OntoSphere: more than a 3D ontology visualization tool

Alessio Bosca[1], Dario Bonino[1], Paolo Pellegrino[1]

[1] Politecnico di Torino, Torino, Italy
{alessio.bosca, dario.bonino, paolo.pellegrino}@polito.it

Abstract. In this paper a new approach to ontology visualization is introduced, using a more than 3-dimensional space. The ontology information is represented on a 3D view-port enriched by visual cues. A preliminary application based on the presented principles is described, and results show that the approach is feasible and can actually lead to a more effective modeling process and to an easy detection of "conceptual inconsistencies". The opinions of domain experts are also quite positive and seem to indicate that the approach is promising and can be extended for creating "user friendly" ontology editors for people with low ICT skills.

Introduction

In the last years the Semantic Web has been constantly evolving from a vision of few people to a tangible presence on the Web, with many tools, portals, ontologies, etc. Such a great evolution involved many researchers, from different countries, and has been primarily focused on technologies. At now a Web developer can start to seriously consider the opportunity to provide semantically tagged content as the needed tools and standards are available. However, the current web panorama shows a very little adoption of semantics. The motivations for such a low adoption can be various and related to very different aspects: technology immaturity, failing dissemination, user and developer resilience to changes, etc. In the sea of possible failures and shortcomings, interfaces have a relevant role often discriminating good solutions from bad ones. This is particularly true for tools related with knowledge modeling and visualization, where the involved information can be quite complex and involve multidimensional issues.

Several attempts aim at providing effective interfaces for knowledge modeling, i.e. for ontology creation and visualization. Protégé and OntoEdit for example are complete IDEs (Integrated Development Environments) that address in a single application all the aspects related to ontology creation, checking and visualization (through proper plug-ins). Such tools, although adopting rather different paradigms for editing and inspecting ontologies, have in common a bi-dimensional approach to ontology visualization. The bi-dimensional approach can be variously efficient and there are actually good solutions available on the web: GraphViz, Jambalaya and OntoViz, just

for naming a few. Nevertheless, mapping the many dimensions involved by an ontology like the concepts hierarchy, the semantic relationships, the instances and the possible axioms defining a given knowledge domain, on only two dimensions can sometimes be too restrictive.

In this paper we propose a new approach and a new application called OntoSphere for inspecting and, in a near future, for editing ontologies using a more than 3-dimensional space. The proposed approach visualizes the mere topological information on a 3D view-port, thus leveraging one more dimension with respect to the current solutions. This allows, at least, to better organize the visual occupation of represented data. Being the 3-dimensional view quite natural for humans, especially for what concerns navigation, the proposed approach can be more effective in browsing as it involves "manipulation-level" operations such as zooming, rotating, and translating objects. In addition, many more dimensions are introduced to convey information on the visualized knowledge model (meta-information). The extension of the subtrees lying under the currently viewed concepts is, for example, visually rendered by increasing the size of the visual cues adopted for them. The same approach is applied to colors, which are used to add insights on the representation: blue spheres, for example, indicate that the corresponding concepts are terminal nodes in the ontology. Transparency is used for distinguishing inherited, or inferred, from direct relationships; shapes are used for differentiating concepts and instances and so on.

Together with the ways to convey more information to the users through several visual dimensions, the proposed work also aims at tackling the space allocation issues for ontology visual models. In fact, in the traditional solutions, big ontologies can easily lead to overcrowded representations that are difficult to browse and that can be more confusing than aiding. Some attempts exist to overcome these problems, as in OntoRama, where the nodes being inspected are magnified with respect to the other nodes in the ontology. However, even these approaches tend to collapse when visualizing big ontologies such as SUMO, counting over than 20'000 concepts. The proposed application, instead adopts a dynamic collapsing mechanism and different views, at different granularities, for granting a constant navigability of the rendered model.

The paper is organized as follows: section 2 presents some related works while section 3 introduces the adopted approach and details the several solutions introduced for conveying information in efficient ways. Section 4 describes the technical implementation of the OntoSphere application and provides some preliminary results. Eventually section 5 provides conclusions and proposes some future works.

Related Work

The existing techniques for the visualization of ontologies can be summarized in four main visual schemes, possibly cooperating in more complex scenarios: network, tree, neighborhood, and hyperbolic. The network view represents an ontology as a generic network of connected elements and is usually exploited when the knowledge elements cannot be conveniently organized in hierarchies. The tree (or hierarchical) view, instead, is generally used for more structured ontologies. However, the simple

hierarchical representation provided by this view is unable to represent connections between distinct sub-trees that violate the dominant taxonomic structure. In such a case, the connections violating the hierarchy are indicated in separate views, so complicating the navigation of the structure. The most common examples of tree views are based on indentation, as in file system browsers, or on diagrams with nodes and arcs. However, a treemap view has also been proposed by Schneiderman [1], at the Maryland University, which uses nested rectangles to represent sub-classes (Figure 1, C).

