The final publication is available at ieeexplore.ieee.org published at: Kriglstein, S., "OWL Ontology Visualization: Graphical Representations of Properties on the Instance Level," Information Visualisation (IV), 2010 14th International Conference , vol., no., pp.92,97, 26-29 July 2010

OWL Ontology Visualization: Graphical Representations of Properties on the Instance Level

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

Abstract

For several ontology applications, a combination of classes with their instances, their properties on the class level and on the instance level are from interest. However, the focus of most visualization approaches is on the hierarchical and non-hierarchical relationships on the class level. This paper presents an approach to visualizing datatype properties and object properties on the instance level. For this purpose, three different layouts were developed for the ontology visualization tool Knoocks. Furthermore, the paper discusses results of an evaluation that was motivated to identify which one of these layout versions was preferred by the users. The evaluation should also reveal if the concept of the representation of the properties was understandable for them.

Keywords---**Information visualization, OWL Lite, Knoocks, graphical representation of instances, datatype properties, object properties**

1. Introduction

An ontology is a conceptualization of a specific domain and can be used as a skeletal foundation for the knowledge base to provide a common understanding of a specific domain [1, 2]. For example, in the context of education (see e.g., [3, 4]) ontologies present curricular structures to support students for planning their study or to help lecturers to organize their courses.

OWL is a popular language to describe ontologies in a human-readable way and allows users to communicate, to analyze, to exchange or to share ontologies. The basic elements of OWL ontologies are [5]:

- ***Classes:*** present the relevant concepts.
- ***Instances:*** stand for the individual objects of the class.
- ***Object properties:*** define the interconnections within the ontology and relate instances to instances.
- ***Datatype properties:*** relate instances to datatypes which allow to specify additional information about the instances.

Based on the fact that ontologies can become very large and complex, graphical representations can support humans to work with them more easily. Therefore, several visualization tools for ontologies were developed in the past few years. A good overview of different visualization tools and approaches is given in [6, 7].

Most visualization approaches concentrate mainly on classes. Therefore, object properties are often visualized – if at all – only on the class level and not directly on the instance level. For several applications, however, instances and the representation of their properties play an important role [8]. For example, in case of a curriculum ontology, courses can be defined as instances, whereas ECTS points of courses may be stored as datatype properties. Content dependencies between the courses can be described as object properties. Or in case of a travel ontology to find hotels, hotel names can be defined as instances and the contact information of the hotels can be defined as datatype properties.

These examples show that it is often not sufficient to visualize only an overview about the structure of the ontology on the class level. It is also necessary that the graphical representations of ontologies allow users to effectively see the instances with their datatype properties and their interconnections to other instances.

In this paper, we present an approach which allows users to directly see datatype properties of instances and object properties on instance level. The object properties representation includes hyperlinks between instances which allow users to jump rapidly between instances. Therefore, fast access to other instances is possible and this allows to explore the interconnections between instances without leaving the detail view. For this purpose, we developed three different layout versions which are integrated in the ontology visualization tool Knoocks [9]. This tool allows users to explore and analyze OWL Lite ontologies. In order to help users to understand the structure of classes, instances and properties, an overview of classes with their relationships is combined with a detail view to represent instances in connection with their classes and their properties.

This paper is structured as follows. Section 2 gives a short overview how other visualization approaches present properties on the instance level. A short description of Knoocks is given in Section 3 and in Section 4 the three layout versions for the representation of properties are shown. The evaluation of the three

layout versions is described in Section 5. Finally, results and future work will be discussed.

2. Related work

In general, there exist two strategies to represent properties on the instance level. One strategy shows the properties in a separate window (e.g. Grokker [10], GOSurfer [11] and TGVizTab [12]) and the other strategy presents the properties directly at the class or instance node (e.g. Jambalaya [13], OntoViz [14] and Ozone [15]).

For the representation of datatype properties, tables are often used. For example, OntoViz and Ozone allow users to choose datatype properties for instances and classes which are visualized as tables in the graph. In case of OntoViz, sizes of tables are constant and therefore if the values of datatype properties are too long, texts are truncated with "..." without the possibility to see the full values. In contrast to OntoViz and Ozone, TGVizTab shows the datatype properties of instances in a separate window after double clicking on the instance. In case of Jambalaya, the datatype properties are embedded as own window after users zoomed in the instance level. The separate window for properties in case of Jambalaya and TGVizTab, presents every datatype property as own table.

