Roadmapping and Navigating in the Ontology Visualization Landscape

Marek Dudáš, Ondřej Zamazal, and Vojtěch Svátek

Department of Information and Knowledge Engineering,
University of Economics, W. Churchill Sq.4, 130 67 Prague 3, Czech Republic
{marek.dudas,ondrej.zamazal,svatek}@vse.cz

Abstract. Proper visualization is essential for ontology development, sharing and usage; various use cases however pose specific requirements on visualization features. We analyzed several visualization tools from the perspective of use case categories as well as low-level functional features and OWL expressiveness. A rule-based recommender was subsequently developed to help the user choose a suitable visualizer. Both the analysis results and the recommender were evaluated via a questionnaire.

1 Introduction

Numerous visualization methods have been proposed for OWL ontologies and many software tools implementing them appeared in the past decade, ranging from semantic-web-specific approaches to proposals of UML[1] profiles [4,14,16] leveraging on the similarity with software engineering models. However, so far no visualization method has been accepted by the majority of the semantic web community as de facto standard. One reason clearly is that different use cases require different ontology visualization methods.

The goal of this paper is, first, to *survey* existing software tools from the use case perspective ('roadmapping' aspect of the paper), and, second, to build a prototype *ontology visualization tools recommender* based on our insights ('navigation' aspect of the paper). We believe that such a recommendation tool could find a broad audience of ontology/vocabulary users within the semantic web and linked data realms.

Section 2 provides a classification of ontology visualization use cases based on existing literature. Section 3 surveys existing visualization tools and evaluates them with respect to use case categories. Section 4 briefly presents our recommender, both structurally and in use. Section 5 overviews the outcomes of a questionnaire survey over both the findings from our analysis and the recommender. Section 6 summarizes related work. Finally, Section 7 wraps up the paper with conclusions and future work.

[1] http://www.omg.org/spec/UML/2.4.1/

K. Janowicz et al. (Eds.): EKAW 2014, LNAI 8876, pp. 137–152, 2014.

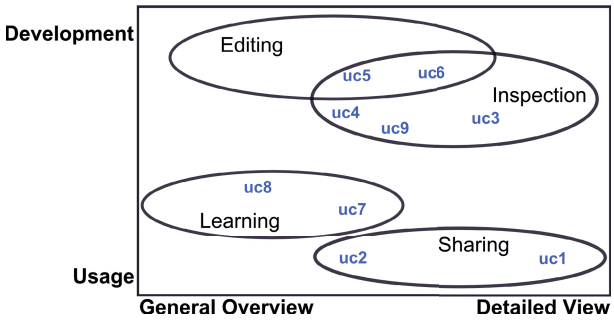


Fig. 1. Ontology visualization use case categories

2 Visualization Use Case Categories and Tool Features

Based on ad hoc analyses of literature and own experience, we collected nine possible *use cases* of ontology visualization: making screenshots of selected parts of an ontology *(uc1)* or of its overall structure *(uc2)*; structural error detection *(uc3)*; checking the model adequacy (how well the ontology covers its domain) *(uc4)*; building a new ontology *(uc5)*; adapting an existing ontology, e.g., adding entities or transforming the style, to fit specific usage *(uc6)*; analyzing an ontology in order to annotate data with (or create instances of) its entities *(uc7)*; deciding about the ontology suitability for a specific use case *(uc8)*; analyzing an ontology in view of mapping its entities to those from another ontology *(uc9)*.

We then aggregated the use cases to four *categories*, named *Editing*, *Inspection*, *Learning* and *Sharing*, positioned in a 2-dimensional space. The first dimension distinguishes whether the user actively 'develops' the ontology or just 'uses' it. The second dimension is the required level of detail, i.e. whether an overview (e.g., class hierarchy) is enough or a detailed view (e.g., axioms related to a class) is needed. The categorization, incl. use cases,[2] is shown in Fig. 1. *Editing* is a 'development' category that needs both a general and detailed view, while *Inspection* focuses on a detailed view. Within visualization for 'usage', *Learning* tends to use the general overview and *Sharing* tends to use the detailed view. The categories are not completely disjoint: *Inspection* overlaps with *Editing*, since the user usually needs to inspect the impact of her/his edits.