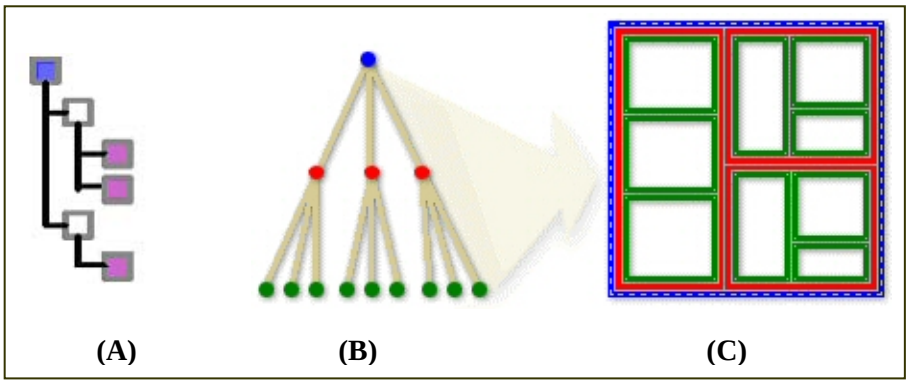

Figure 1 - TreeViews: indented (A), nodes and arcs (B), TreeMap (C)

The main advantage of tree views is that they can be displayed with rather little effort in comparison with network-oriented views. More importantly, entire sub-trees can be easily collapsed (i.e., temporarily hidden) to concentrate the attention on the rest of the knowledge base. The next two schemes apply similar principles on network-based structures: in fact, both the neighborhood and the hyperbolic views (Figure 2) focus the attention on a chosen node and its nearest neighbors. In the former case only the semantically nearest nodes are displayed, whereas in the latter case the nodes are displaced onto a hemi-spherical surface, projected onto the visual window, therefore magnifying the central nodes while shrinking the peripheral nodes.

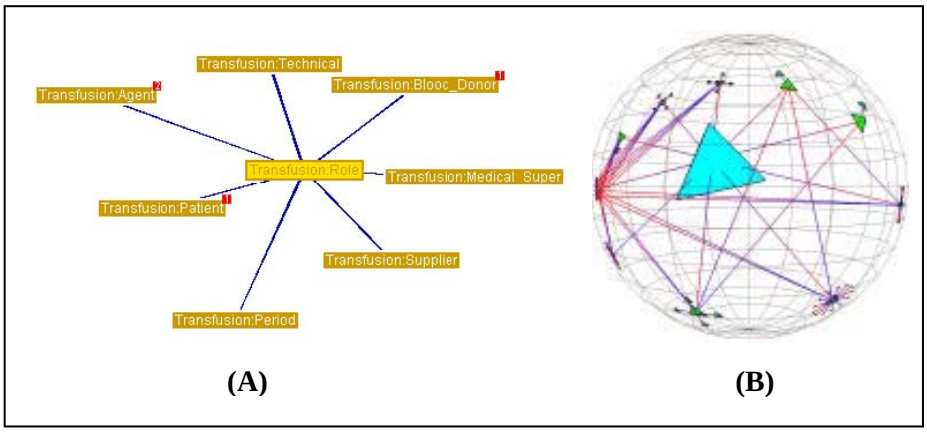

Figure 2 - Neighborhood View (A), Hyperbolic View (B)

The aforementioned representation schemes have been utilized in numerous applications with assorted enhancements. Some of them are discussed in the next paragraphs.

Protégé [12, 6] is an open source ontology editor providing support for knowledge acquisition. Its framework natively allows the interactive creation and visualization of classes in a hierarchical view. Each concept in the tree can be displayed along with additional information about the related classes, properties, descriptions, etc., which can all be quickly edited. Other panels manage class instances, alternative user interfaces, queries, and possibly other extensions which can be easily added to the framework as plug-ins. Particularly, various plug-ins are available for enhancing the visualization of the ontology and are therefore here discussed.

The OntoViz [7] plug-in displays a Protégé ontology as a graph by exploiting an open source library optimized for graph visualization (GraphViz [5]). Intuitively, classes and instances are represented as nodes, while relations are visualized as oriented arcs. Both nodes and arcs are labeled and displaced in a way that minimizes overlapping, but not the size of the graph. Therefore, the navigation of the graph, enhanced only by magnification and panning tools, does not provide a good overall view of the ontology, as the graphical elements easily become indistinguishable.

This problem is less critical in Jambalaya [3, 8], another ontology viewer for Protégé, based on a treemap scheme or rather *nested interchangeable views*, namely Simple Hierarchical Multi-Perspective (SHriMP). SHriMP is a domain-independent visualization technique designed to enhance how people browse and explore complex information spaces. An animated view of the ontology graph facilitates the navigation and browsing at different levels of abstractions and details, both for classes and relations, while keeping low the learning curve through well-known zooming and hypertext link paradigms. However, text labels and symbols tend to overlap when the ontology grows in complexity and it is difficult to understand the relations among classes or instances.