Object properties are mainly visualized as node-link representations. For example, OntoViz shows the object properties as labeled edges between classes and instances. However, OntoViz makes no difference between different object properties. Therefore it is difficult to distinguish between the different relationships. Furthermore, if the connections are too long, it is tedious to follow them [4]. Jambalaya provides different views to visualize ontologies. The nested graph view is one of these views and shows the instances embedded in their classes and the object properties as edges between the classes. The different object properties have different colors and the labels of the connections are also visible as a tooltip. The object properties are only visualized as edges on the class level and therefore users have to zoom in the instance level to see an embedded window that represents for every object property a list of instances in regard to their outgoing direction. However, it is not possible to directly see the instances from the incoming direction. For example, in case of a curriculum ontology it is not possible to see which courses are the prerequisites of the selected course.

3. Knoocks

Knoocks (**Kno**wledge Bl**ocks**) is a visualization tool for OWL Lite ontologies and is implemented in C#. For parsing OWL Lite ontologies, the publicly available OwlDotNetApi [16] is applied and OpenGL is used for displaying purposes (e.g., for alpha-blending or texturing). For the development of Knoocks, ontology visualization requirements (based on literature study, e.g., [8, 17]), usability and interface design aspects (based on [18, 19, 20]) were considered.

Knoocks divide the ontology in blocks. A block is generated for each class which is directly connected to OWL:Thing and the blocks also contains of the class' descendents. Classes are represented as rectangles. The *subclassOf* relationships are presented in such a way that the right class is the subclass of the left class. The layout of blocks is inspired by the Icicle Plot concept [21] to make clustering of objects easily noticeable. Classes contain their instances as a list, which is motivated by space filling approaches, in particular of the treemap [22]. The list representation allows user to quickly scan the instances and automatically avoids overlapping of instances.

Furthermore, Knoocks provides two views: a detail view and an overview. The overview shows the blocks and object properties are presented as edges between the classes. Every object property has its own color to make them better distinguishable from each other. The detail view presents the instances in combination with their object properties and datatype properties. Users can switch between the views via a switch button. Figure 1 shows both views. The latest version of Knoocks - as of this writing - is described in more detail in [9].

4. Layout Design for Properties

Whereas, object properties are presented as edges on the class level to give a general overview about the connections, the properties on instance level give more precise information about the instances themselves. Properties of the instances pop up if the user clicks on the label of the instance in the detail view and properties are closed after the user clicks the label of the instance again. The properties have a slightly transparent background to allow users to see the properties in connection with the selected instance in the detail view without losing the orientation.

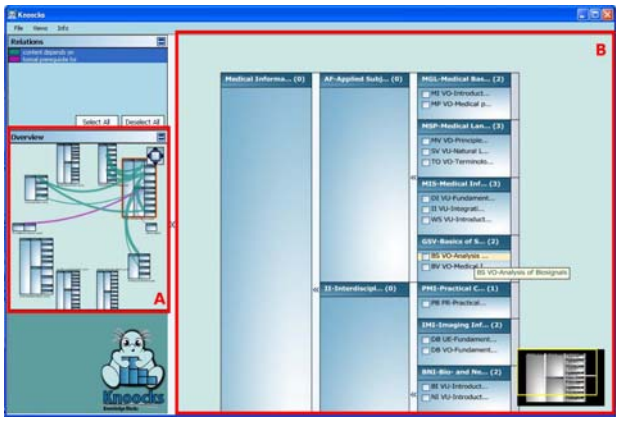

Figure 1: Main window of Knoocks shows the detail view (B) and the preview window presents the overview (A).

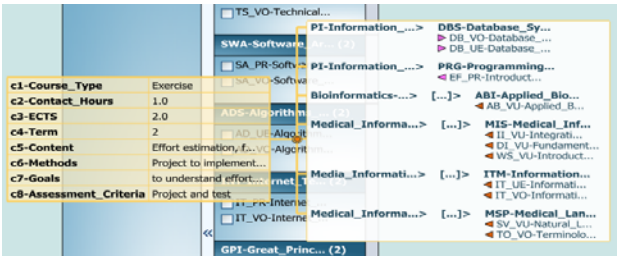

Figure 2: Layout version A for properties on the instance level which shows the object properties in a single white box.

