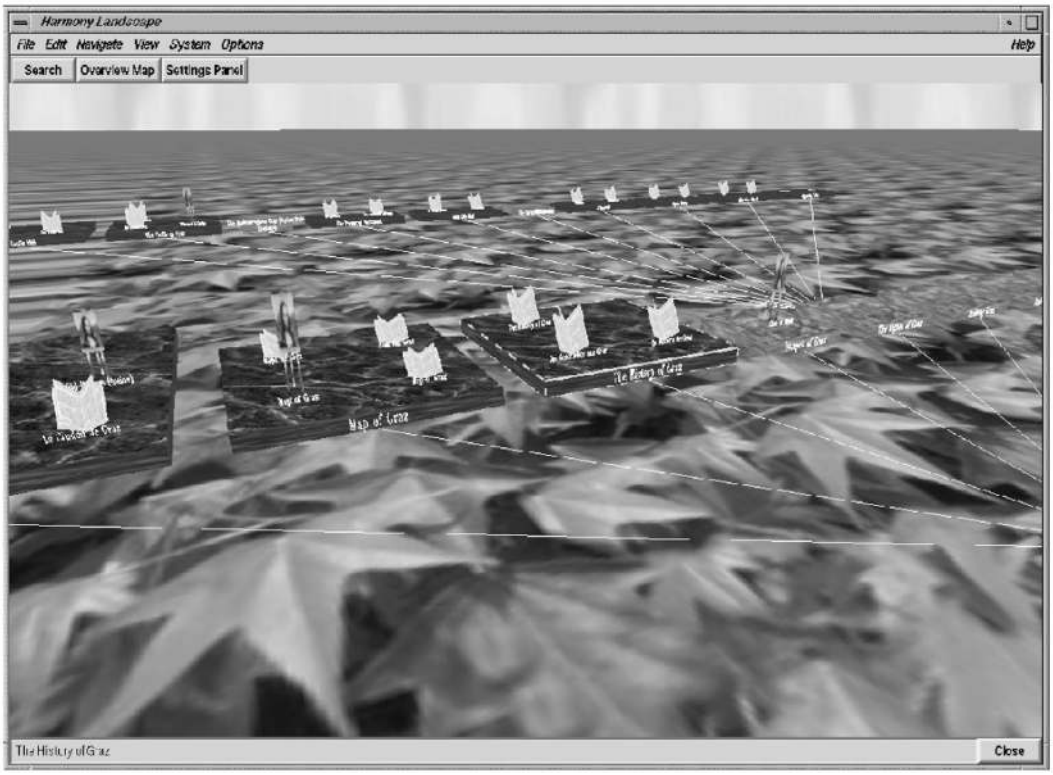

Fig. 13. Harmony Information Landscape.

Harmony Information Landscape [Eyl 1995] was designed for hypertext documents and arranges the nodes, which are represented as 3D objects, directly on the plane (Figure 13). As in FSN, the 3D objects are color- and size-coded to reflect certain document attributes.

However, as the documents are hypertext documents, their hypertext relations are represented as well. They are presented as black lines connecting a selected node to its related nodes. In the case of an ontology, this would be very useful for the visualization of role relations.

Table VII summarizes the characteristics of information landscape visualizations.

11. VISUALIZING TIME IN THE CONTEXT OF ONTOLOGIES

Another issue related to ontology visualization is that of the representation of time in the context of ontologies. Time may affect ontologies in two different ways, the one relevant to the domain the ontology refers to and the other to the process of designing an ontology. Both involve ontologies that are not static but evolving, with their evolution being of interest to ontology users or designers. This section briefly summarizes existing approaches to the issue of ontology evolution.

Katifori et al. [2006b] present the requirements, modeling and implementation as a Protégé plug-in of **OntoTime**, which contains a set of tools for the visualization of historical information presented in an ontology. It proposes a way to display to the user information on classes and instances that reflect entities that have evolved over time and whose evolution is of interest to the user. Such a visualization is particularly useful

Table VII. 3D Information Landscape Visualization Characteristics. None of the Methods Has Been Used for Visualizing Ontologies. "No^{+}" Under Multiple Inheritance Means that the Tool Currently Does Not Support Multiple Inheritance Through Node Replication but Could Be Extended to Accommodate Such Support

Method	Classes and Instances	Class Hierarchy	Multiple Inheritance	Role relations	Properties	Keyword Search	Software availability
File System Navigator	Classes are square pedestals on a plane, and instances columns on the pedestals	There are lines that connect the pedestals to visualize the hierarchical structure	No^{+}	No	Properties are displayed in a separate window	No	No
Harmony Information Landscape	Nodes are labeled rectangles on a plane	There are lines that connect the rectangles to visualize the hierarchical structure	No^{+}	Relations are black lines that connect the nodes above or beneath the landscape	Properties are displayed in a separate window	No	No

in the context of a historical archive ontology, where the organization represented has been transformed in the time span that the archive covers. It attempts to complement existing ontology versioning and class and instance evolution approaches by adding history support, thus allowing the user to explore the ontology in the time dimension as well.

The system **PromptDiff** [Noy et al. 2004] has been developed in the context of a collaborative environment for managing ontologies in order to support ontology versioning, and is available as a Protege plug-in [Protégé Project http://protege.stanford.edu]. Given two versions of an ontology, it allows the user to: (1) examine the changes between versions visually; (2) understand the potential effects of changes on applications; and (3) accept or reject changes. The visualization of differences is based on the Microsoft Word *Compare Documents* paradigm. The two versions are presented one next to the other with highlighting on the parts where changes have occurred. **PromptViz** [Steven and Perrin 2004] is a tool providing advanced visualizations using treemaps to help users understand the location, impact, type, and extent of changes that have occurred between versions of an ontology.

The notion of **Polyarchies** [Robertson et al. 2002] could also be applied in the domain of ontology versioning. Polyarchies are structures composed of multiple intersecting hierarchies and in Robertson et al. [2002] a Web-based visualization technique called Visual Pivot is proposed for the representation of polyarchies. The authors propose this method for exploration of hierarchical data available from different databases, however it would be interesting to see this method applied in ontology version browsing or integration.

12. DISCUSSION–METHOD ADVANTAGES & DISADVANTAGES

This section contains a discussion of the main advantages and disadvantages of the presented methods. For these conclusions, existing evaluations like Kobsa [2004] and Katifori et al. [2006a] were used; we compared and combined their results in order to gain a better insight into the impact of the method characteristics on user performance while executing various ontology- or hierarchy-related tasks. The following subsections summarize the strong points and weaknesses of each general method category with commentary on individual methods when appropriate.

12.1. Indented List

The main advantage of the indented list visualization, the Protégé Class Browser for example, is its simplicity of implementation and representation, and its familiarity to the user, as the same concept is used in numerous file browsers, including Microsoft Windows Explorer. It offers a clear view of the class names and their hierarchy. In the case of node labels, it has a clear advantage in comparison with almost all the other techniques: there is no label overlap and it is not required to move the mouse over an item in order to view the label, as in other techniques like Jambalaya or CropCircles. Retraction and expansion of nodes is a useful feature for focusing on specific parts of the hierarchy, especially for large hierarchies. Furthermore, the simplicity of the interface makes it convenient for quick browsing. This is probably the reason why it has been so effective in information retrieval and it is the main tool used for ontology editing. Tasks like locating a specific class or instance or identifying the children or instances of a class are easier in this case than in most of the other visualizations, as the top-down layout of a tree browser allows for a systematic exploration of the whole ontology. Furthermore, Rivadeneira and Bederson [2003] suggest that it allows direct access to the contents of the classes—in this case the instances.