Inspired by categorization of tasks that should be supported by an information visualization application as defined in [19] (and adapted in [13]) and evaluation criteria classes described in [7], we further identified seventeen relevant *features* implemented in ontology visualization tools:

F1. *Zoom-Out Overview*: zooming out to get a summary view of the ontology.
F2. *Radar View*: displaying a small 'minimap' of the displayed ontology.
F3. *Graphical Zoom*: enlarging the displayed graphical elements.

[2] The positions of use cases are merely w.r.t. their categories and not in terms of quantitative coordinates. E.g., *uc3* may not need a more detailed view than *uc4*.

F4. *Focus on Selected Entity*: centering the view on a selected entity and its surroundings and hiding other parts of the ontology.
F5. *History*: keeping the history of navigation steps performed by the user, thus allowing for undo/redo actions.
F6. *Pop-Up Window*: displaying details on a chosen entity in a separate window.
F7. *Incremental Exploration*: starting by a small part of the ontology and gradually expanding the nodes selected by the user (as detailed in [9]).
F8. *Search*: text-based search leading to highlighting the matched entities.
F9. *Hide Selected Entity*: hiding parts of the ontology the user is currently not interested in, thus avoiding a cluttered view.
F10. *Filter Specific Entity Type*: e.g., hiding all object properties at once.
F11. *Fisheye Distortion*: zooming in for a part of the graph and zooming out for the rest; focuses on a detail but keeps the context, see, e.g., [18].
F12. *Edge Bundles*: grouping edges with similar paths, thus alleviating the clutter; implemented, e.g., in GLOW [10].
F13. *Drag&Drop Navigation*: moving a graph that is bigger than the screen around by dragging it with the mouse.
F14. *Drag&Drop User Layout*: allowing to move the individual nodes around.
F15. *Clustering*: 'intelligent' grouping of nodes or displaying a subset of 'important' nodes, as in KC-Viz [15].
F16. *Integration with Editing*: the user can select a visualized node or edge and edit its properties in, e.g., a pop-up window.
F17. *Graphical Editing*: the tool supports creating new entities by, e.g., drawing edges between the displayed nodes.

Referring to [7], F1–F5 are associated with a criteria class called *Help and User Orientation*, F6–F8 with *Navigation and Browsing*, F9–F10 with *Dataset Reduction*, F11–F14 with *Spatial Organization*, and F15 with *Information Coding*. F16 and F17 have no corresponding class in [7]. In each use case category most of the 17 features should ideally be supported; however, their importance varies. The alignment between the categories and tool features is, mostly, intuitive:

Editing ranges from developing a new ontology to merely changing a property value. Usually, the purpose of visualization is to find the entity to be changed or to become parent of a new entity. Important features are *Pop-Up Window* (to see detailed properties of a selected entity), *Search* (to easily find entities to be edited) and obviously *Integration with Editing* and *Graphical Editing*.
Inspection needs a detailed view of the ontology to see errors or deficiencies. It is often implied by *Editing*, as the user needs to see the state of the ontology before and after an edit. *Pop-Up Window* and *Search* are thus important here as well (while *Integration with Editing* and *Graphical Editing* not so). Additionally, *Hide Selected Entity*, *Filter Specific Entity Type* and *Focus on Selected Entity* are useful, as it is easier to spot errors after hiding the previously checked elements and/or focusing on the unchecked ones.
Learning means gaining knowledge about the *domain* the ontology covers or learning about the ontology *itself* so as to use it. The view need not be

as detailed as for discovering errors (in *Inspection*). The user often only needs to see the available classes and properties, their hierarchy and their domain/range relationships; especially non-technical users who only want to learn the *domain* are unlikely to understand complex axioms anyway. Important features are thus *Zoom-Out Overview*, *Radar View* (to see an overview of the ontology) and *Incremental Exploration* (for user-friendly exploration of the ontology in more – but not too much – detail).