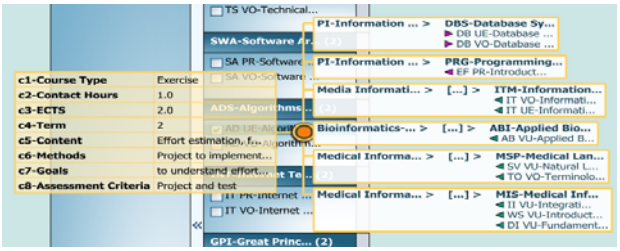

Figure 3: Layout version B for properties on the instance level. For the object properties, every class path is represented as own rectangle.

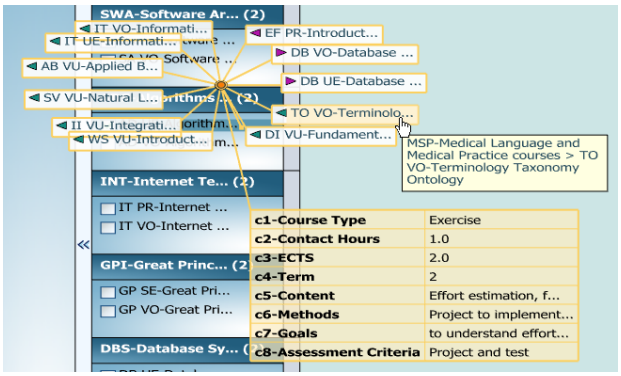

Figure 4: Layout version C for properties on the instance level. A circular layout is used for the representation of the object properties.

The datatype properties are listed in an orange two-column table. The first column contains the names of the datatype properties and the second column presents the corresponding value. For the representation of object properties, three different layouts were developed.

The first version (version A) shows the object properties as a list (see Figure 2). Names of classes are presented as path and instances are grouped in regard to their common class paths. This grouping allows users to quickly see which instances are members of the same subclass. The inspiration of this path representation was the well-known breadcrumb navigation. Similar to breadcrumb navigation, the class path specifies the hierarchy separator as greater than (>) symbol. Instances are presented directly under the name of their class. Clicking on the name of an instance allows users to jump directly to the selected instances in the detail view. The arrow symbol beside the instance label shows the direction of the connection and the color of the arrow corresponds to the type of the object property. For space saving, classes are abbreviated with "[...]", if they do not contain any instances. The complete omitted path is visible via tooltip. Furthermore, users can drag and drop the table of datatype properties as well as the list of object properties. This gives users more flexibility and allows to compare properties of several instances.

Version B (see Figure 3) is similar to version A, but in contrast to version A, every class path is represented as own rectangle to make the differentiation of the groups clearer. Furthermore, the representation of separate groups allows users to move each rectangle individually.

Layout C (see Figure 4) presents every instance with the arrow symbol for direction and type of object properties in a separate rectangle. The class path information is only visible via tooltip. The instances are arranged in a radial pattern, because the representation of object properties as a list has the drawback to grow in the height for a large number of entries. Furthermore, users can also move the collection of instances as a whole.

5. Evaluation

The goal of the evaluation was to find out more about the usability of the different layout versions. Furthermore, we wanted to identify which one of these layouts was preferred by the participants.

5.1. Test Case Ontology

As test case ontology, a bachelor curriculum of computer science ontology is used for all three layout versions. This ontology describes the modules of the curriculum as classes and the courses are defined as instances. Additional information about the courses (e.g., ECTS points, course type, course contents and course goals) are defined as datatype properties. Object properties show the formal dependencies and dependencies in terms of the content between courses. Summarized, the ontology consists of 86 classes, 122 instances, 2 object properties and 8 datatype properties.

5.2. Participants

The findings of the evaluation are based on 21 participants (14 students of computer science and 7 lecturers of computer science). The reason for this choice of participants was that they were familiar with the curriculum information and thereby the focus was on the visualization itself and not on learning a new ontology.

Each of the three layout versions for the properties representations of the instances was evaluated by seven participants and the testing session for each participant took about 90 minutes.

5.3. Methods

The evaluation is based on task scenarios in combination with observations and thinking aloud

protocols. Before the users started with the tasks, they had the possibility to get a first impression of the visualization and to interact with Knoocks until they had the feeling to have a good overview and understanding of the tool and its functionalities.

The tasks concentrate primarily on the identification of the datatype properties and the object properties between instances (e.g., to find the content dependencies of a specific courses to other courses). To identify strengths and weaknesses of the different layouts, the participants were additionally asked about their impression of the visualization of the properties and how well the design was understandable.