One problem of this technique is that it in fact represents a tree and not a graph. As a result, it only displays inheritance (isa) relations, not role relations. Furthermore, the multiple inheritance cases are not very obvious. Protégé handles such cases by placing the child node under all its parents; however, it is not always clear to the user inexperienced with ontologies why the same class seems to appear two or more times in subhierarchies of the ontology. As already mentioned, there is no visual representation of the role relations. They are accessible only indirectly, through the class slots. Parsia et al. [2005] also point out that in large ontologies, only a small portion of the ontology may be visible at once, as the indented list, top-down layout results in rather poor space filling and needs scrolling during browsing. Furthermore, the nodes at the same level are not immediately recognized as siblings if their subhierarchies are expanded. This problem has been identified in Katifori et al. [2006a] as well as in Plaisant et al. [2002]. Additionally, this type of visualization is not very helpful for tasks related to the general ontology structure, like identifying the depth of the hierarchy or finding nodes with many children or deep hierarchies. In the Katifori et al. [2006a] evaluation, many users suggested or seemed to miss the existence of "Expand All" and "Retract All" buttons in the Protégé Class Browser.

However, it has been proven in several evaluations, Rivadeneira and Bederson [2003], Kobsa [2004], and Cockburn and McKenzie [2000] for example, that this type of visualization seems to perform better than the other visualizations used for hierarchies. In Katifori et al. [2006a] as well, it had the best performance. This is the reason why it is used as a baseline system in many evaluations. It is still an open issue whether familiarity with file browsers is the main reason for the success of this method. A very possible reason is the fact that it seems natural to the user, being accustomed to it in his/her everyday tasks, like scanning the contents at the beginning of a book or writing down a list of tasks she/he has to perform. It could be difficult to envision any ontology visualization environment without it. Its use in conjunction with other visualizations that compensate for its drawbacks may lead to a very powerful visualization tool.

12.2. Node–Link and Tree

Tree–like node link diagrams are another common and intuitive way to represent hierarchy. As nodes are displayed in a top down (or left to right) positioning, a good overview of hierarchical structures is offered, as different levels and features such as hierarchy depth or width are easily distinguishable. According to Carriere and Kazman [1995], their cone tree implementation, fsviz, is most suited to helping users answer structural and trends-related questions.

According to Plaisant et al. [2002], on the other hand, tree node-link methods typically make inefficient use of screen space, leaving the root side of the tree completely empty, usually the top or left of the screen, and overcrowding the opposite side. Even trees of a hundred nodes often need multiple screens to be completely displayed, or require scrolling since only part of the diagram is visible at a given time. Van Ham and Van Wijk [2002] and Bruls et al. [2000] support this and state that traditional node link diagrams lead to cluttered displays when used to visualize more than a few hundred nodes.

The Protégé **OntoViz** visualization received very negative reactions in the Katifori et al. [2006a] evaluation. It attempts to alleviate the problem of node clutter by allowing the user to select the nodes she/he would like to display, along with their subhierarchies or related nodes, through a configuration panel. However, several interaction issues seemed to lead to a rather bad performance. All users commented on the lack of interaction and had experienced problems with the navigation, such as having to drag the scrollbars to navigate. Furthermore, the zoom in and out commands and clicking

accidentally on an instance, which resulted in focusing on its class, resulted in the loss of the item on focus. They found the presentation "poor" and "chaotic" and commented on the lack of a search tool and the fact that some labels are not fully visible, forcing the user to guess their meaning; absence of sorting (instances are not presented in alphabetical or any other deterministic order) was also negatively commented. However, some users commented that the visualization could be effective for smaller ontologies or if the user is very familiar with the ontology, as it seemed to them useful for the presentation of hierarchies.

SpaceTree tackles the problem of clutter by introducing expansion and retraction of subhierarchies. SpaceTree performed really well [Plaisant et al 2002] in tasks related to returning to a previously visited node and to hierarchy overview, because it maintains a constant positioning of the nodes in combination with the clear view of the hierarchical structure inherent in this type of visualization. Its performance for locating a node was significantly better in comparison with CropCircles and Treemap in the Wang and Parsia [2006] evaluation. The node that controlled expansion of subtrees—expanding children up to a certain level—seems to be effective.

TreePlus in Lee et al. [2006a] was found to have significantly better performance than a **TGViz**-like graph visualization in several of the evaluation tasks. In a task that included finding a specific node with a maximum number of connections to another type of node, users preferred an orderly browsing using TreePlus rather than attempting to locate the node with the most connections in a cluttered and chaotic TGViz–like visualization. As a result, one possible answer to the visualization of large ontology structures is the support for localized browsing in combination with an effective overview.

The use of 3D in this type of visualization is another proposed solution to the problem of screen clutter. The designers of the **Cone Tree** method [Robertson et al. 1991] point out its advantages concerning the better use of available screen space. However, even though transparency is used, according to Wiss et al. [1998] a data set with many levels and many subhierarchies will result in occluded subtrees. The **Cone Tree** seems to produce a clutter for "bottom heavy" data sets, hierarchies with many wide subhierarchies, a problem is evident even with relatively small data sets of a few hundred nodes. And, according to Plaisant et al. [2002], 3D node link diagrams seem to increase the complexity of the interaction as well.

The Carriere and Kazman. [1995] evaluation of **fsviz** seems to support these conclusions. **Cone Trees** are effective for offering an overview of the structure but not so effective for tasks related to locating specific nodes. This visualization has an inherent problem with label representation, as occlusion is inevitable for nodes that are at the back side of the cone. Using rotation of the cone base in order to browse sibling nodes had mixed reactions from the users: some found it preferable to scrolling while others found disorienting the fact that nodes were changing position. However, lack of familiarity with the interface was noted as another probable reason for bad performance. Users found the 3D interface attractive, which means that there is room for improvement, and further evaluations are needed to better identify strong points and weaknesses of 3D trees.

TreeViewer, the more realistic, real tree like visualization, trailed most systems in performance in the Kobsa [2004] evaluation, particularly for property-related tasks. A reason for this is that it lacks basic functionalities such as search. Furthermore, the different sizes of branches, the turns that branches take, the fact that same-level branches split off at different heights, and finally the occlusion of branches, all make it difficult to tell when two branches are of equal levels.

To sum up, tree-like node link diagrams seem to be effective for representing an overview of the hierarchy. However, this is the case only for small trees because they tend to fall short when more than a couple of hundred elements have to be visualized

simultaneously. Efforts to alleviate this problem include node filtering, retraction and expansion, and the use of 3D, all to the detriment of quick node locating and overview representation.

12.3. Zoomable

Zoomable interfaces (ZUIs) seem to be effective for locating specific nodes, as they provide a comprehensive view of the hierarchy level the user is zoomed in. There were some problems however which were encountered during evaluations.

In Katifori et al. [2006a], **Jambalaya** in general got positive reactions. Most users commented positively on the effective search tool and the animated transition when double clicking on an instance or class. They liked "flying together with the visualization to locate the information." Some noted that they would like the animation to be faster ("I lose time waiting") or slower ("not enough time to understand the transition") or to display the steps of the transition to the side. It was interesting that none of the users tried to use the visible relation links and almost all noted as a negative point the appearance of the links and the fact that after browsing some classes there come to be so many relation links that they obstruct the view to the visualization. They also noted that labels overlap in the case of many instances. In **Grokker**, problems with labels were noted as well [Rivadeneira and Bederson 2003]. As in **Jambalaya**, users had a problem knowing which is the current parent node that had been zoomed in, or if the node had already been visited.

For the **Information Cube**, according to the Wiss et al. [1998] evaluation, there is excess space inside each cube if there are fewer than $\lceil \sqrt[3]{n} \rceil s$ children or if the children are of varying sizes. The resulting size of the surrounding cube will then not represent the contents very well. Another problem is that if the difference between the biggest and the smallest subhierarchies is large, the smallest child cube will be so small that it is difficult to see. Furthermore, the visualization shows misrepresented sizes as soon as the contained cubes are of varying sizes. This is often the case when a parent node contains both leaves and subhierarchies. The ideal data set for the **Information Cube** would be a hierarchy where all leaves are at the same level. Lastly, it is not possible to retain global context while zooming in with an **Information Cube**.

