The final publication is available at ieeexplore.ieee.org published at: Kriglstein, S.; Wallner, G., "Knoocks - A Visualization Approach for OWL Lite Ontologies," Complex, Intelligent and Software Intensive Systems (CISIS), 2010 International Conference on , vol., no., pp.950,955, 15-18 Feb. 2010

Knoocks - A Visualization Approach for OWL Lite Ontologies

Simone Kriglstein
University of Vienna
Department of Knowledge and Business Engineering
Vienna, Austria
simone.kriglstein@univie.ac.at

Günter Wallner
University of Applied Arts Vienna
Department of Geometry
Vienna, Austria
wallner.guenter@uni-ak.ac.at

***Abstract*—With increasing popularity of ontologies in various communities, visualizations of their content and structure became more and more important. In the past few years a number of visualization approaches were developed with the focus either on the representation of the relationships between classes or on the hierarchical structure and instances. However for several applications, a visualization which combines information about instances, classes and hierarchical as well as non-hierarchical relationships is from interest. In this paper we present Knoocks (Knowledge Blocks), which is a visualization approach with focus on both the interconnections within the ontology and the instances in conjunction with their hierarchical structure.**

***Keywords*-Ontology visualization; OWL; Knoocks;**

I. INTRODUCTION

Ontologies define concepts and the interconnectedness of a domain and can therefore be used as a skeletal foundation for a knowledge base [1]. Because such conceptualizations can contain a large number of classes, instances and properties, it can be very difficult to comprehend the ontology and its dependencies at first glance without any graphical representation. Visualizations allow to explore and browse the structure of ontologies. Therefore, users can build valuable knowledge during their usage of a visualization, which supports them in their decisions, makes things visible or presents things in a new light of which users were not aware of before [2].

A number of visualization tools for ontologies were developed in the past few years. An extensive survey of different approaches and tools is given by Lanzenberger et al. [3] and Katifori et al. [4]. The approaches adapt well-known information visualization techniques, which are used e.g., to represent hierarchical and non-hierarchical structures. For example node-link approaches are most frequently applied to represent *subclass-of* relationships and object properties between the concepts (e.g. TGVizTab [5]). The relationships between two nodes are represented as edges and the layout orientation is mainly top-to-bottom or left-to-right. Node-link representations are an intuitive way to show relationships between nodes and provide a good overview about the structure [4]. Especially for larger graphs, however, it can happen that the graph is overcrowded and links can be too long, which makes it difficult to follow them or to distinguish between the different connections. In addition to the representation of relationships, the assignment of instances to classes is also from interest. Furthermore, Wang and Parsia [6] point out that users are more likely interested in hierarchical information about classes to see if ontologies are suitable for their tasks, especially if they are not familiar with the ontology. For this purpose, container approaches have been developed (e.g. CropCircles [6] or Cluster Maps [7]) which represent instances or subclasses nested inside their parent class. Therefore, users can rapidly jump between classes or instances. However, if the ontology contains many subclasses or instances, it can have a negative effect on the overview, as it is the case with node-link approaches. Furthermore, ontologies are something more than only a hierarchy of concepts with their instances. They may also contain object properties and datatype properties, which are of interest for several applications.

Therefore, it is not a simple task to create a visualization approach that will effectively display all this kind of information. Existing visualizations focus either primarily on the representation of relationships and properties between classes or on the hierarchy structure in combination with instances. Jambalaya [8] is a possible solution, which combines the different strengths of node-link visualizations and container approaches. It provides a collection of different kinds of visualization techniques and thereby users have the possibility to select between different views.

In this paper, we present Knoocks (**Kno**wledge Bl**ocks**), an ontology visualization approach which should support ontology experts (e.g. developers) and non-experts (e.g. students and lecturers in case of curricula ontologies). In contrast to Jambalaya, Knoocks combines different visualization techniques, which are simultaneously visible. Knoocks was designed to improve accessibility of instances and to allow users to explore and grasp the structure and interconnections of OWL Lite ontologies. The focus therefore lies primarily on the representation of the hierarchical structure of classes and their instances in combination with object properties and datatype properties. The combination of a container approach for the hierarchical structure and a node-link approach for object properties allows users to clearly differentiate between hierarchical and non-hierarchical relationships. Additionally, the representation of instances within their classes gives user a better clarity about their distribution.