Sharing is similar to *Learning*, except for one more actor to whom the ontology is to be explained and shared with. The visualization should thus support displaying a part of the ontology; e.g., a picture of it can be made for an article describing the ontology. *Hide Selected Entity* is important for displaying the desired part only, and *Drag&Drop User Layout* is useful for achieving an appropriate layout, e.g., placing important entities into the center.

3 Multi-aspect Analysis of Visualization Tools

3.1 Analysis Overview

Initially we identified 21 visualization tools. For the 11 we considered as 'usable', we then characterized their supported features and language constructs. In Table 1, the "Plugin for" column specifies if the tool is a plugin for an ontology development environment (otherwise it is a standalone application). "Editor" is ticked if the tool supports editing of the visualized ontology. "Method" lists available visualization methods as defined in [13].[3] "Supports" contains "RDFS" if the tool only visualizes RDFS constructs and "OWL" if it visualizes at least some constructs from OWL. Finally and most important, in "State" we specify if we consider, by our hands-on experience, the tool as stable enough to be used in real use cases ("Usable"), still in early state of development ("Devel.") or not publicly available for download ("N/A"). The eleven 'usable' tools have been both evaluated in detail and included into the knowledge base of our recommender. The detailed analysis aimed to find out what features the tools implement (and how well), what language constructs they visualize and how they deal with larger ontologies. To test the last aspect we applied them on SUMO[4] (with several thousands of classes) and Biochemistry[5] (with several hundred classes).

3.2 Analysis of Selected Tools: Summaries

Table 2 shows the features (from Section 2) we found as supported in each tool. The number indicates the support level as evaluated by us: "1" means "only implemented partially and/or in a rather user-unfriendly way"; "2" means "fully implemented"; empty cell means "not implemented". The reasons for evaluating some of the features for certain tools with "1" are discussed in Section 3.3. Table 3 shows the language-level expressiveness of the tools, i.e. which OWL constructs can be visualized in it. Finally, Table 4 (left part) shows the 'suitability scores'

[3] Due to space we omit a mapping of these methods to our list of tool features.
[4] `http://www.ontologyportal.org/`
[5] `http://ontology.dumontierlab.com/biochemistry-complex`

Table 1. Ontology visualization tools overview

Tool	Plugin for	Editor	Method	Supports	State
CmapTools		x	Concept maps	OWL	N/A
CropCircles	SWOOP		Euler diagrams	RDFS	N/A
Entity Browser	Protégé 3/4	x	Indented list	RDFS	Usable
GLOW	Protégé 4.x		Node-link	RDFS	Devel.
Jambalaya	Protégé 3.x	x	Node-link, Space-filling	OWL	Usable
KC-Viz	Neon-Toolkit		Node-link	RDFS	Usable
Knoocks		x	Space-filling, Node-link	RDFS	Devel.
Navigowl	Protégé 4.x		Node-link	RDFS	Devel.
Ontograf	Protégé 4.x		Node-link	OWL	Usable
Ont. Visualizer	Neon-Toolkit		Node-link	RDFS	Usable
Ontoself			3D Node-link	RDFS	N/A
Ontosphere			3D Node-link	RDFS	Devel.
Ontoviewer			2.5D Node-link	RDFS	N/A
Ontoviz	Protégé 3.x		UML	RDFS	Usable
OWL VisMod		x	Space-filling, Node-link	RDFS	N/A
OWLeasyViz		x	Euler diagrams	RDFS	N/A
OWLGrEd		x	UML	OWL	Usable
OWLViz	Protégé 3.x		UML	RDFS	Usable
SOVA	Protégé 4.x		Node-link	OWL	Usable
TGVizTab	Protégé 3.x		Node-link	RDFS	Usable
TopBraid		x	Node-link	OWL	Usable

(*ss*) of each tool for each of the *use case categories*. The scores are calculated from the values in Table 2 using the following simple formula:

$$ss = \sum ImportantFeatureScores \cdot \alpha + \sum OtherFeatureScores \cdot \beta$$

Important features for each use case category are specified in Section 2. 'Other features' are all features which are not specified as important for the given use case category, with the exception of *Integration with Editing* and *Graphical Editing*, which are only taken into account for *Editing*. The feature score is 0, 1 or 2 as shown in Table 2. The multiplication coefficients α and β were set to 3 and 0.5, respectively, since these values provided good discriminatory ability in our initial test with the recommender system. The suitability scores are normalized to interval <-3;3> in the recommender (Section 4) and used as weights for the appropriate rules.