GopherVR is a simple and clear visualization [Wolte 1998]. The nodes are presented with labels only if they are close to the viewpoint. Its main disadvantage is that since it presents only one level at a time it does not provide an overview of the hierarchy. Furthermore, the nonconventional navigation methods used are not very intuitive and, as a result, not very useful for reducing the user cognitive load.

ZUIs in general seem to be successful for browsing to locate specific nodes. However they do not offer an effective overview of the hierarchical structure and they do not support the user in forming a mental image of the hierarchy. Rivadeneira and Bederson [2003] suggest that ZUIs could be improved with navigational cues that could inform users which elements have already been visited and hierarchical cues that could tell users which level they are in and how deep the structure is.

12.4. Space filling

According to Plaisant et al. [2002] and Van Ham and Van Wijk [2002] space filing techniques have been successful at visualizing trees that have property values at the leaf (instance) node level, which is the case in ontological structures. The reason for this is that these techniques allow color and size coding of properties at instance level. They are effective when the user cares mostly about leaf nodes and their properties but does not need to focus on the topology of the tree, or the topology is trivial, at

most 2 or 3 levels. This is also confirmed in the results of the Kobsa [2004] evaluation. Wang and Parsia [2006] confirm good performance of CropCircles for tasks related to distribution of nodes at the leaf level, like identifying a node with a large number of children.

Van Ham and Van Wijk [2002] note that standard **Treemaps** have two problems. First, they often lead to high aspect ratio rectangles, and second no space remains for the internal nodes of the tree. This makes it difficult to reconstruct the hierarchical information from the **Treemap**, especially when dealing with large, deep hierarchies. Although **SequoiaView** attempts to remedy this problem, it still requires significant cognitive effort to extract the actual tree structure from the visualization. **SequoiaView** users in Kobsa [2004] had worse performance than **TreeMap** users in structure related questions, specifically regarding level and sibling detection. Its shaded $2\frac{1}{2}$ D "cushions" seemed to hinder the evaluation subjects in the evaluation of a tree structure that contained many leaf nodes of similar size. Node boundaries were not easy to distinguish in many cases. **Treemap** techniques also require training because of their unfamiliar layout [Babaria 2004].

Kobsa [2004] also suggests that the usefulness of the **TreeMap** may be enhanced by integrating more string search functionality and a function that highlights search results, as well as a detail-on-demand functionality. Furthermore, no significant differences were found between Treemap and Windows Explorer and it is doubted whether increased practice would enable Treemap users to outperform Windows Explorer users.

Wang and Parsia [2006] state that CropCircles was found significantly better than TreeMap for returning to previously visited nodes. This result suggests that the CropCircles visualization is probably better suited then TreeMap for aiding spatial memory.

BeamTrees achieved the worst quantitative results in Kobsa [2004]. Although in structure related tasks it seems to perform relatively well, global structural tasks were a problem because nodes of the same level did not appear to be on the same level in the 3D visualization. The subjects seemed to miss "Undo" and system reset. Furthermore as Van Ham and Van Wijk [2002] state, many non-leaf nodes have touching edges, making it more difficult to perceive them as separate visual entities.

Andrews and Heidegger [1998] state that the **Information Slices** technique appears to be particularly well-suited to the rapid navigation of deep hierarchies. It is very easy to rapidly traverse many levels of a hierarchy and gain an overview of the relative sizes of parts of a tree. Broad hierarchies can result in dense, thin slices, which are sometimes initially overwhelming. This is somewhat alleviated by allowing the user to select particular (dense) slices of interest and fan them out in 180 degrees of their own in the right-hand disc.

As already stated, space-filling techniques seem to be particularly suited for tasks that include overview of certain properties of the ontology instances or an overview of areas with many or few nodes. However they are not as effective for structure related tasks.

12.5. Focus + Context and Distortion

Focus + context techniques have several advantages. Every node of interest can be easily moved towards the center of the tree in order to be displayed with more details, at the same time retaining the context of nodes related to the one on focus. On the other hand they do not maintain a constant positioning of the nodes, which may be somewhat disorienting.

In the evaluation of **HyperTree** Souza et al. [2003], experienced users stated that the HyperTree visualization is far more effective than specially formatted Excel documents, but expressed reservations that novice users might be discouraged.

StarTree attempts to make better use of screen space as it breaks lose from the traditional tree orientation using circular layouts. It uses animation to readjust the focus point of the visualization. According to Plaisant et al. [2002], the animation is striking but the constant redrawing of the tree may be distracting. Labels are hard to browse because they are not aligned and sometimes overlap. In addition, the unconventional layout may not match the expectation of the users. **StarTree** performance in Kobsa [2004] was found to be "average" on every task. The user has to rotate the tree a lot to scan lower level nodes. Furthermore, nodes with the same distance from the centre are not necessarily on the same tree level. This is also the case with **TGVizTab**. This may hinder tasks related to the ontology hierarchy, like identifying sibling nodes.

The **3D Hyperbolic Browser**, according to Munzner [1997], may easily handle more than 20,000 nodes and is very effective for a representation of a large graph on a small screen space, as it uses distortion to provide focus and context. Important structures and relations between them are claimed to be easily distinguishable. On the other hand, the weaknesses of the system are that the initial view provides only part of the sphere, that the labels are not visible away from the center and that sometimes the animation may be disorienting.

Another advantage of the **3D Hyperbolic Browser** is the ability to present nontree links in context, in order to view relationships between a part and the far-flung reaches of the whole. Although the details of the nontree link destinations are usually distorted, a rough sense of their direction helps the user construct and maintain a mental model of the overall graph structure. The details become clear in a smooth transition when that area of the structure is brought towards the center. In the 3D system the nontree links can follow paths that are unlikely to intersect the surrounding spanning tree links.

TGVizTab received intense but contrasting reactions in the Katifori et al. [2006a] evaluation. Some users disliked it and for some it was the best. The main reason users gave for this was the "spontaneous" movements of the ontology. Some users found it "playful," "nice," or "funny," while others were not very content having to "chase the concept which is moving by itself" or found the effect "dizzying." Some users commented that the visualization gave them a clear view of the hierarchy while others found it "chaotic." It is interesting however, that even the users who disliked TGVizTab performed well in it, as it helped them to locate nodes very quickly. On the other hand, almost all commented on the lack of an effective search tool accompanying the visualization and the fact that in some cases, labels occlude the ones behind them.

In the case of the **BiFocal Tree**, Ricardo et al. [2002] mention that the drawback of the technique is the lack of stability of the context area layout when a change of focus node occurs. Depending on the new focus node, the diagram can be drastically different from the previous one.

On the whole, focus + context techniques seem to be very effective at providing global overviews and displaying many nodes at once. They can be used for focusing on certain nodes and viewing their related nodes, and for quick browsing of the ontology to locate specific classes or instances. However they do not offer a very obvious representation of the hierarchy structure as the user has to see the link label in order to distinguish parent from child nodes. And if role relations are also visible, the display seems to clutter even for an ontology of a few hundred nodes. Label clutter seems to be a problem and the constant redrawing of the graph does not help the creation of a mental model of the ontology.

12.6. 3D Information Landscapes

3D information landscapes attempt to present hierarchies using a landscape metaphor. 3D in this case would be useful providing an extra dimension where node properties could be coded and relation links presented.

In the evaluation of Wiss et al. [1998] it is pointed out that the **Harmony Information Landscape** produces some excess space in the x direction when subhierarchies are of varying size, which in turn makes the landscape wide. With such a landscape it is difficult to see the entire subtree without zooming in or out. The Information Landscape has problems with data sets where a node has many children. This creates a wide layout that cannot be seen all at once, and as a result it is not possible to retain global context while zooming-in with an Information Landscape. On the other hand, according to Wolte [1998], large hierarchies are clearly laid out in the Harmony landscape. The visualisation of the hyperlinks is not very effective, due to clutter. Text labels also tend to overlap or occlude other objects.