After they finished all tasks, all three design layout versions are shown to each participant. The participants were asked to rate arrangement and understandability of each layout in regard to the representation of the datatype properties and the object properties. For this purpose, a 7-point Likert scales were used from *"not well-arranged"* to *"very well-arranged"* and from *"not understandable"* to *"very understandable"*. Finally, we asked them which one of these layout versions they preferred and which one of the three versions they found less attractive.

5.4. Results

The findings of our observations and thinking aloud protocols showed us that in general the participants had no problems to identify datatype properties of an instance independent from the layout version they used. They said they were *"helpful"* and *"fast to understand"*. In contrast to datatype properties, participants found the representation of the object properties more difficult at the beginning. Based on the fact that they got no introduction of Knoocks, they noted that the direction of the arrows and the meanings of the arrow color were not instantly clear without additional hints (9 statements). To make the directions of the arrows clearer, several participants suggested to group the arrows in regard to their outgoing and incoming directions (6 statements). However, we observed that they learned the meaning of the direction and color very fast. Participants, who tested layout C, saw the orange button in the middle of the circular arrangement as point of reference for the arrow directions.

Participants pointed out that they missed a header with the name of the instance and a close button (5 statements), because if many properties of instances were open it was not clear which representation of properties was for which instance. Especially participants, who tested the version C, found it more difficult to close the properties of an instance than in version A and B.

Several participants disliked that the texts were truncated (12 statements). Although one participant liked the solution with tooltips to see more information, most of the participants would have found it more helpful to have multiple rows for the datatype properties.

	A (%)	B (%)	C (%)
preferred layout version	33.33	**52.38**	14.29
less attractive layout version	23.81	9.52	**66.67**

Table 1. Percent distribution of the rating for the layout versions A, B and C.

Only for three participants was the representation of drag and drop functionality for the properties of the instances not clear enough at the beginning. Four participants explicitly stated that they liked to move the properties of an instance, because it allowed them to arrange the properties how they liked. Furthermore, jumping to another instance directly from the representations of properties, was for the participants also clear. One participant stated explicitly that it is a good solution.

After we showed the participants all three layout versions, most participants found version B very well-arranged (see Figure 5) and understandable (see Figure 6). Especially, the representation of the object properties was very understandable for 50% of the participants. Furthermore, eleven participants preferred layout B (see Table 1). Reasons for their decision were that version B gave them a clear overview (4 statements) and they liked that the object properties from the instance to other instances were organized in regard to their classes (6 statements). Four participants stated as advantage that the layout B allowed them to move every single box with their object properties how they liked. However, one participant found it unnecessary to move every single box and one participant noted that version B looked more overloaded than version A.

The rating of the three layout versions showed that most of the participants found the version C least attractive (see Table 1). Especially the participants rated version C lower in regard to well-arranged representation of the object properties (see Figure 5) and only 14.29% of the participants found the object properties in version C very well-arranged. They disliked that object properties were partially overlapped (3 statements) and they said it was confusing if one instance had many relationships to other instances (6 statements). Furthermore, they noted that they missed an arrangement of the object properties (5 statements) and one participant pointed out that, s/he missed the breadcrumbs. Furthermore, another participant stated that if the object properties were represented in a list, such as in version A or B, it was for him/her clearer than the radial layout. In contrast, three participants preferred version C, because they liked the radial layout and found the layout easier to understand.

Seven participants preferred layout version A. Similar to version B, they found that version A gave a clear overview (4 statements).

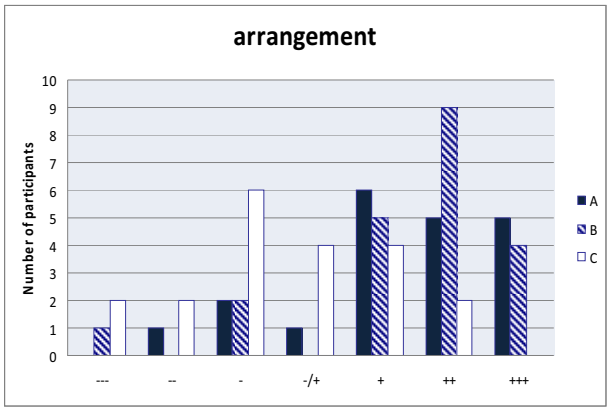

Figure 5. The rating result of layout versions A, B and C in regard to their arrangement of properties.