For the purposes of further evaluation using a questionnaire, we generalized the findings from the analysis into Table 4 (right part). It shows the performance of each tool in several aspects expressed on a scale of "very weak" (−−), "weak" (−), "strong" (+) and "very strong" (++). The "OWL" aspect means how complete the visualization is: whether the tool only displays classes or also object properties, datatype properties etc. The "C. Classes" column shows how well the tool displays complex classes. The remaining columns show the performance of the tool in each category (based on the suitability scores and our experience

Table 2. Features implemented in each tool

Tool	Zoom-out Overview	Radar View	Graphical Zoom	Focus on Selected Entity	History (undo/redo)	Pop-up Window / Tooltip	Incremental Exploration	Search	Hide selected entity	Filter Specific Entity Type	Fisheye Distortion	Edge Bundles	Drag&Drop Navigation	Drag&Drop User Layout	Clustering	Integration with Editing	Graphical Editing
Entity Browser	2					2	2	2								2	
Jambalaya	1		1	1		2	1	2	2	2	1			2		2	
KC-Viz	2		2		2	2	2	1	2	2			2	2	2		
Ontograf			2	2		2	2	2		1			2	2		1	
Ontology Visualizer	2		2	2	2		2	2						2			
Ontoviz			2						2				2				
OWLGrEd	1	2	1		2	2						1		2		2	2
OWLViz	2		1	2		2	2	1	2							2	
SOVA	1		2					2		2			2	2			
TGVizTab	1		1	2			2	2	2	2	1		2				
TopBraid		2	1						2					2			2

Table 3. Types of language constructs visualized by each tool

Tool	Classes	Object Prop.	Datatype Prop.	Instances	Annotations	Univ./Exist. Rest.	Cardinality	Enumeration	Intersection	Union	Complement	equivalentClass	disjointWith	subClassOf	Property Char.
Entity Browser	x	x	x	x											
Jambalaya	x	x		x		x						x		x	
KC-Viz	x	x		x					x					x	
Ontograf	x	x		x		x		x	x	x	x	x		x	
Ontology Visualizer	x													x	
Ontoviz	x	x	x	x		x	x	x	x	x	x	x			
OWLGrEd	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
OWLViz	x														
SOVA	x	x		x		x	x	x	x	x	x	x	x	x	x
TGVizTab	x	x												x	
TopBraid	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

Table 4. Suitability scores and generalized performance of each tool in various aspects

	Suitability Scores				Strong/weak aspects					
Tool	**Editing**	**Inspection**	**Learning**	**Sharing**	**OWL**	**C. Classes**	**Inspection**	**Editing**	**Learning**	**Sharing**
Entity Browser	20,0	14,0	14,0	4,0	−	−−	−	+	+	−−
Jambalaya	23,5	30,0	12,5	17,5	−	−	++	++	−	++
KC-Viz	18,0	28,0	20,5	20,5	−	−	++	−	++	++
Ontograf	20,5	25,0	12,5	12,5	+	+	++	+	−	+
Ontology Vizualizer	12,0	17,0	17,0	12,0	−−	−−	−	−−	+	+
Ontoviz	3,0	8,0	3,0	8,0	+	++	−−	−−	−−	−−
OwlGrEd	22,5	10,5	13,0	10,5	++	++	−−	++	−	−
OWLViz	19,5	23,5	16,0	11,0	−−	−−	++	+	+	−
SOVA	10,5	15,5	8,0	10,5	++	++	−	−−	−−	−
TGVizTab	12,5	27,5	15,0	12,5	−−	−−	++	−	+	+
TopBraid	9,5	8,5	8,5	13,5	++	++	−−	−−	−−	+

with the tools). The performance of the tools regarding large ontologies is only discussed verbally in the next subsection.