According to Wolte [1998], on **fsn** the mapping of properties like size and type to visual representations simplifies navigation, since each node gets its specific look, which is easy for the user to recognize. For instance, large nodes can act as landmarks, so the user easily knows which part of the hierarchy she/he is focused on. Due to the 3D perspective, the user's view is focused on the selected node and its subnodes. All other, probably less interesting, nodes are smaller objects towards the horizon or are invisible. So the user is not distracted by uninteresting objects. To focus on a directory is easy, but for a good structural overview, a separate overview window is needed.

To sum up, it is not yet very clear if information landscapes could be useful in the context of ontologies. They have not yet been used much in practice and there is a lack of extensive evaluations as well. Navigation in these environments also plays a very important part. Information Landscapes could probably be effective for hierarchy overview related tasks, if coupled with appropriate search and filtering tools and intuitive, simple, and effective navigation mechanisms.

13. TASK SUPPORT

Based on ontology visualization characteristics, this section attempts an analysis of tasks related to ontologies, with the aim of assessing which visualizations best support each task type. The categorization of tasks is based on the task analysis proposed by Shneiderman [1996], who presents seven high-level tasks that an information visualization application should support. These are the following:

1. *Overview*. Gain an overview of the entire collection.
2. *Zoom*. Zoom in on items of interest. When zooming, it is important that global context can be retained.
3. *Filter*. Filter out uninteresting items.
4. *Details-on-demand*. Select an item or group and get details when needed.
5. *Relate*. View relationships among items.
6. *History*. Keep a history of actions to support undo, replay, and progressive refinement.
7. *Extract*. Allow extraction of subcollections and query parameters. This extraction refers to saving desired subparts of the collection and is typically supported by the ontology management tools, not the visualization methods per se. Since the current work is focused on visualization methods, rather than individual tools, this task category will not be examined.

The first six high-level tasks are refined into lower-level tasks based on Lee et al. [2006b], Katifori et al. [2006a] and Wiss et al. [1998]. The main visualization categories presented in the previous sections have different levels of support for the identified ontology tasks. The task support table (Table VIII) that follows is derived by evaluation results presented in the "Discussion" Section 12, but it needs further study and evaluations in order to validate it. Furthermore, it should be noted here that some

Table VIII. Ontology-Related Tasks versus Visualization Methods. ++ Shows That the Method Supports this Task Very Effectively, + That it Supports It but Not Very Effectively,— That It Does Not and—That it Does Not Support it and in Some Cases it May Hinder it. Comments Are Provided When Appropriate

		Indented List	Node-Link and Tree	Zoomable	Space-Filling	Focus + Context and Distortion	3D Information Landscapes	Time-Related Visualizations
Overview	**Hierarchy Overview (is a hierarchy along with multiple inheritance)**	+	++	−−	−	−	++	
	View depth of the hierarchy	+ The user has to expand all nodes to find out.	++	−−	−	−	++	
	Identify areas with many/few classes/instances	−	+	−−	++	+	++	
	Overview of instances related to some property	−−	+	−	++	+	++	
	Overview of role relations	−−	+	+(Jambalaya) −− (Others)	−−	−	+(Harmony) −− (fsn)	
	Number of instances per class	This task is partly tool-dependent. A number should be available, otherwise the user has to count.						
	View total number of classes	This task is partly tool-dependent. A number should be available, otherwise the user has to count.						
	View total number of instances	This task is partly tool-dependent. A number should be available, otherwise the user has to count.						
	Quick Browsing	++	++	+	++	++	+	
Zoom	**Find Class or Instance by name or other property.**	This task is partly tool-dependent. A search tool is necessary in this case, especially for big ontologies. If search is not available, then methods that better support this task are those supporting quick browsing, in order to examine all ontology nodes quickly.						
	View sub-hierarchy (retaining context)	+	++	−	+	+	++	
	View path to instance or class	++	++	−−	−	+	++	

(Continues)

Table VIII. Ontology-Related Tasks versus Visualization Methods. ++ Shows That the Method Supports this Task Very Effectively, + That it Supports it but Not Very Effectively,—That it Does Not and—That it Does Not Support it and in Some Cases it May Hinder it. Comments are Provided When Appropriate (continued)

	View class/instance properties	This task is partly tool-dependent; see the individual method "Properties" features in Tables II to VII									
	View class siblings	+	++		+		+	+	++		
Details demand	**View number of class subclasses**	This task is partly tool-dependent. A number should be available, otherwise the user has to count.									
	View number of instances per class	This task is partly tool-dependent. A number should be available, otherwise the user has to count.									
	View number of related classes to a class	This task is partly tool-dependent. A number should be available, otherwise the user has to count.									
Filter	**Hide nodes**	+	+		−		+(TreeMap) − Others	++		−−	
	Hide sub-hierarchies	++	++		+		−−	+		−	
Relate	**View parent classes**	−	++		−−		−	+		++	
	View sub-classes	++	++		++		−	+		++	
	View role relations	−−	+ ++(TreePlus)	+(Jambalaya)	−− Others		−−	−		+	(Harmony) −− (fsn)
	Compare classes or instances	+	−		−		+	+		−	
	View class/instance timeline	−−	−−		−−		−−	−−		−−	OntoTime
History	**Ontology Evolution**	−−	−−		−−		−−	−−		−−	PromptViz, PromptDiff
	Undo/Redo – Back/Forward	This task is tool-dependent. The tool containing the visualization should provide this functionality.									
	Return to Initial View	This task is tool-dependent. The tool containing the visualization should provide this functionality.									
	Return to previously visited class or instance	++	+ ++(TreePlus)		−		++	−		++	

methods have features of more than one of the defined categories, resulting in a task support level that may differ from that corresponding to their category. These cases are addressed in the discussion section and also noted in Table VIII.

As seen from the table, not all tasks can be effectively supported through a single visualization. This fact supports the view that more than one visualization method should be made available to ontology designers and users. Furthermore, not all tasks may be supported by visualization, thus supplemental information retrieval aids should be provided. Locating a specific node, for example, may be accomplished by browsing the ontology, using the visualization, but it is much quicker and more effortless to do so using a search tool. This fact was proven in Katifori et al. [2006a]. Cardinality-related tasks, for example, finding the number of class siblings or children, can be performed using the visualization alone, but the user would have to count the nodes; certain tools facilitate these tasks by providing the numbers (by default or on request), but these facilities are strongly tool-dependent, rather than visualization method-dependent.

"Going back to a previously visited node" could be supported by the tool if it provided an elaborate history mechanism, but also by the visualization. If the visualization supports learning of the ontology structure and the creation of a mental image, then the user may easily return to previously visited nodes. Methods that are more effective to this end are the ones that maintain a constant positioning of the nodes and allow quick browsing at the same time. Last, tasks like "Forwards-Back" or "Initial View" are solely tool-related.

14. 2D VS 3D

The issue of 3D visualizations is a rather controversial one. Human vision is based on 3D projections of the real world and one could easily assume that visualizations that are closer to this 3D projection would also be more effective. Things are not that simple, however, and 3D has not yet dominated our computer desktops. Especially in the case of abstract data representation, where more factors than the faithful representation of the real world should be taken into account, things are even more complicated.

Certainly 3D offers one extra dimension in order to use the available screen space more effectively, as Robertson et al. [1991] suggest. Furthermore, according to Bosca et al. [2005], mapping the many features of an ontology, like the class hierarchy, the role relations, the properties, and the instances, on two dimensions can be somewhat restrictive, while 3D offers the possibility of a richer representation. 3D visualizations also seem to have a strong user preference on their side [Smallman et al. 2001].