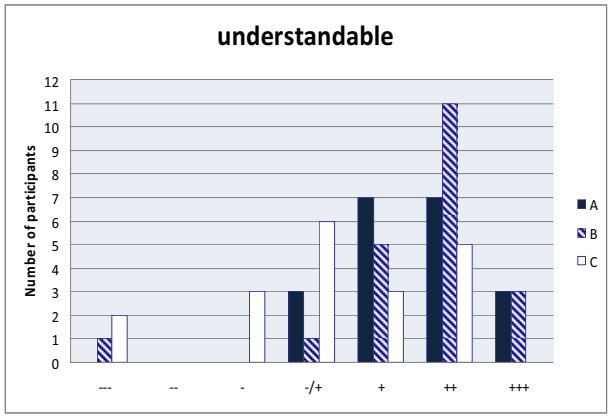

Figure 6. The rating result of layout versions A, B and C in regard to their understandable representation of properties.

They noted as advantage of version A that it needed lesser screen space than version B and therefore it was possible to represent more object properties (4 statement). However, five participants found version A less attractive, because two of them disliked the list representation of the object properties. Only 28.57% of the participants rated the representation of the object properties as very understandable and only 14.29% found the object properties very well-arranged. Named reason was that they missed separations between the groups of object properties and therefore they found the overview less clear than in version B. Furthermore, one participant noted that s/he missed the possibilities to arrange the object properties individually such as in version B.

6. Discussion

The results of the evaluation showed that the concept of the properties representation in the detail view mapped the concept of the participants and therefore they had no problems to find the properties of instances. They found it very intuitive and simple to click on the label of the instance to open the properties. However, closing the properties was more difficult for them, because they searched for a close button. Furthermore, they noted that it would be helpful if the representation of properties would have an additional header with the name of the instance. Because if many properties of various instances were open, they pointed out that the affiliation between instances and their properties was not readily identifiable.

Furthermore, the observations show that the participants preferred the representation of properties in the detail view instead of the node-link representation in the overview. They noted that the connections were clearer than in the overview. Whereas they found that the node-link representation was more suitable for tasks, like getting a fast overview about the distribution of the connections. Based on the fact that the table representation was only for instances, participants noted that it would be beneficial to have this representation also for general information about a class in the detail view (e.g., which datatype properties and object properties the class contains).

Layout version C was more often named as the least attractive layout version of all three, because of the number of overlaps of the object properties' entries. Furthermore, several participants stated that the list representation of the object properties was clearer arranged for them than the radial layout.

Although version B and A were similar, version B was more often named as preferred layout version. Reasons were the additional separations to group the object properties and the possibility to move the collection of properties as a whole or every group individually. The possibility to drag and drop the properties of an instance in all three layout versions allowed the participants to compare different properties of different instances.

The arrow symbols to represent the direction of the connections were for several participants not clear enough at the beginning. They learned the meaning of the direction quickly without additional help, but they were unsure in regard to their decision. Therefore they would have liked additional hints for the meaning of the color via tooltip or to have a help document to control if their assumptions of the direction were correct or not. A possible reason for their uncertainness was that they had lesser experience with properties, because user tests to evaluate Knoocks functionalities with ontology developers (see [9]) showed that they had no problems to understand the color and direction of object properties immediately.

Conclusions

In contrast to other approaches, which show the object properties primarily as node-link representation on the class level, this paper described an approach to represent properties directly on the instance level. For this purpose, three different layout versions were developed which were integrated in the visualization tool Knoocks. The properties representation pops up if the

user clicks on the instance in the detail view of Knoocks. The basic elements of the property representation are tables for the datatype properties and lists for object properties. Furthermore, it allows users to jump to the instances which are presented in the list of object properties without losing the focus on the detail view.

Whereas the visualization of datatype properties is identical for all three versions, the representation of object properties varies in all three versions. To find out if the basic concept of the representation of properties is clear for the users and which one of these layout versions is preferred, we conducted user tests with 21 participants. The result of the evaluation showed that the participants had no problems to find and understand the presentation of properties. However one layout version emerged as preferred layout, because of its list representation of the object properties and its clear separation of the different groups.

One of our next steps is to rework the preferred layout version in regard to the usability issues which were found during the user tests. Further extensive usability evaluation of Knoocks and comparing study with other visualization approaches with focus on the properties on the class level and on the instance level are planned to confirm the underlying concepts of our approach.