TGViz [9], similarly to OntoViz, visualizes Protégé ontologies as graphs. In this case however, the displacement of nodes and arcs is computed using the spring layout algorithm implemented in the Java TouchGraph library [16]. The main advantage of

this approach is the optimized exploitation of the bi-dimensional space in which the nodes and arcs are dynamically distributed. However, the level of detail is not adjusted according to the level of zoom, often resulting in overcrowded pictures.

The ezOWL [10] plug-in, differently from the previous viewers, enhances Protégé with a graph-based editing of ontologies, though reducing to a minimum the optimizations for the graph organization. Even in this case it may be difficult to maintain both a good understanding of the overall ontology and a sufficient level of detail about a chosen sub-graph.

OntoEdit [11] is a commercial Java-based tool that, similarly to Protégé, offers a graphical environment for the management and development of ontologies, and can be enhanced with various plug-ins. In particular, the Visualizer plug-in proposes a bi-dimensional graph-based view of the ontology using colored icons as nodes accompanied by contextual tooltips, such as colored borders or spots other than the usual labels, which unfortunately are often hidden or overlapping.

IsaViz [12] is another graph-based visual editor for RDF models based on the GraphViz library. In this case, the principal enhancement to the previously mentioned approaches based on graphs is the Radar View, which, similarly to Jambalaya, displays a simplified network overview of the overall ontology in a small window, highlighting the currently edited region in a rectangle. In addition, icons and colors are also exploited to concentrate information, while different visualization styles and layouts are supported through the GSS (Graph Style Sheet) language, derived from the well-known CSS (Cascading Style Sheet and SVG (Scalable Vector Graphics) W3C recommendations. However, it is still not possible to customize the level of details for big ontologies.

OntoRama [15] is an ontology browser for RDF models based on a hyperbolic layout of nodes and arcs. As the nodes in the center are distributed on more space than those near to the circumference, they are visualized with a higher level of detail, while maintaining a reasonable overview of the peripheral nodes. In addition to this pseudo-3D space, OntoRama also introduces the idea of *cloned* nodes. Since the browser supports generic ontologies, with properties for classes, multiple relations, sub-classing, and multiple inheritance, certain nodes and their sub-trees are cloned and visualized multiple times so that the number of crossed arcs can be reduced, and the readability enhanced. The duplicate nodes are displayed using an ad-hoc color in order to avoid confusion. Unfortunately, this application does not support editing and can only manage RDF data.

Eventually, the approaches and functionalities for each of the mentioned application are summarized in the following table.

		View scheme			
Viewer / Plugin	**Editor**	**Network**	**Hierarchical**	**Neighborhood**	**Hyperbolic**
Protégé	✓		✓		
OntoViz	×	✓			
Jambalaya	×	✓	✓	✓	
TGViz	×	✓		✓	
ezOwl	✓	✓			
OntoEdit	✓		✓		
Visualizer	✓	✓		✓	

IsaViz	✓	✓			
OntoRama	×	✓	✓	✓	✓

Proposed Approach.

Applications for ontology visualization abstract from the formalism of the underlying data and graphically model the information contained in the knowledge base. Each tool presents the data according to its own approach but generally all of them share the same choice of a 2D space. Well known and widespread visualization tools as those mentioned in the previous section represent ontologies through a bi-dimensional paradigm; our approach, instead, leverages the use of a 3D space as a means to effectively represent and explore data through an intuitive interface.

Our application objective consists in enhancing the performances of current solutions in terms of completeness and readability; in fact OntoSphere application aims to graphically represent both the taxonomic and the not taxonomic links as well as selecting and presenting information on the screen at an appropriate detail level according to what is relevant to the user's interest. Furthermore, the tool intends to provide an intuitive navigation interface, featuring direct manipulation of the scene (rotation, panning, zoom, object selection, etc.) and designed to particularly meet the demands of domain experts who have little technical skills in the field of Semantic Web, and therefore specifically rely on graphical interfaces.

The choice of a three-dimensional environment constitutes our starting point in designing the tool, as a 3D space offers one more dimension than traditional 2D approaches to represent ontology data, so simplifying its interpretation.

In order to achieve completeness and readability, OntoSphere operates according to two different principles:

- To increase the number of "dimensions" (colors, shapes, transparency, etc.) which represent concepts features and convey additional information without adding the burden of further graphical elements, such as labels, on the scene.
- To automatically select which part of the Knowledge Base has to be displayed and the detail level that has to be used in the process, on the base of user interaction with the scene.