However, it has not yet been made clear if 3D visualizations should be preferred to 2D ones. As Smallman et al. [2001] state, there is a growing literature on the advantages and disadvantages of 3D visualizations versus 2D with somewhat conflicting results. In their evaluation of a 3D versus 2D display and also in the Hicks et al. [2003], the 2D seemed to have better performance. According to Plaisant et al. [2002], 3D representations only marginally improve the screen space problem while increasing the complexity of the interaction. Cockburn and McKenzie [2002] have shown that navigation in a 3D space can be difficult for a novice user, while even simple tasks such as selecting an object can be problematic.

Apart from OntoSphere, 3D visualization has not yet been applied extensively to the ontology domain and as a result there are not yet conclusive results as to its effectiveness. Evaluations of 3D visualizations of hierarchies like Wiss et al. [1998] have provided useful results as to strong points and weaknesses of such visualizations and the ongoing research on this field will most certainly produce interesting results as to

the use and effectivenes of 3D in the field of ontology visualization. As Kobsa [2004] suggests, the negative results of 3D visualizations are in some cases the result of the lack of other features such as an effective search tool, highlighting of search results, filtering, or navigation.

15. NAVIGATION AND INTERACTION ISSUES

All static hierarchical presentations have limits as to the quantity of information they are capable of presenting on a finite display space Babaria [2004]. When these limits are reached, navigational techniques must be used, creating the potential for loss of context. In most visualizations, depending upon the drawing algorithm and the size of the display space, a hundred or so nodes can be adequately represented on screen without the need for panning or zooming.

The various visualization techniques presented here differ in the level of interaction they offer to the user. Some of the methods allow the user to only view the presented ontology as a static image. Others allow the retraction and expansion of nodes, the movement and rotation of the presented ontology, zooming or clicking to change hierarchy level or the node on focus. Other, mostly tool-related, features are history functionalities, overview windows, and the use of animated transitions. All these features are useful for exploring the ontology to find specific nodes, focus on nodes of interest, or examine relations between nodes. The following table summarizes which of the previously mentioned features is provided by each of the visualization methods.

Retraction and expansion of nodes, viewpoint movement, and rotation, and zooming, are features that most of the visualizations support, since they are necessary to navigate hierarchies with more than a hundred nodes. In these cases, the interaction techniques used are essential for the success of the visualization as they greatly affect task completion. This is particularly evident, for example, in the case of OntoViz [Katifori et al. 2006a], the bad performance of which is a direct consequence of ineffective interaction. Expansion and retraction for example is accomplished by using a configuration panel where the user selects nodes she/he would like to expand.

Zooming is another important issue. According to Plaisant et al. [2002], semantic zooming is preferred over geometrical scaling; it is important to provide the user the means to focus on specific nodes and be able to view their details, not just scale the visualization as an image. Another issue with zooming is the loss of the sense of where the user is and where she/he came from. As already mentioned, navigational cues such as informing the user of the current level of the hierarchy and the path she/he followed to get there are essential to this end.

Another useful feature is Overview tools and Back and Forward navigation aids. Overview tools are especially effective in zoomable visualizations where the user may easily lose sense of his/her position. “Back” and “Forward,” on the other hand, allow the user to retrace his/her steps during browsing.

Movement and rotation of the graph is another interaction feature that should be carefully designed. Although it allows the user to manipulate and examine the ontology in order to locate specific nodes or areas of interest, it may disorient the user. Furthermore it does not help the creation of a cognitive model of the ontology as nodes continuously change position.

This is also the case of animated transitions. They are used as a means to change the view while zooming, rotating the graph, expanding or retracting, focusing on another part of the ontology and so on, while helping the user to understand the change and retain a clear picture of his/her previous and current locations in the graph. However, the reaction of the users to it is not always positive and it may be conflicting. In the case of its use for moving automatically from one place to the other, the user may find

Table IX. Visualization Methods Categorized According to the Interaction and Navigation Techniques They Employ

Retraction and Expansion of nodes/Pruning	Movement and/or rotation of the graph	Movement and/or rotation of the viewpoint	Zooming	Overview Window	History/Back and Forward	Animated Transitions
Class Browser, OntoViz, SpaceTree, OntoTrack, GOBar, GOSurfer, Cone Tree, fsviz, Reconfigurable Disk Tree, Information Slices, TGVizTab, BiFocal Tree, TreePlus	Tree Viewer, BiFocal Tree, OntoSphere, BeamTrees, TGVizTab, 3D Hyperbolic Browser	Class Browser, OntoViz, IsaViz, SpaceTree, OntoTrack, Cone Tree, fsviz, Reconfigurable Disk Tree, OntoSphere, Jambalaya, Information Pyramids, TGVizTab, Gopher VR, Harmony Information Landscape, fsn	OntoViz, IsaViz, SpaceTree, OntoTrack, Cone Tree, fsviz, Reconfigurable Disk Tree, Tree Viewer, OntoSphere, Grokker, Jambalaya, Information Cube, Information Pyramids, CropCircles, TreeMap, SequioaView, BeamTrees, TGVizTab, Gopher VR, Harmony Information Landscape, fsn	OntoViz, IsaViz, OntoTrack, Information Pyramids, Jambalaya, TreePlus	Jambalaya, Information Pyramids, CropCircles	SpaceTree, OntoTrack, Cone Tree, fsviz, Reconfigurable Disk Tree, Jambalaya, TGVizTab, OZONE, BiFocal Tree, 3D Hyperbolic Browser, TreePlus

the animation useful because it shows the transition path, or annoying because it is time consuming.

On the whole, interaction and navigation techniques are essential for the success of a visualization method. They form an integral part of the method, as without them the visualization would be a static image. More research and evaluations are needed in order to couple visualization and interaction effectively to create a useful and easy to use tool.

16. SCALABILITY ISSUES

Little is known in terms of the scalability issue in visualizing large hierarchies [Fekete and Plaisant 2002]. Current systems tend to avoid the problem of scalability by limiting the number of visible items to about 10000. **Ontosphere** for example reports problems with many nodes (more than 1000) such as occlusion and label overlap. According to Fekete and Plaisant [2002], control panels, labels, margins, waste space, and data structures are not optimized for speed, and the graphics libraries they employ are not sufficient.

Another issue in big ontologies is that of the node labels display, especially important in an ontology, which is basically composed of concepts that the user should be able to read to understand. Fekete and Plaisant [2002] state that text labels are not preattentive but nevertheless important to understand the context in which visualized data appear. Labeling each item cannot be done statically on a dense visualization.

The visualization of relation links is also problematic and the display may become cluttered very quickly. Katifori et al. [2006a] report that both **TGVizTab** and **OntoViz** became impossible to use when relation links were visible, even for an ontology for less than 300 nodes. In **Jambalaya** too, users did not exploit the relation links—they even seemed to hinder them. A solution to the problem of relation link clutter is not to display them all on the graph but rather allow the user to select which ones to display. Several visualizations like the **3D Hyperbolic Browser**, **Jambalaya**, **OntoViz** and **TGVizTab**, support this.

OntoViz also becomes cluttered very quickly when the number of nodes increases, as shown in the Katifori et al. [2006a] evaluation, which used an ontology of approximately 250 nodes. For node-link diagrams, Bruls et al. [2000] set 200 nodes as the limit for successful visualization. According to Carriere and Kazman [1995], **Cone Tree** techniques tend to lose their efficacy once the hierarchy to be visualized exceeds approximately 1000 nodes. At the time of the publication of their work, their implementation of the cone tree, **fsviz**, seemed to suffer from extremely poor interactive performance for trees of about 2000 nodes. However, larger hierarchies of 5000 nodes are said to have been rendered successfully: without having any node obscure any other node. **SpaceTree**, which incorporates expansion and retraction of nodes, was evaluated successfully on a tree of more than 7000 nodes along with Hypertree and Explorer [Plaisant et al. 2002].