This paper is structured as follows. In Section 2 several visualization approaches related to Knoocks will be discussed in more detail and previous versions of Knoocks are presented in Section 3. Based on user feedbacks of the previous versions, the modified version of Knoocks is

described in Section 4. Additionally, user feedback to the new version is presented in Section 5. Finally, results and future work will be discussed.

II. Related Work

A number of visualization approaches with focus on OWL ontologies have been described over the years. This section presents a short overview of different visualization approaches and their techniques, which were selected based on their similarity to the design of Knoocks. Therefore, the focus lies primarily on approaches, which show the children nodes within their parents. Such layouts are greatly influenced by Euler diagrams.

Jambalaya, which is a plug-in for Protégé [9], provides different views to visualize ontologies and one of these views represents the ontology as nested graph. A nested graph, according to Storey et al. [10], is defined as a graph, which includes composite nodes that contain other nodes to represent a hierarchical structure. In case of ontologies, the nodes are classes or instances and different colors are used to clearly differentiate between class and instance. Classes can contain instances on the one hand or *subclass-of* relationships between classes on the other hand [11]. Similar to Knoocks, object properties between nodes are displayed as directed edges [12]. To distinguish the different object properties, every object property has its own color.

Another plug-in for Protégé is PromptViz. PromptViz adapted the Treemap layout [13] to show changes of two merged version of ontologies [14]. A Treemap represents nodes as rectangles and every rectangle is subdivided into further rectangles in regard to the nodes' children. PromptViz represents classes as nodes and different colors for the nodes define how the node has changed between the two versions. Furthermore, arcs are used to visualize the movements of classes.

CropCircles [15] is implemented in Java and is available for the ontology editor SWOOP [16]. The layout of CropCircles presents only the class hierarchy of the ontology without instances. Classes are represented as circles and subclasses are nested in their parent class. Four layout strategies exist for the arrangement of the circles within their parent circles depending on the size distribution of the subclasses [6].

Cluster Map [17] is developed by the software company Aduna and is used in several applications (e.g. DOPE [18] or AutoFocus [17]). It visualizes lightweight ontologies and represents classes, their instances and their hierarchical structure. The *subclass-of* relationships between classes are connected by a directed edge [7]. Instances are visualized as spheres and instances of the same class are grouped in clusters. If classes share instances, the overlapping of these instances is represented as own cluster and is similar to Venn diagrams.

III. Previous Versions

The first version of Knoocks, which was described in detail in [19], was rather limited in its functionality. One of the main limitations was that only *subclass-of* relations were visualized. This was a direct consequence of visualizing only one hierarchy block[1]. This was confirmed by the results of a conducted user study along with the desire to allow filtering and searching for specific elements of the ontology. However, participants endorsed the basic concept of presenting *subclass-of* hierarchies. These issues have been addressed in the development of a new version which was evaluated in a usability test with 22 participants. The next section describes the current version which already includes the improvements, suggested by the participants.

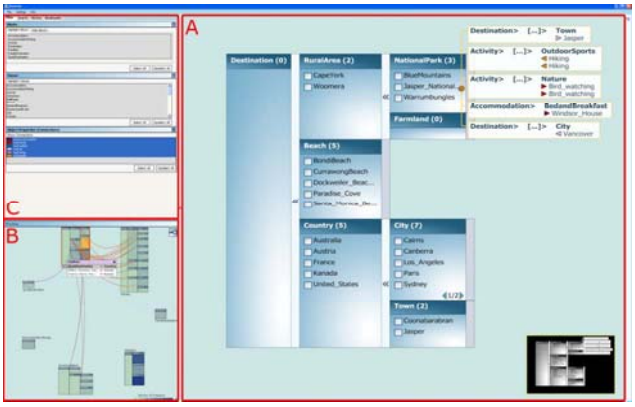

Figure 1. The three main components of Knoocks: main window (A), preview window (B) and toolbox (C). The detail view and overview can be switched between the two windows. The toolbox has functions for searching and filtering, a history and a list of bookmarks.