In particular, the latter guideline is particularly important for improving overall system performances since scale factor indeed constitutes a strong issue in visualizing complex graph structures like ontologies. As cardinality of elements increases, the number of items to be concurrently displayed on the screen worsens the graphical perception of the scene and complicates spotting details. When the amount of visualization space needed to represent all the information within the KB outnumbers the one available on the screen, a few options remain available: to scale down the whole image to the detriment of readability, to present on the screen just a portion of it and allow its navigation or to summarize the information in a condensed graph and

provide means for its exploration and expansion. As the effectiveness of these options depends on the use case involved (consistency checking, domain comprehension, KB updates) a combined usage of them can offer a better approach.

In order to fulfill the demanded requirements, our solution consists in exploiting different scene mangers (RootFocus, TreeFocus and ConceptFocus SceneManagers) that present and organize the information on the screen, each one of them according to a differently detailed perspective. Such scenes interchange in managing the graphical space as user attention shifts from one concept to another and the attention focus is implicitly inferred from user's interaction with the scene (e.g., a concept selection with a mouse click). In this way we introduce the idea of "focusing" as the application capability of highlighting the elements of interest while leaving out the others.

The user interface we propose is very simple and pursues direct manipulation of the scene as rotations, panning and zoom; it allows to browse the ontology as well as to update it and to add new concepts and relations. Every concept within a given scene is clickable with two different results: a left click performs a focusing operation, shifting the scene to a more detailed level, while a right click maintains the current perspective and simply navigates through elements. For example, left-clicking on a concept in the global scene would lead to the visualization of the related concept tree while right clicking on it would lead to the visualization of its children in the same "earth-like" perspective, as explained in more details in the next sections.

RootFocus Scene

This perspective presents a big "earth-like" sphere bearing on its surface a collection of concepts represented as small spheres (Figure 3). The scene does not visualize any taxonomic information and only shows direct "semantic" relations between elements of the scene, usually a graph not fully connected.

Atomic nodes, the ones without any subclass, are smaller and depicted in blue while the others are colored in white and their size is proportional to the number of elements contained in their own sub-tree.

This view is particularly intended for representing the ontology primitives, i.e., the root concepts, but can also be used, during the navigation of the ontology, in order to visualize direct children of a given node; a pretty useful option in case of heavily sub-classed concepts.

Topmost concepts within the ontology and the relations between them define the conceptual boundaries of the domain and provide a very good hint to the question: "what's the ontology about?"