Techniques based on zooming, which use different node sizes for the representation of the lower levels, also become illegible as the number of nodes increases. The zoomable techniques that do not visualize all the levels at the same time may become difficult to navigate after a point. The reason is that when the number of nodes and hierarchy levels increases, it becomes more and more difficult for the user to keep track of his/her position.

The more efficient techniques for large ontology sizes are most probably the techniques that use distortion or expansion and retraction of the nodes, because they can provide detail, maintaining at the same time the general impression of the context. The **3D Hyperbolic Browser** has been reported by its creators [Munzner 1997] to perform well for thousands of nodes. These are distinguished as main or labeled ones;

Table X. Categorization of the Methods According to the Maximum Number of Nodes They Have Been Reported to Effectively Support

Up to 1000	Between 1000 and 10000	More than 10000
IsaViz, OntoViz, GoSurfer, GoBar, Cone Tree, Grokker, Jambalaya, Information Cube, Information Pyramids, CropCircles, TreePlus	Class Browser, SpaceTree, fsviz, OntoTrack, BeamTrees, HyperTree, Tree Viewer, , BiFocal Tree, OntoSphere,Information Slices, OntoRama, TGVizTab, Ozone, fsn, GopherVR, Harmony Information Landscape	TreeMap, Sequoia View, 3D Hyperbolic Tree

peripheral, which are small but distinguishable, and fringe, which are not individually distinguishable but are useful to display the structure. The **3D Hyperbolic Browser** can show up to 50 main nodes, 500 hundred peripheral ones, and thousands of fringe ones.

In the user survey in Ernst and Storey [2003], five ontology size categories are identified:

1. Fewer than 100 nodes,
2. Between 101 and 1,000 nodes,
3. Between 1,001 and 10,000 nodes,
4. Between 10,001 and 100,000 nodes,
5. More than 100,001 nodes.

The number of nodes in this case includes both classes and instances.

Most users are anticipated to be working with the second category of ontologies, whereas none is anticipated to be working with the last. In our case, we will use the three categories in Table X as a criterion for the classification of the ontology visualization methods (the two first categories of Ernst and Storey [2003] are merged into a single one, and so are the last two). In Table X each category lists the method that could be effectively used, up to the number of mentioned nodes. The classification is based on the existing literature as presented in this section. When there was no information regarding which category the method belongs to, an estimation was made comparing it with others of its category.

As seen from Table X, only three methods claim to provide support for more than 10,000 nodes. This fact shows that the issue of scalability in the visualization domain is still an important one.

Van Ham and Van Wijk [2002] propose three solutions to the problem of visualization of many nodes:

1. Increase available display space, by either using three dimensional and/or hyperbolic spaces.
2. Reduce the number of information elements by clustering or hiding nodes.
3. Use the given visualization space more efficiently by using every available pixel.

Such solutions have been employed by most of the presented visualizations with varying degrees of effectiveness.

On the whole, as Munzner [1997] also states that information density should not be the only metric in ontology visualization: when taken too far, it becomes a clutter. Drawing for example all the links in a highly connected graph yields a picture that can give a high level overview of the global structure but is useless for examining the details. There is always a trade-off between maximum number of nodes displayed

and clarity and details in the visualization. Allowing the user to configure the visualization according to his/her needs and the related task is probably the best solution possible.

17. REASONING

A very important issue related to ontologies, which are mainly knowledge representations, is that of reasoning. An ontology is more than a simple graph, it is a structure with rich semantics and the ability to use logic operations on it so as to reach conclusions and produce new information. The issue of coupling visualization and reasoning has not yet been sufficiently treated in existing literature and very few methods support it. OntoTrack, for example, has a connection with an external Reasoner in order to detect problems while editing, which are outlined with red on the visualization. OZONE on the other hand, as a visual query tool allows the user to extract information from the ontology. However, this issue should be further investigated in order to create visualizations that will support all the ontology features more effectively.

18. CONCLUSIONS—FUTURE WORK

Much work has been done in the field of graph and hierarchy visualization both in 2D and 3D. The visualization of ontologies is a particular subproblem of this area with many implications due to the various features that an ontology visualization should present. The current work is an attempt to summarize the research that has been done so far in this area, providing an overview of the existing methods and their main advantages and disadvantages. As the results imply, there is not one specific method that seems to be the most appropriate for all applications and, consequently, a viable solution would be to provide the user with several visualizations, so as to be able to choose the one that is the most appropriate for his/her current needs. This is a feature proposed by Wiss et al. [1998] and Golemati et al. [2006]. Some ontology management tools already provide combinations of visualization methods. Protégé [Protégé Project http://protege.stanford.edu] for example includes several visualization plugins that are coupled with the Protégé indented list Class Browser.

Furthermore, an important conclusion of most of the evaluations taken into account for this work is that visualizations should be coupled with effective search tools or querying mechanisms. Browsing is not enough for tasks related to locating a specific class or instance, especially for big ontologies. Most users also seem to dislike chaotic and too cluttered overviews, and tend to prefer visualizations that offer the possibility of an orderly and clear browsing of the presented information, even if in some cases it requires focusing on a specific part of the ontology or hierarchy. This fact implies that visualizations should also take advantage of the semantic context of the information and even the user profile, in order to guide and support the hierarchy or ontology exploration.

In some applications it is preferable or more convenient to provide only a single visualization of the ontology. In this case the designer has to make a choice among the available methods, based on certain characteristics of the ontology, the application, the user profile, expertise, and so forth. It is hoped that the current work will be useful in order to make that choice.

This work along with the Katifori et al. [2006a] evaluation is the first step for a more detailed evaluation of the presented methods that will involve experiments with several user groups. That way we hope we will be able to provide more conclusive results as to the effectiveness of each method, and proposals as to how to improve them.

REFERENCES

3D HYPERBOLIC TREE. http://graphics.stanford.edu/~munzner/h3/

ALANI, H. 2003. TGVizTab: An ontology visualization extension for Protégé. In *Proceedings of Knowledge Capture (K-Cap'03), Workshop on Visualization Information in Knowledge Engineering*, Sanibel Island, Florida.

AMANN, B. AND FUNDULAKI, I. 1999. Integrating ontologies and thesauri to build RDF schemas. In *Proceedings of the Third European Conference on Research and Advanced Technology for Digital Libraries*. 234–253.

ANDREWS, K., AND HEIDEGGER, H. 1998. Information slices: Visualizing and exploring large hierarchies using cascading, semicircular discs. In *Proceedings of the IEEE Information Visualization Symposium*, Carolina. 9–12.

ANDREWS, K. 2002. Visual exploration of large hierarchies with information pyramids. In *Proceedings of the Sixth International Conference on Information Visualization (IV'02)*. IEEE Computer Society Press, London, England, 793–798.

ANDREWS, K., WOLTE, J., AND PICHLER, M. 1997. Information Pyramids^{TM}: A new approach to visualizing large hierarchies. In *Proceedings of the IEEE Visualization '97*, Phoenix, Arizona, 49–52.

BABARIA, K. 2004. Using treemaps to visualize gene ontologies. Human Computer Interaction Lab and Institute for Systems Research. University of Maryland, College Park, MD. Available at www.cs.umd.edu/hcil/treemap/GeneOntologyTreemap.pdf.

BAEHRECKE, E. H., DANG, N., BABARIA, K., AND SHNEIDERMAN, B. 2004. Visualization and analysis of microarray and gene ontology data with treemaps. BMC Bioinformatics. Available at http://www.biomedcentral.com/1471-2105/5/84.

BEAMTREES. http://www.win.tue.nl/~fvham/beamtrees/.

BOSCA, A., BOMINO, D., AND PELLEGRINO, P. 2005. OntoSphere: more than a 3D ontology visualization tool. In *Proceedings of SWAP, the 2nd Italian Semantic Web Workshop*, Trento, Italy, December 14–16, CEUR. Workshop Proceedings, ISSN 1613-0073, online http://ceur-ws.org/Vol-166/70.pdf.