IV. Design

This section describes the design of our visualization approach. In contrast to the first version described in [19] the new prototype is implemented in C# instead of Java. For parsing OWL Lite Ontologies the publicly available OwlDotNetApi [20] is used whereas OpenGL is utilized for displaying purposes. This allows us to easily use certain features like alpha-blending or texturing. Knoocks provides two views to the user: a detail view and an overview. These two views can be switched between a larger window (intended for interaction with the visualization) and a small preview window at any time. We have opted for this solution, because most existing ontology visualization tools either focus on the details or concentrate on providing a large scale view of the ontology. However, the results of a user study [21] conducted earlier, show that experts and semi-experts expect a good overview about the structure and interconnections along with an easy navigation from the overview to individual instances. Beside the two views a history, bookmarks as well as search and filter functions are available to the user. The main components are depicted in Fig. 1. The detail view can be seen e.g. in Fig. 2 and the overview is shown in Fig. 4.

[1] The concept of a block will be described in Section IV.

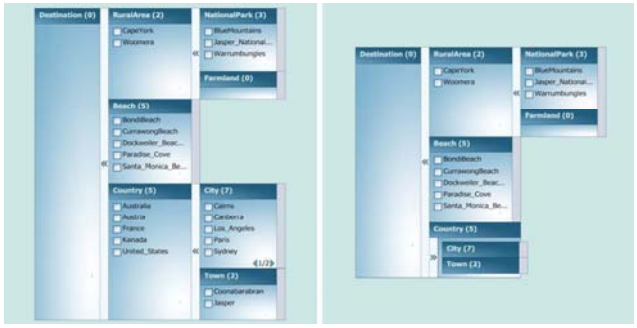

Figure 2. Left: A block showing the complete hierarchy of the class *Destination*. The number of instances of the class *City* exceeds the limit of instances which are shown at once (five in this case) and therefore buttons for paging are available. Right: Subclasses can be closed by clicking on the arrows located to the right of each class which contains subclasses. In this case the class *Country* has been closed.

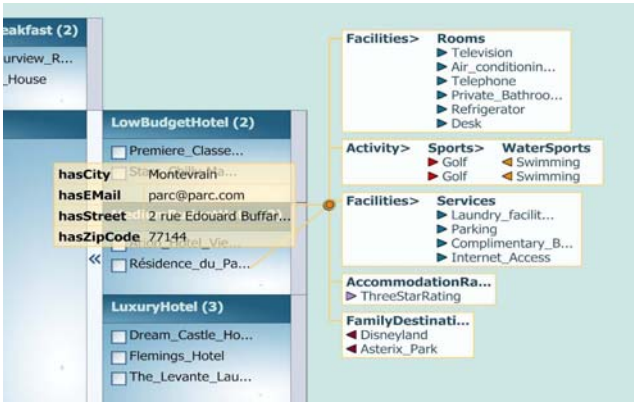

Figure 3. The table of datatype properties (left) and the list of connections for a specific instance (right). Instances in the list are displayed under their respective class. Arrows left to each instance show the direction of the connection and their color reflects the object property (placing the mouse over such an arrow shows directly the name of the object property).

A. *Detail View*

The main entity of Knoocks is a block (a group of classes). For each class, which is directly connected to OWL:Thing, a block is constructed. *Subclass-of* relationships are denoted by placing subclasses to the right side of their predecessor. This structure resembles an Icicle Plot [22], which makes clustering of objects easily noticeable. Furthermore the orientation of Icicle Plots (top-to-bottom or left-to-right) is familiar to users, because it is also used in common node-link approaches, as stated in the work of Barlow and Neville [23].

A class is represented as a rectangle with a header and a body. The header shows the name of the class and the number of instances contained in this class is given in parenthesis behind the class name. The body itself contains a list of the class's instances. This avoids overlapping of names as common with node-link approaches and therefore allows to quickly scan the list of instances. In cases where classes share instances, the instance is shown in each class. If the number of instances in a class exceeds a given threshold n_{max} then paging inside the class is enabled to avoid vertically stretched rectangles in such a case. The height of the rectangle is therefore either given by the total height of all subclasses or by the height needed to enclose n_{max} instances.

The width is defined by the longest name contained in a rectangle, whereas names longer than 15 characters are truncated with "..." in the current version (tooltips will show the complete name of the entity if the mouse is placed over a truncated text) to avoid misshaped rectangles which may otherwise occupy unnecessary space. An arrow to the right side of each rectangle allows closing and opening of all underlying classes. This gives the user the possibility to suppress elements which are not relevant to him. A closed class is only represented with its header. All underlying subclasses are also closed and intended to occupy less display space. Fig. 2 shows the difference between a completely open and a partially closed block. Because the size of a block may exceed the space of a single screen, navigation is possible via a thumbnail in the lower right corner.