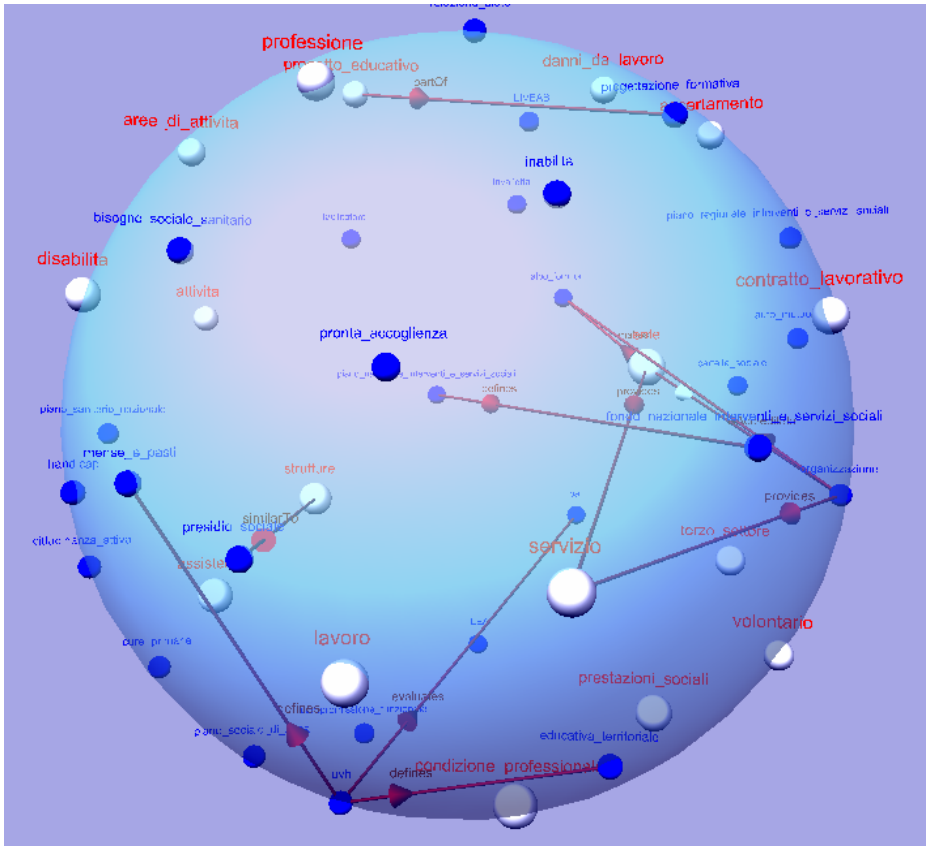

Figure 3 - Global view

TreeFocus Scene

The scene shows the sub-tree originating from a concept; it displays the hierarchical structure as well as semantic relations between classes. Since use evidence proves that too many elements on the screen, at the same time, hinder user attention, the scene completely presents only three fully-expanded levels at a time and, as user browses the tree, the system automatically performs expansion and collapse operations in order to maintain a reasonable scene complexity. The reader may note in Figure 4 how focusing the attention on the concept "*ente_pubblico_locale*", on the left in the figure, causes (with a simple mouse click) the vanishing of the uninterested branches, then collapsed in their respective parents. Collapsed elements are colored in white and their size is proportional to the number of elements present in their sub-tree; instead concepts located at the same depth level within the tree have the same color in order to easily spot groups of siblings. Relations of type *Is_a* within the scene are displayed with a neutral color (gray) and

without label, whereas other semantic relations involving two concepts already in the scene are displayed in red, accompanied by the name of the relation (as in the perspective described in the previous section). Otherwise, if an element of the tree is related to a node that is not present on the scene a small sphere is added for that node in the proximity of the given element, so terminating the end of the arrow: in such cases, incoming relations are represented with a green arrow, while outgoing links with a red one.

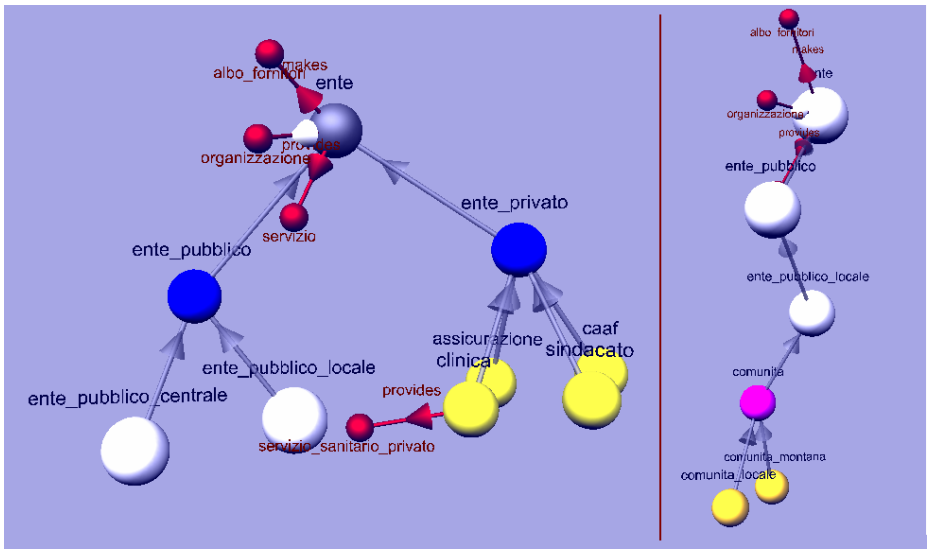

Figure 4 - Sub-tree navigation scenes

ConceptFocus Scene

The perspective depicts all the available information about a single concept, at the highest possible level of detail; it reports the concept's children and parent(s), its ancestor root(s) and its semantic relations, both the ones directly declared for the given concept and the ones inherited from its ancestors. Semantic relations are drawn as arrows terminating in a small sphere (Figure 5): red if the relation is outgoing and green otherwise. Direct relations are drawn close to the concept and with an opaque color, while inherited ones are located a bit farther from the center and depicted with a fairly transparent color.

This scene is pretty useful during consistency checking operations because it ease the spotting of inconsistent concept or relations whenever a concept inherits from an ancestor a property that "conceptually" contrasts with other features of its own.