BRULS, M., HUIZING, K., AND VAN WIJK, J. J. 2000. Squarified treemaps, data visualization. In *Proceedings of the joint Eurographics and IEEE TCVG Symposium on Visualization*. Springer, Vienna, 33–42.

CARRIERE, J. AND KAZMAN, R. 1995. Interacting with huge hierarchies: Beyond cone trees. In *Proceedings of InfoViz'95, IEEE Symposium on Information Visualization*, Atlanta, Georgia, 30–31. IEEE Computer Society Press, 74–78. Available at http://citeseer.ist.psu.edu/ere95interacting.html.

COCKBURN, A., AND MCKENZIE, D. 2000. An evaluation of cone trees, In *People and Computers XV, Proceedings of the 2000 British Computer Society Conference on Human Computer Interaction*. University of Sunderland. Springer-Verlag, http://citeseer.ist.psu.edu/cockburn00evaluation.html.

COCKBURN, A., AND MCKENZIE, D. 2002. Evaluating the effectiveness of spatial memory in 2D and 3D physical and virtual environments. In *Proceedings of ACM Computer-Human Interaction Conference on Human Factors in Computing Systems*. ACM Press, 203–210.

CROPCIRCLES. http://www.mindswap.org/2005/cropcircles.

EKLUND, P. 2002. Visual displays for browsing RDF documents. In *Proceedings of the 7th Australasian Document Computing Symposium*, Sydney, Australia.

EKLUND, P. W., ROBERTS, N., AND GREEN, S.P. 2002. OntoRama: Browsing an RDF ontology using a hyperbolic-like browser, In *Proceedings of the First International Symposium on CyberWorlds (CW2002)*. Theory and Practices, IEEE press, 405–411.

ERNST, N. A. AND STOREY, M.-A. 2003. *A Preliminary Analysis of Visualization Requirements in Knowledge Engineering Tools*. University of Victoria.

EYL, M. 1995. *The Harmony Information Landscape: Interactive, Three Dimensional Navigation Through an Information Space*. Master's thesis, Graz University of Technology, Austria.

FEKETE, J.-D. AND PLAISANT, C. 2002. Interactive information visualization of a million items. In *Proceedings of IEEE Symposium on Information Visualization*, Boston, 117–124. Available at http://citeseer.ist.psu.edu/fekete02interactive.html.

GENE ONTOLOGY CONSORTIUM. http://www.go.org.

GOBAR. http://katahdin.cshl.org:9331/GO.

GOLEMATI, M., HALATSIS, C., VASSILAKIS, C., AND KATIFORI, A. 2006. A context-based adaptive visualization environment. In Proceedings *of the 10th Information Visualization Conference*, IV06, London.

GOMINER. http://discover.nci.nih.gov/gominer/.

GOPHERVR. ftp://boombox.micro.umn.edu/pub/gopher/Unix/GopherVR/ and ftp://boombox.micro.umn.edu/pub/gopher/Macintosh-TurboGopher/TurboGopherVR/.

GOSURFER. http://www.gosurfer.org.

GRAPHVIZ. http://www.graphviz.org/.

GROKKER. http://www.groxis.com.

GRUBER, T. R. 1993. A translation approach to portable ontology specifications, knowledge acquisition. *Special issue: Current Issues in Knowledge Modelling*, Vol 5, Issue 2, 199–220.

HERMAN, I., MELANÇON, G., AND MARSHALL, M. S. 2000. Graph visualization and navigation in information visualization: A survey. *IEEE Trans. Visual. Comput / Graph.* Vol. 6, No. 1, January–March. 24–43.

HICKS, M., O'MALLEY, C., NICHOLS, S., AND ANDERSON, B. 2003. Comparison of 2D and 3D representations for visualizing telecommunication usage. *Behav. Inform. Tech.*, Vol. 22, No. 3, 185–201.

JANKUN, K. T. J. AND KWAN, L. M. 2003. MoireGraphs: Radial focus+context visualization and interaction for graphs with visual nodes. In *Proceedings of IEEE Symposium on Information Visualization*. Seattle, Washington. 20–21.

JEONG, C. AND PANG, A. 1998. Reconfigurable disc trees for visualizing large hierarchical information space. In *Proceedings of Information Visualization*. 19–25.

KAON. http://kaon.semanticweb.org/.

KATIFORI, A., TOROU, E., HALATSIS, C., VASSILAKIS, C., AND LEPOURAS G. 2006a. A comparative study of four ontology visualization techniques in Protégé: Experiment setup and preliminary results. In *Proceedings of the 10th Information Visualization Conference*, London.

KATIFORI, A., VASSILAKIS, C., LEPOURAS, G., DARADIMOS, I., AND HALATSIS, C. 2006b. Visualizing a temporally-enhanced ontology. In *Proceedings of the AVI Conference*, May 23–26, Venice, Italy.

KEIM, D. A. 2002. Information visualization and visual data mining. In *IEEE Trans. Visual. Comput. Graph*. Vol. 7, No. 1, January-March.

KLEIBERG, E., VAN DE WETERING, H., AND VAN WIJK, J. J. 2001. Botanical visualization of huge hierarchies. In *Proceedings of the IEEE Symposium on Information Visualization (InfoVis'2001)*. IEEE Computer Society Press.

KOBSA, A. 2004. User experiments with tree visualization systems. In *IEEE Symposium on Information Visualization (INFOVIS'04)*. 9–16.

LAMPING, J. AND RAO, R. 1996. The hyperbolic browser: A focus + context technique for visualizing large hierarchies. *J. Visual Lang. Comput.*, vol. 7, 33–55.

LEE, J. S. M., KATARI, G., AND SACHIDANANDAM, R. 2005. GObar: *A Gene Ontology-Based Analysis and Visualization Tool for Gene Sets*. BMC Bioinformatics.

LEE, B. PARR, C., PLAISANT, C., BEDERSON, B. B., VESKLER, V. D., GRAY, W. D., AND KOTFILA, C. 2006a. TreePlus: Interactive exploration of networks with enhanced tree layouts. In *IEEE TVCG Special Issue on Visual Analytics*. Available at http://hcil.cs.umd.edu/trs/2006-04/2006-04.pdf.

LEE, B., PLAISANT, C., PARR, C., FEKETE, J., AND HENRY, N. 2006b. Task taxonomy for graph visualization. In *Proceedings of the 2006 AVI Workshop on BEyond Time and Errors: Novel Evaluation Methods for Information Visualization*. Venice, Italy. 1–5.

LIEBIG, T. AND NOPPENS, O. 2004. OntoTrack: Combining browsing and editing with rasoning and explaining for OWL lite ontologies. In *Proceedings of the 3rd International Semantic Web Conference ISWC 2004*. Hiroshima, Japan. 8–11.

MUNZNER, T. 1997. H3: Laying out large directed graphs in 3D hyperbolic space. In *Proceedings of the 1997 IEEE Symposium on Information Visualization*, Phoenix, AZ. 2–10.

MUNZNER, T. 1998. Exploring large graphs in 3D hyperbolic space. IEEE *Comput. Graph. Appl*. Vol. 18, No. 4, 18–23.

NOY, N. F., FERGERSON, R. W., AND MUSEN, M. A. 2000. The knowledge model of Protege-2000: Combining interoperability and flexibility. In *Proceedings of the 2nd International Conference on Knowledge Engineering and Knowledge Management (EKAW'2000)*, Juan-les-Pins, France.

NOY, N. F., KUNNATUR, S., KLEIN, M., AND MUSEN, M. A. 2004. Tracking changes during ontology evolution. In *Proceedings of the Third International Conference on the Semantic Web (ISWC-2004)*, Hisroshima, Japan.

NOY, N. F. AND MCGUINESS D. L. 2001. *Ontology Development 101: A Guide to Creating Your First Ontology*, Stanford Knowledge Systems Laboratory Tech. Rep. KSL-01-05 and Stanford Medical Informatics Tech. Rep. SMI-2001-0880, March.