OWL Lite defines three types of non-hierarchical relationships: datatype, object and annotation properties. Knoocks – in its current version – supports the former two. Annotation properties were classified as not very important in a user study which was conducted earlier [21]. Datatype properties are represented in a two-column table. The first column contains the names of the properties and the second column lists the corresponding values. The table pops up if the user clicks onto an instance. If an instance is connected with other instances via object properties, then these relations are shown in a collection of lists. Fig. 3 depicts the datatype property table and the object property lists. Instances are grouped according to common class paths. In particular, all instances which can be reached along the same path are shown within a single rectangle. This grouping has the advantage that users can easily notice which instances have the same subclass hierarchy in common. The path begins with the root node of the corresponding block so that the user can quickly infer this block. If classes along this path do not contain any linked instances, then these classes are abbreviated with "[...]" and tooltips show the complete omitted path. This way non-relevant classes do not occupy much space and the path can nonetheless be obtained. All entries in the collection are links, in a way that they allow to quickly jump to the respective entry on which the detail view is automatically focused. Users can move the collection as a whole or each rectangle individually. This gives the users complete freedom to place the elements wherever they like. Tables and collections have a slightly transparent background, so that they do not cover the background completely and therefore the user has still a faint impression of what is obstructed.

B. *Overview*

Blocks in the overview are laid out in a radial pattern as can be seen in Fig. 4. The placement of blocks is calculated by a simple genetic algorithm which minimizes edge

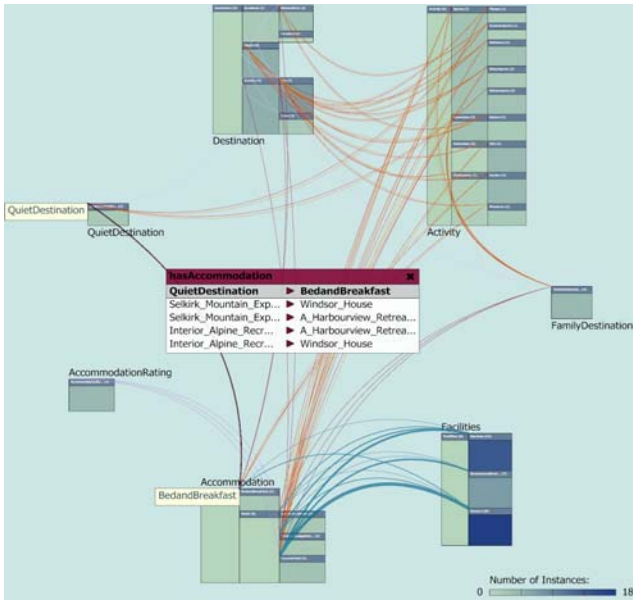

Figure 4. Overview with all blocks and their interconnections. Color and width of the curves (meta-edges) reflect the object property and the number of contained edges. If the mouse moves over a particular curve, this curve is highlighted with a black outline and a table shows all of the contained connections. Tooltips at both ends of the curve depict the names of the two connected classes. The color of the individual rectangles reflects the number of contained instances.

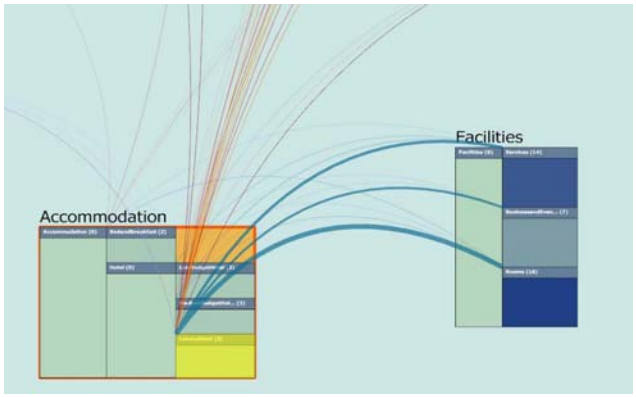

Figure 5. If a specific class is selected than this class is highlighted (yellow) and only curves connected with this class are shown in full saturation, the others are faded out.

crossings. The size of blocks is chosen in such a way that no overlapping occurs. However, individual instances are not shown in the overview because due to space restrictions they would not be readable and therefore only clutter the visualization. Instead, the background color of a rectangle reflects the number of instances.