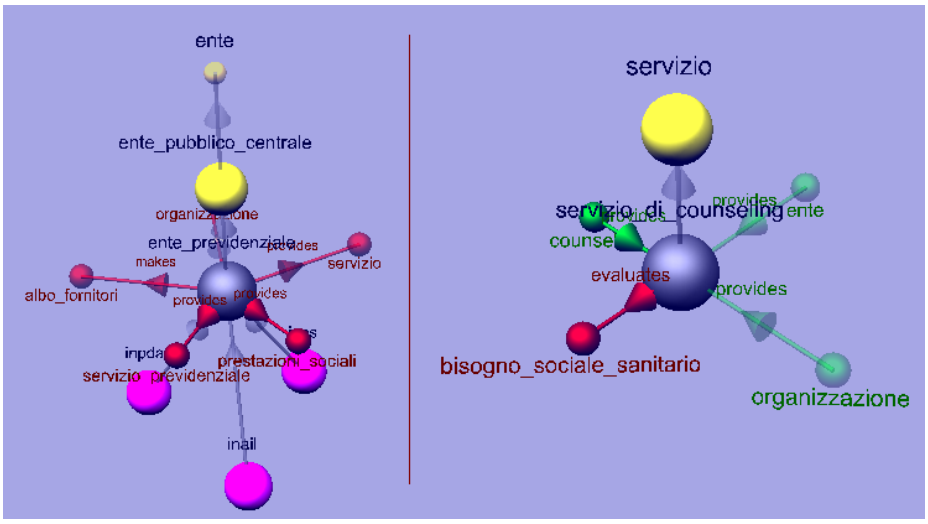

Figure 5 - Detailed concept

Implementation and preliminary results

The work presented in this paper has been entirely developed in Java. The choice is related to the current panorama of ontology editors and of tools for ontology creation and maintenance, which are in the majority of cases developed in this language. Among the other advantages, Java permits to use such tools in different operating environments, from devices with low computational power, to high performance workstations.

The visualization engine uses the Java 3D API to produce a three-dimensional interactive representation of ontology concepts and relationships. This API is directly linked with an underlying Open GL engine that provides the required graphics capabilities. The ontology related part, instead, is based upon the well-known Jena semantic framework from HP which allows to easily load, manage and modify ontologies and taxonomies written either in RDF, RDFS, DAML, OWL or N-triple. These two main parts are the core modules of the implemented application, conciliating in a single working space capabilities for visualizing and editing ontologies in various formats.

In order to understand if the proposed approach is valuable and scalable enough, the authors set up three different test beds. The first one assesses the compliance of the tool with the initial requirements; the second one evaluates the tool utility when it is applied to real world cases and the last test bed investigates whether the current deployment is able to manage complex ontologies or not.

In the first experiment the application has been tested according to the standards for agile development and for requirements satisfaction checking. All the modules composing the platform have been developed starting from a rather precise specification

and have been tested according to pre-defined J-Unit tests. After passing the basic functionality checking, the entire application has been tested against three different use cases including: simple ontology browsing, "conceptual consistency" checking and ontology development. In the ontology browsing case, a group of 8 users has been required to load and browse 5 different ontologies. The goal was to guess the domain of the chosen ontology and to analyze the granularity of the knowledge model. The 5 ontologies used in this experiment are: the well-known Pizza ontology from Protègè, the SUMO (Suggested Upper Merged Ontology) ontology, a music ontology developed from scratch by the authors, the CABLE ontology and the Passepartout ontology developed by the authors in collaboration with the Passepartout service of the Turin's municipality.
Results for each of the ontologies are reported in Table 1.

Table 1. Results for the ontology browsing use case

Ontology	**Topics involved**	**Topic and level of detail identification**							
		U1	U2	U3	U4	U5	U6	U7	U8
SUMO (OWL)	Many general topics	✓	×	×	✓	✓	×	✓	✓
Music (RDF/S)	Instruments, Music actors, Music genres, etc.	✓	✓	✓	✓	✓	×	✓	×
CABLE (OWL)		✓	✓	✓	✓	✓	✓	✓	✓
Passepartout (OWL , RDF/S)	Disability, Aids for disable people, bureaucratic issues related to handicap, etc.	✓	×	✓	×	×	✓	✓	✓
Pizza (OWL)	Pizzas…	✓	✓	✓	✓	✓	✓	✓	✓
× = not recognized ✓ = correctly recognized									

Checking the ontology for "conceptual consistency" is rather different from the formal consistency checking done by logic reasoners. What has to be checked is not the ontology consistence for reasoning and inference, but whether a user can detect domain-related inconsistencies created during the ontology design process. For example, a concept may inherit some relationships that are not appropriate for it, either because of a wrong parent-child relation or because of a previously undetected error in the domain modeling: in this case the ontology is formally consistent but not conceptually. The ontologies involved in this test were the same used in the previous one, as well as the users.
Some interesting aspects came out from the experimentation: the detection of "conceptual inconsistencies" through the observation of the ontology representation appears, in fact, strongly dependent on the dimension of the knowledge domain and on the expertise that the user has in that domain. So, for example, by looking at Table 2

it is clearly noticeable that in the SUMO ontology no inconsistencies were found, as it is, in fact, huge and well designed, while the involved testers had a very poor background on the SUMO domain. On the other side, in the CABLE ontology almost all inconsistencies were detected since the ontology is small (80 concepts) and the domain was well known by all the experimenters. In conclusion, determining whether the proposed OntoSphere application is or is not able to evidence inconsistence is very difficult, since the involved factors are diverse and can interact in complex patterns.

Table 2. The results of the "conceptual inconsistencies" checking.