ONTORAMA. http://www.ontorama.com.

ONTOSPHERE. http://ontosphere3d.sourceforge.net/.

ONTOTRACK. http://www.informatik.uni-ulm.de/ki/ontotrack.

OZONE. http://www.cs.umd.edu/hcil/ozone/.

PARSIA, B., WANG, T., AND GOLDBECK, J. 2005. Visualizing Web ontologies with cropCircles. In *Proceedings of the 4th International Semantic Web Conference*, 6–10.

PIETRIGA, E. IsaViz, http://www.w3.org/2001/11/IsaViz/.

PLAISANT, C., GROSJEAN, J., AND BEDERSON, B. B. 2002. SpaceTree: Supporting exploration in large node link tree, design evolution and empirical evaluation. In *Proceedings of IEEE Symposium on Information Visualization*, Boston, 57–64.

PROTÉGÉ PROJECT. Stanford University, http://protege.stanford.edu.

REKIMOTO, J. AND GREEN, M. 1993. The Information Cube: Using transparency in 3D information visualization. In *Proceedings of the Third Annual Workshop on Information Technologies and Systems (WITS'93)*, 125–132. http://www.csl.sony.co.jp/person/rekimoto/cube.html.

RICARDO, C. A., LUZZARDI, P. R. G., AND FREITAS, C. M. D. S. 2002. The Bifocal Tree: A technique for the visualization of hierarchical information structures. In *Proceedings of Workshop on Human Factors in Computer Systems (IHC2002),* Fortaleza, Brazil.

RIVADENEIRA, W. AND BEDERSON, B. B. 2003. *A Study of Search Result Clustering Interfaces: Comparing Textual and Zoomable Interfaces*, University of Maryland HCIL Tech. Rep. HCIL-2003-36, October.

ROBERTSON, G. G., CAMERON, K., CHERWINSKI, M., AND ROBBINS, D. 2002. Polyarchy Visualization: Visualizing multiple intersecting hierarchies. In *Proceedings of the Conference on Human Factors in Computing Systems (CHI'02)*, 423–430. http://research.microsoft.com/users/marycz/chi2002poly.pdf.

ROBERTSON, G. G., MACKINLAY, J. D., AND CARD, S. K. 1991. Cone Trees: Animated 3D visualizations of hierarchical information. In *Proceedings of the CHI '91 Human Factors in Computing Systems*. ACM, New York, 189–202.

SEQUOIAVIEW. http://www.win.tue.nl/sequoiaview/.

SHNEIDERMAN, B. 1992. Tree visualization with tree-maps. A 2-d space-filling approach. *ACM Trans. Graph.*. Vol. 11, No. 1, September, 92–99.

SHNEIDERMAN, B. 1996. The eyes have it: A task by data type taxonomy for information visualizations. In *Proceedings of 1996 IEEE Visual Languages*. IEEE, 336–343.

SINTEK, M. 2003. Ontoviz tab: Visualizing Protégé ontologies, http://protege.stanford.edu/plugins/ontoviz/ontoviz.html.

SMALLMAN, H. S., ST. JOHN, M., OONK, H. M., AND COWEN, M. B. 2001. Information availability in 2D and 3D displays, *IEEE Comput. Graph. Appl.*, vol. 21, no. 5, pp. 51–57, Sept/Oct.

SOUZA, K. X. S., DOS SANTOS, A. D., AND EVANGEISTA, S. R. M. 2003. Visualization of ontologies through hypertrees. In *Proceedings of the Latin American Conference on Human-Computer Interaction*, Rio de Janeiro, Brazil. 251–255.

SPACETREE. http://www.cs.umd.edu/hcil/spacetree/.

STARTREE. http://www.inxight.com/.

STEVEN, D. AND PERRIN, J. 2004. PROMPT-Viz: *Ontology Version Comparison Visualizations with Treemaps*. Master Of Science Thesis in the Department of Computer Science, University of Victoria. Retrieved from http://www.cs.uvic.ca/~chisel/thesis/David_Perrin_Thesis.pdf.

STOREY, M.-A., MUSSEN, M., SILVA, J., BEST, C., ERNST, N., FERGERSON, R., AND NOY, N. 2001. Jambalaya: Interactive visualization to enhance ontology authoring and knowledge acquisition in Protégé. In *Proceedings of Workshop on Interactive Tools for Knowledge Capture, K-CAP-2001*, Victoria, BC, Canada, http://www.thechiselgroup.org/jambalaya.

STRASNICK, S. L. AND TESLER, J. D. 1996. *Method and Apparatus for Displaying Data Within a Three-Dimensional Information Landscape*. US Patent 5,528,735, Silicon Graphics, Inc., June. Filed 23rd March 1993, granted 18th June, 1996.

SUH, B. AND BEDERSON, B. B. 2002. OZONE: A zoomable interface for navigating ontology information. In *Proceedings of Advanced Visual Interfaces*. ACM.

SURE, Y., ANGELE, J., AND STAAB, S. 2002. OntoEdit: Guiding ontology development by methodology and inferencing. In *Proceedings of International Conference on Ontologies, Databases and Applications of Semantics (ODBASE'02)*, Irvine.

TAO, Y., LIU, Y., FRIEDMAN, C., AND LUSSIER, A. Y. 2004. Information visualization techniques in bioinformatics during the postgenomic era. *BIOSILICO*, Vol. 2, No. 6, 237–245.

TOUCHGRAPH. http://www.touchgraph.com/.

TREEMAP. http://www.cs.umd.edu/hcil/treemap.

VAN HAM, F AND VAN WIJK, J. J. 2002. Beamtrees: Compact visualization of large hierarchies. In *Proceedings of the IEEE Conference on Information Visualization*. IEEE CS Press, 93–100.

VAN WIJK, J. J. AND VAN DE WETERING, H. 1999. Cushion Treemaps: Visualization of hierarchical information. In *Proceedings of the IEEE Symposium on Information Visualization (InfoVis'99)*. IEEE Computer Society, 73–78.

WANG T. AND PARSIA, B. 2006. Cropcircles: topology sensitive visualization of owl class hierarchies, in *Proceedings of the International Semantic Web Conference (ISWC 06)*, http://www.mindswap.org/papers/2006/cropcircles-iswc.pdf.

WISS, U. AND CARR, D. 1998. *A Cognitive Classification Framework for 3-Dimensional Information Visualization*. Research Report LTU-TR—1998/4—SE, Luleå University of Technology.

WISS, U., CARR, D., AND JOHNSON, H. 1998. Evaluating three-dimensional visualization designs: A case study of three designs. In *Proceedings of the Second International Conference on Information Visualisation (IV'98)*. p. 137.

WOLTE, J. 1998. *Information Pyramids—Compactly Visualizing Large Hierarchies*, Master's thesis at Graz University of Technology, Institute for Information Processing and Computer Supported New Media (IICM), Graz University of Technology A-8010 Graz, Austria.

WU, J. AND STOREY, M.-A. 2000. A multi-perspective software visualization environment. In *Proceedings of the 2000 Conference of the Centre for Advanced Studies on Collaborative Research*. ACM.

YOUNG, P. 1996. *Three Dimensional Information Visualization*. Computer Science Tech. Rep. 12/96, November 1.

ZHONG S., STORCH, F., LIPAN, O., KAO, M. J., WEITZ, C., AND WONG, W. H. 2004a. GoSurfer: A graphical interactive tool for comparative analysis of large gene sets in gene ontology space. *Applied Bioinformatics*, 3(4): 1–5.

ZHONG, S., TIAN, L., LI, C., STORCH, K. F., AND WONG, W. H. 2004b. Comparative analysis of gene sets in the gene ontology space under the multiple hypothesis testing framework. In *Proceedings of the 2004 IEEE Computational Systems Bioinformatics Conference*.

Received November 2005; revised July 2006, February 2007; accepted March 2007