Connections via object properties are displayed as cubic Bezier curves and each object property has an individual color assigned to make differentiation easy. The colors were chosen in such a way that they are harmonious and the contrast is high. We decided to use curved lines, because if more than two nodes lie on the same straight line, connecting these with straight line segments can lead to ambiguities since the segments will fully coincide in such a case. In connection with the radial layout and because these lines are attracted towards the center the probability that lines cross over other blocks is reduced.

To avoid visual clutter, we use an edge bundling technique similar to the one described by Holton [24]. However, Holton bundled the edges visually be drawing edges connected to the same two nodes close together. In our case it is primarily a bundling of edges into one meta-edge. All edges between instances of the same two classes with the same object property are therefore merged into one such meta-edge whereas the thickness of the curve depends on the number of these edges. Since multiple meta-edges can connect to the same two nodes the middle control point of the cubic curve is altered to slightly offset them from each other.

The curves are drawn with alpha-blending to avoid obscuration of other curves and blocks. Moving the mouse over such a curve highlights it and shows a table with all contained edges. This connection table can also be pinned down in which case the curve remains highlighted. The connection table shows the type of the object property, the two linked classes and the individual instances which are connected. Arrows between two instances show the direction of the relation (see Fig. 4). The color of the table header and of the arrows resembles the object property. All entries in the connection table are clickable, meaning that one can immediately focus the detail view on the respective entry. This automatically brings the detail view into the large window, because users noted in a previous evaluation that doing it manually each time is bothersome.

Users can also move individual blocks if they are not satisfied with the automatic alignment to arrange/group them in any manner they like. Edges and pinned down connection tables are moved accordingly, meaning that the relative position of a connection table along the Bezier curve is preserved. The overview window also allows the user to select a specific block by simply clicking on its enclosing box. In contrast, clicking somewhere on the background will deselect all blocks. If a block is selected, only edges which are connected to this block are shown. Furthermore single classes can be selected or deselected respectively by clicking on the respective rectangle. If a class is selected then only edges which are connected to it are shown in full saturation. All other edges linked to the block in which the class resides are drawn with higher translucency. This way, currently important curves stick out from the environment and are therefore easily noticeable (see Fig. 5 for an example). Data properties are not shown in the overview window.

C. *Filtering and Searching*

One of the main drawbacks of the previous versions were missing filter and search function, which are implemented in the current version.

Filtering allows the user to hide and highlight certain elements of the ontology. Blocks can be highlighted or hidden completely in which case all relations connected with this block are also hidden. Individual classes of a block can also be highlighted, whereas such classes have a higher

priority over highlighted blocks. Furthermore edges with certain object properties can be hidden in the overview.

Searching on the other hand allows the user to find specific instances or classes of the ontology. A quick search function permits searching by name whereas an advanced search function gives the possibility to search by data properties. The logical functions, which can be applied to the value of the property, depend on its datatype. Multiple queries can be combined with logical AND and OR operations. In either case the results are displayed in a listbox where the user can click on the individual results to focus the detail view on the respective class or instance.

D. Bookmarks and History

A checkbox left to each instance allows users to bookmark instances which are of interest to them. All of these bookmarks are shown, alphabetically ordered, in a listbox. Clicking on an entry in this list enables users to focus the detail view on the respective instance. Furthermore, a history is available which records every jump from one instance/class to another instance/class. This way, users can track their progression through the ontology and – if necessary – go back to a previous entry in the list.

V. User Feedback

The presented version was evaluated by three ontology developers in regard to usability and functionality. Although this is a rather small number of participants and therefore no significant quantitative measurements can be derived, we received valuable qualitative feedback. Testing sessions for each participant took about 180 minutes and consisted of: task scenarios, observations in combination with thinking aloud protocols, semi-structured interviews and comparisons with Jambalaya as well as TGVizTab. One set of tasks focused on the identification of specific instances, datatype properties or classes. Another set of tasks concentrated mainly on the dependencies between instances and between blocks. To obtain comparable results we used the same curriculum ontology as in previous evaluations. This ontology consists of 86 classes, 122 instances, 2 object properties and 8 datatype properties.

A. Results

Responding to questions about users' experiences with ontologies and their expectations of ontology visualizations, the participants stated that their ontologies usually contain less than 50 classes and include more instances than classes. This confirms our approach to improve the accessibility of instances and to provide a clear representation of the connection with their respective classes. As current used visualization tools, they use Jambalaya and Gruff. From a visualization they expect: a clear representation of classes and their relationships, simple access to instances, the ability to represent a large number of instances and filtering.