Ontology	**Number of known conceptual inconsistencies**	**Number of detected inconsistencies**							
		U1	U2	U3	U4	U5	U 6	U 7	U8
SUMO (OWL)	? (not known)	0	0	0	0	0	0	0	0
Music (RDF/S)	6	0	0	4	2	6	0	4	2
CABLE (OWL)	2	2	2	2	2	0	2	0	2
Passepartout (OWL , RDF/S)	12	4	8	0	3	10	3	0	7
Pizza (OWL)	0	0	0	0	0	0	0	0	0

When the proposed application is used for ontology development, the support provided for detecting conceptual inconsistencies is much more evident. The adoption of OntoSphere for inspecting the work in progress allowed, in fact, to easily detect modeling errors. In particular, the mostly recognized errors were about relationship propagation along the ontology hierarchy and wrong definitions of parent-child (*isA*) relationships. Although it is quite difficult to fill-up a table for showing how, and to what extent, the proposed application supports the process of ontology creation, interviews with users evidenced that many times the experimenters were able to quickly spot the modeling errors. Their opinion indicated the intuitive visualization and the capability to visually represent inherited and inferred relationships as the main factors for achieving success in their own modeling process.

This last experiment actually lies between the functional tests and real world test cases. However, to provide a more grounded experimentation (please note that the results here presented are still very preliminary) the authors performed a real world test in the occasion of the final meeting of CABLE, a European MINERVA project on “CAse Based e-Learning for Educators” []. In that meeting, a demo of the OntoSphere application has been presented to visualize the ontology developed in the context of the CABLE project. The exciting result is that, rather than complaining about the complexity of the provided interface, or about the appearance or the controls for

browsing the ontology, the first observation was: “No! That relation can not subsist between those two concepts!” What surprisingly happened is that the application was able to highlight the inherited relations so that the errors were spotted in few minutes of ontology browsing. This is clearly a not scientific result since experiments are to be conducted in a controlled environment, shall have a clear objective and must be carried on by a significant group of users. And the aim of this paragraph is not to sustain the thesis that assumes such reaction as a good result. However, the user reactions in the CABLE meeting are encouraging signals that the still preliminary OntoSphere application can be a valuable instrument in ontology design and development.

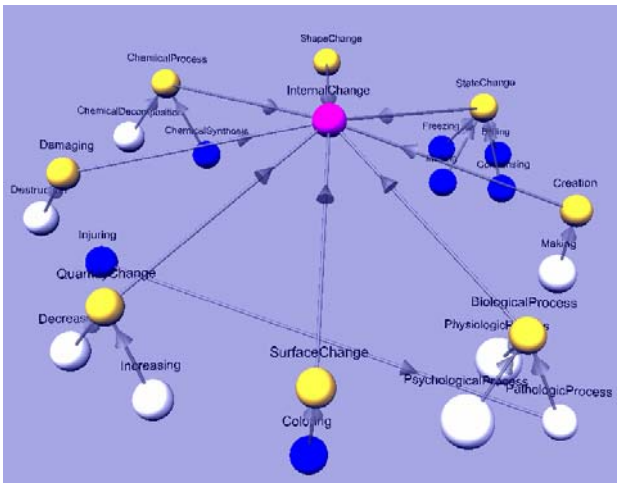

Figure 6. The Suggested Upper Merged Ontology viewed with OntoSphere.

As last experiment, a simple scalability test was performed: the goal was to understand if OntoSphere is able to load and visualize ontologies having great amounts of concepts and relationships. The entire SUMO ontology was therefore loaded and browsed and the loading process took around 3.5 seconds, while navigation was performed in real-time. SUMO is the Suggested Upper Merged Ontology and it is currently released under a GPL license. It counts about 20,000 concepts related by over 60,000 axioms. The screenshots show OntoSphere running with the SUMO ontology and visualizing the “Internal Change” branch that stems from the root concept “Process” (Figure 6), and the sub-tree lying under the ontology root concept “Entity” (Figure 7).

There are still some issues to be fixed when browsing really huge ontologies: the visualized concepts tend to clash if the number of concepts visualized at the same time is high. Also the labels tend to overlap making the visualization more difficult to manage (as in many other viewers). Moreover, since a human cannot take into account more than a reasonable number of objects at a time, huge graphs shall be collapsed and different ontology navigation patterns and interfaces shall be provided.