Acknowledgements

I gratefully acknowledge the critical feedback and support of Günter Wallner from the University of Applied Arts Vienna.

References

[1] B. Swartout, R. Patil, K. Knight and T. Russ. Toward Distributed Use of Large-Scale Ontologies. In *Proceeding of AAAI97 Spring Symposium Series, Workshop on Ontological Engineering*. AAAI Press. 1997.

[2] D. Fensel. Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce, Berlin, Springer. 2001.

[3] R. Vas. Educational Ontology and Knowledge Testing. In *Electronic Journal of Knowledge Management*. Vol. 5, 123 -130. 2007.

[4] S. Kriglstein. Analysis of Ontology Visualization Techniques for Modular Curricula. In *4th Symposium of the Workgroup Human-Computer Interaction and Usability Engineering of the Austrian Computer Society on HCI and Usability for Education and Work*, A. Holzinger, Ed. Springer, Lecture Notes in Computer Science. 2008.

[5] W3C. OWL Web Ontology Language Guide url:http://www.w3.org/TR/owl-guide/

[6] A. Katifori, G. Halatsis, G. Lepouras, C. Vassilakis and E. Giannopoulou. Ontology visualization methods—a survey. In *ACM Computing Surveys*. Vol. 39. 2007.

[7] M. Lanzenberger, J. Sampson and M. Rester. Visualization in Ontology Tools. In *Proceeding of International Conference on Complex, Intelligent and Software Intensive Systems. 2nd International Workshop on Ontology Alignment and Visualization*. IEEE Computer Society. 2009.

[8] S. Kriglstein. User Requirements Analysis on Ontology Visualization. In *Proceeding of International Conference on Complex, Intelligent and Software Intensive Systems. 2nd International Workshop on Ontology Alignment and Visualization*. IEEE Computer Society. 2009.

[9] S. Kriglstein and G. Wallner. Knoocks - A Visualization Approach for OWL Lite Ontologies. In *Proceeding of International Conference on Complex, Intelligent and Software Intensive Systems. 3rd International Workshop on Ontology Alignment and Visualization*. IEEE Computer Society. 2010.

[10] W. Rivadeneira and B. B. Bederson. A Study of Search Result Clustering Interfaces: Comparing. Textual and Zoomable User Interfaces. University of Maryland HCIL. 2003.

[11] S. Zhong, K. F. Storch, O. Lipan, M. C. Kao, C. J. Weitz and W. H. Wong. GoSurfer: a graphical interactive tool for comparative analysis of large gene sets in Gene Ontology space. In *Appl Bioinformatics*. Vol. 3. 2004.

[12] H. Alani. TGVizTab: An Ontology Visualisation Extension for Protégé. In *Proceeding of Knowledge Capture (K-Cap'03). Workshop on Visualization Information in Knowledge Engineering*. 2003.

[13] 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. In *Proceeding of Intl. Workshop on Interactive Tools for Knowledge Capture*. 2001.

[14] P. Mutton and J. Golbeck. Visualization of Semantic Metadata and Ontologies. In *Proceeding of Information Visualization (IV '03)*. IEEE Computer Society. 2003.

[15] S. Bongwon and B. B. Benjamin. OZONE: a zoomable interface for navigating ontology information. In *Proceedings of the Working Conference on Advanced Visual Interfaces*. ACM. 2002.

[16] B. Pellens. OwlDotNetApi url:http://users.skynet.be/bpellens/OwlDotNetApi/index.html

[17] N. F. Noy and D. L. McGuinness. Ontology Development 101: A Guide to Creating Your First Ontology. In *Stanford Knowledge Systems Laboratory Technical Report and Stanford Medical Informatics Technical Report*. 2001.

[18] International Organization for Standardization. ISO 9241-110:2006 Ergonomics of human-system interaction - Part 110: Dialogue principles. 2006.

[19] J. Nielsen. Usability Engineering, San Francisco, Morgan Kaufmann. 1994.

[20] B. Shneiderman. Designing the User Interface. 3rd ed., Addison Wesley. 1998.

[21] J. B. Kruskal and J. M. Landwehr. Icicle Plots: Better Displays for Hierarchical Clustering. In *The American Statistician*. Vol. 37, 162 - 168. 1983.

[22] B. Shneiderman. Tree visualization with tree-maps: 2-d space-filling approach. In *ACM Transactions on Graphics*. Vol. 11, 92-99. 1992.