In general, Knoocks got predominantly positive reactions and the following strengths were named: visibility of siblings of parent, flexibility (e.g. moving tables and blocks), clear differentiation between non-hierarchical and hierarchical relationships, and representation of all datatype properties at first glance as well as visualization of instances in connection with their classes. All test persons realized that the width of curves reflects the number of edges and one expert described it as a useful feature. Furthermore, the layout of blocks and closing and opening of classes (which helped them to understand the structure of blocks) were clear. In the overview window, double click to zoom in a block was intuitive and for all three experts it was clear that orange tables show the datatype properties and that lists show the connections. Arrow symbols and colors of object properties were also understandable. For tasks, which required identification of instances or classes, all experts used the search function. Highlighting of instances or classes helped them to easily locate search results in the detail view. We observed that highlighting was also very useful to focus the users' attention after jumping to other instances or classes. Furthermore, navigation via the thumbnail was clear, although they stated that it was a bit unusual at the beginning. The responses to questions about the design of the graphical representation showed that they found colors and design attractive and well balanced.

However, two participants noted that they missed information about the types (e.g. integer, string) of datatype properties in tables. Two experts would also like to get general information about a class, in particular information about contained datatype and object properties. They suggested to show this information after clicking on the class header. Closing of tables and connection lists in the detail view was not clear enough, because they initially overlap their associated instances. Therefore, they stated that it would be helpful to have a button to close all tables in the detail view at once and another one for closing all connection tables in the overview window. Furthermore, the meaning of bookmarks in connection with instances was not clear in the beginning. After explanation, however, they saw the benefit of them. One expert noted that it was sometimes annoying that texts were truncated. Because in the test ontology, several instance names start with the same letters and therefore 15 characters were often not enough. Therefore, using multiple rows in case of datatype properties was proposed as possible solution. Furthermore, it was named as useful to have the possibility to export the current visualization, which is represented in the main window, as an image. Further suggestions for possible improvements were: user definable colors for object properties, hidden blocks should be marked in search results and connection tables in the overview window should also close after clicking on the corresponding highlighted edge.

All participants confirmed that the graphical representation met their expectations, although one expert noted as possible weakness that Knoocks is not a plug-in for Protégé.

In comparison with Jambalaya and TGVizTab, participants stated that TGVizTab makes it difficult to distinguish between hierarchical and non-hierarchical relationships because of missing visual differentiation. In contrast to TGVizTab, they liked the design of Jambalaya and stated that Knoocks and Jambalaya are similar in regard to functions and handling. However in contrast to Knoocks,

they found that Jambalaya quickly overcrowds with increasing number of represented relationships. Furthermore, they noted that the layout of Knoocks is better for getting a fast overview of the general structure of an ontology, which they find rather difficult in case of a nested graph layout.

VI. CONCLUSION

This paper described the design concepts behind the ontology visualization approach Knoocks. Contrary to most of the existing approaches which either focus on the hierarchical structure with instances or on interconnections within the ontology, Knoocks aims to merge both approaches by providing two views: an overview and a detail view. These two views are visible at the same time and are linked with each other, allowing the user to seamlessly go from detail view to overview and vice versa. First reactions of ontology developers were favorable and confirmed the basic concept. For the next version we will try to resolve the usability issues which were revealed during the evaluation. Furthermore we will verify how missing features (e.g. display of types and general class information) addressed by the experts can be included into the current concept. Further usability evaluations will be carried out in the near future to confirm the underlying concepts of our approach.

REFERENCES

[1] B. Swartout, R. Patil, K. Knight, and T. Russ, "Toward Distributed Use of Large-Scale Ontologies," Proc. AAAI97 Spring Symposium Series, Workshop on Ontological Engineering, 1997, pp. 138 - 148.

[2] J. J. v. Wijk, "The Value of Visualization," Proc. IEEE Visualization, 2005, pp. 79 - 86.

[3] M. Lanzenberger, J. Sampson, and M. Rester, "Visualization in Ontology Tools," Proc. International Conference on Complex, Intelligent and Software Intensive Systems. 2nd International Workshop on Ontology Alignment and Visualization, Fukuoka, Japan, 2009

[4] A. Katifori, G. Halatsis, G. Lepouras, C. Vassilakis, and E. Giannopoulou, "Ontology visualization methods—a survey," ACM Computing Surveys, vol. 39, 2007.