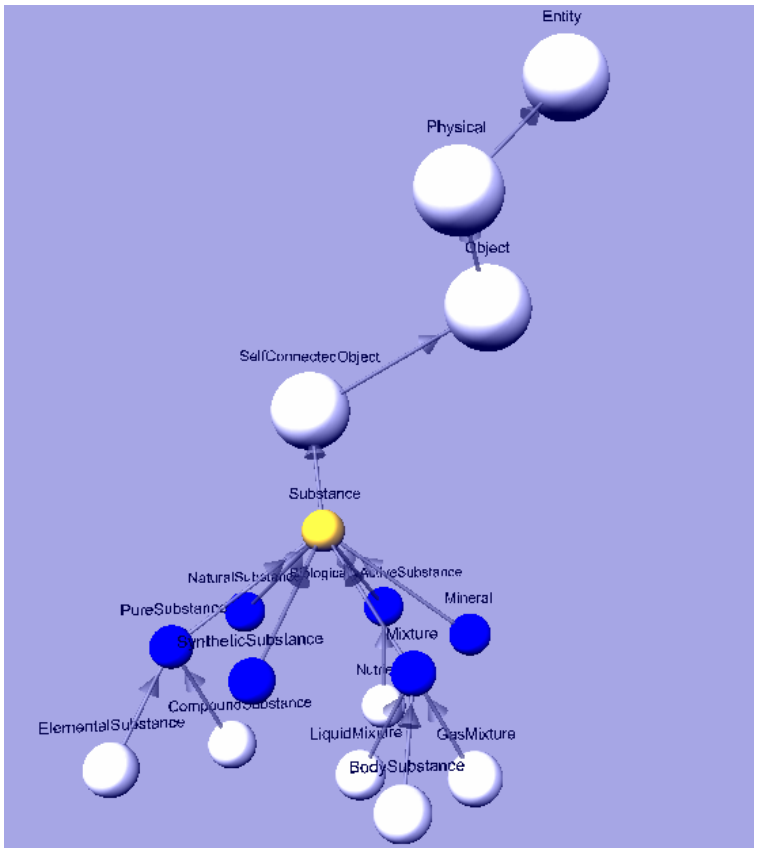

Figure 7. Another view of the SUMO ontology in OntoSphere.

Conclusions

In this paper the authors presented OntoSphere: a 3D ontology visualization tool. Some very preliminary results are provided showing that the proposed approach is feasible and that is judged valuable by domain experts that need to develop and review ontologies. The tool proved to be able to scale up to really huge ontologies, such as SUMO, preserving reasonable loading times and with enough fluidity in navigation.

However, there are still many issues to be addressed: concept clashes shall be resolved in a more efficient way, possibly avoiding label overlap, navigation through the ontology graph needs further refinements, making the entire process more usable and customizable, etc.

At now, the authors are working on some of these improvements and are transforming the OntoSphere application into a Protégé plug-in that will be released under the

LGPL public license. Future works will include a more precise and extensive test campaign and a user centered design of navigation interfaces.

References

1 Ben Shneiderman, “Treemaps for space-constrained visualization of hierarchies” , ACM Transactions on Graphics (TOG) Volume 11 , Issue 1 (January 1992) Pages: 92 - 99 Year of Publication: 1992.

2 Holger Knublauch. An AI tool for the real world: Knowledge modeling with Protégé. JavaWorld, June 20, 2003.

3 M.A. Storey, M. Musen, J. Silva, C. Best, N. Ernst, R. Fergerson, and N. Noy. Jambalaya: Interactive visualization to enhance ontology authoring and knowledge acquisition in Protégé. In Workshop on Interactive Tools for Knowledge Capture, Victoria, B.C. Canada, October 2001.

4 P.W.Eklund, N.Roberts, S.P.Green, OntoRama: Browsing an RDF Ontology using a Hyperbolic-like Browser, *The First International Symposium on CyberWorlds (CW2002)*, pp.405-411, Theory and Practices, IEEE press, 2002

5 Gansner, E. R. & North, S. C. (1999), An open graph visualization system and its applications to software engineering, Software|Practice and Experience 30(11), 1203-1233.

6 The Protégé Ontology Editor and Knowledge Acquisition System.
http://protege.stanford.edu/

7 OntoViz Tab: Visualizing Protégé Ontologies.
http://protege.stanford.edu/plugins/ontoviz/ontoviz.html.

8 Jambalaya.
http://www.thechiselgroup.org/chisel/projects/jambalaya/jambalaya.html.

9 TGVizTab, A TouchGraph Visualization Tab for Protégé 2000.
http://www.ecs.soton.ac.uk/ ha/TGVizTab/TGVizTab.htm.

10 ezOWL: Visual OWL (Web Ontology Language) editor for Protégé.
http://iweb.etri.re.kr/ezowl/index.html.

11 OntoEdit.
http://www.ontoknowledge.org/tools/ontoedit.shtml.

12 IsaViz: A Visual Authoring Tool for RDF.
http://www.w3.org/2001/11/IsaViz/

13 The FRODO RDFSViz Tool.
http://www.dfki.uni-kl.de/frodo/RDFSViz/

14 RDFAuthor.
http://rdfweb.org/people/damian/RDFAuthor/

15 OntoRama.
http://www.ontorama.com/

16 TouchGraph library
http://touchgraph.sourceforge.net/