[5] H. Alani, "TGVizTab: An Ontology Visualisation Extension for Protégé," Proc. Knowledge Capture (K-Cap'03). Workshop on Visualization Information in Knowledge Engineering, Sanibel Island, Florida, USA, 2003

[6] T. D. Wang and B. Parsia, "CropCircles: Topology Sensitive Visualization of OWL Class Hierarchies," Proc. 5th International Semantic Web Conference (ISWC '06), Athens, USA, 2006

[7] C. Fluit, M. Sabou, and F. v. Harmelen, "Supporting user tasks through visualization of light-weight ontologies," in Handbook on Ontologies, S. Staab and R. Studer, Eds.: Springer, 2004, pp. 415 - 434.

[8] R. Lintern and M.-A. Storey, "Jambalaya express: on demand knowledge visualization," in 8th Protégé Conference. Madrid, Spain, 2005.

[9] Stanford Center for Biomedical Informatics Research, "Protégé project," online: http://protege.stanford.edu Last seen:29.09.2009.

[10] M. A. D. Storey, K. Wong, F. D. Fracchia, and H. A. Mueller, "On Integrating Visualization Techniques for Effective Software Exploration," in Proceedings of the IEEE Symposium on Information Visualization (InfoVis '97): IEEE Computer Society Press, 1997.

[11] 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 Protege," Proc. Intl. Workshop on Interactive Tools for Knowledge Capture, Victoria, B.C. Canada, 2001

[12] M. A. Storey, R. Lintern, N. Ernst, and D. Perrin, "Visualization and Protégé," in the 7th International Protégé Conference. Bethesda, Maryland, 2004.

[13] B. Shneiderman, "Tree visualization with tree-maps: 2-d space-filling approach," ACM Transactions on Graphics, vol. 11, 1992, pp. 92-99.

[14] D. Perrin, "Prompt-viz: Ontology version comparison visualizations with treemaps," in Department of Computer Science,. Victoria, Canada: University of Victoria, 2004.

[15] B. Parsia, T. Wang, and J. Golbeck, "Visualizing Web ontologies with CropCircles," Proc. 4th International Semantic Web Conference (ISWC '05). Workshop End User Semantic Web Interaction, Galway, Ireland, 2005

[16] A. Kalyanpur, B. Parsia, E. Sirin, B. Cuenca-Grau, and J. Hendler, "Swoop: A 'Web' Ontology Editing Browser," Journal of Web Semantics, vol. 4, 2005.

[17] C. Fluit, M. Sabou, and F. v. Harmelen, "Ontology-based information visualisation: Towards semantic web applications," in Visualising the Semantic Web (2nd edition), V. Geroimenko, Ed.: Springer, 2005.

[18] H. Stuckenschmidt, A. d. Waard, R. Bhogal, C. Fluit, A. Kampman, J. v. Buel, E. v. Mulligen, J. Broekstra, I. Crowlesmith, F. v. Harmelen, and T. Scerri, "A Topic-Based Browser for Large online Resources," Proc. International Conference on Knowledge Acquisition, Modelling and Management (EKAW'04), Milton Keynes, England, 2004

[19] S. Kriglstein and R. Motschnig-Pitrik, "Knoocks: New Visualization Approach for Ontologies," Proc. Information Visualization (IV '08), London, UK, 2008, pp. 163 - 168.

[20] B. Pellens, "OwlDotNetApi," online:http://users.skynet.be/bpellens/OwlDotNetApi/index.html Last seen:26.09.2009.

[21] S. Kriglstein, "User Requirements Analysis on Ontology Visualization," Proc. International Conference on Complex, Intelligent and Software Intensive Systems. 2nd International Workshop on Ontology Alignment and Visualization, Fukuoka, Japan, 2009

[22] J. B. Kruskal and J. M. Landwehr, "Icicle Plots: Better Displays for Hierarchical Clustering," The American Statistician, vol. 37, 1983, pp. 162 - 168.

[23] T. Barlow and P. Neville, "A Comparison of 2-D Visualizations of Hierarchies," Proc. IEEE Symposium on Information Visualization 2001 (INFOVIS'01), 2001

[24] D. Holten, "Hierarchical Edge Bundles: Visualization of Adjacency Relations in Hierarchical Data," IEEE Transactions on Visualization and Computer Graphics, vol. 12, 2006, pp. 741 - 748.