3.3 Analysis of Selected Tools: Details

Jambalaya [21] can load and display large ontologies thanks to the treemap view. It can perform *Zoom-Out Overview*, but too many edges crossing other edges and nodes, as well as node overlap in the node-link view and hard-to-read labels in the treemap view makes it less useful in comparison with other tools. To use *Graphical Zoom*, the user has to first switch to 'zooming mode' – intuitive zooming with mousewheel is not supported. Nodes can be retracted/expanded, but Jambalaya displays the whole ontology by default – incremental exploration cannot be started from a selected node. *Fisheye-distortion* is applied only on the selected node and does not include its surroundings.

The strong feature of **KC-Viz** is the automated selection of 'most important classes' called 'key concepts'. This makes it very suitable for large ontologies. It offers a large number of well implemented features which makes it suitable for most of the use case categories.

Ontograf[6] can only be used for smaller ontologies and is limited to RDFS expressiveness.

Although **Ontology Visualizer**[7] only displays the class hierarchy and its range of features is limited, it is quite suitable for *Learning* thanks to its implementation of incremental exploration, and it can deal with larger ontologies.

[6] `http://protegewiki.stanford.edu/wiki/OntoGraf`
[7] `http://neon-toolkit.org/wiki/Main_Page`

Ontoviz[8] displays the most detailed view by default and it is incapable of displaying an *overview* of the whole ontology. It can visualize a larger ontology if the user wants to see only a small part of it at once: the user can select exactly what part of the ontology should be displayed.

The main advantage of **OWLGrEd** [2] is its large coverage of OWL constructs. Its use for *Editing* is supported, e.g., by the possibility to directly draw relationships as edges between the nodes. The *Zoom-out Overview* is possible but not very usable: the labels are hard to read and the view is cluttered. It implements some sort of edge bundles but not as well as, e.g., GLOW [11]. As in the case of Ontograf, larger ontologies are not supported well: OWLGrEd was not capable to load SUMO, and for Biochemistry the ontology was loaded correctly but the result was an extremely cluttered visualization without any chance to determine which nodes the edges are connecting.

As **OWLViz**[9] was able to load the SUMO ontology, it can be considered for visualizing large ontologies. However, it displays only classes and their hierarchy.

Protégé[10] contains the indented list ontology visualization, **Entity Browser**, as an integral part of its GUI. As the Neon-Toolkit[11] and TopBraid Composer[12] offer almost identical implementations of indented-list-based entity browsers, we rather provide a generic analysis of this method that applies to all three. It offers a sort of *Zoom-Out Overview* by default (the simple list of entities), while the features related to node-link visualization are obviously unsupported. *Editing* is inherent, and even very large ontologies can be visualized without problems.

SOVA is the only tool in this survey that displays all OWL constructs (but datatype properties) as graphical elements in one view. Large ontologies can be displayed in a simplified alternative view similar to OWLViz (classes-only).

TGVizTab [1] offers a few advanced, but imperfectly implemented, features like fisheye-distortion. Its main disadvantage is that it is available only as a plugin for the older Protégé 3.

The node-link visualization of **TopBraid Composer**[13] shows all types of entities as nodes and properties as edges connecting them. The visualization is rather provided at the RDF level, so even `owl:Class` is shown as a separate node and every class is connected to it through an `rdf:type` edge. While good for learning the OWL language, this feature does not contribute to clearness of the visualization. Even if such redundant elements are hidden, the visualization gets cluttered quite quickly and thus it is not suitable for large ontologies.

[8] `http://protegewiki.stanford.edu/wiki/OntoViz`
[9] `http://www.co-ode.org/downloads/owlviz/`
[10] `http://protege.stanford.edu`
[11] `http://neon-toolkit.org`
[12] Discussed below with respect to its node-link method.
[13] `http://www.topquadrant.com/products/TB_Composer.html`

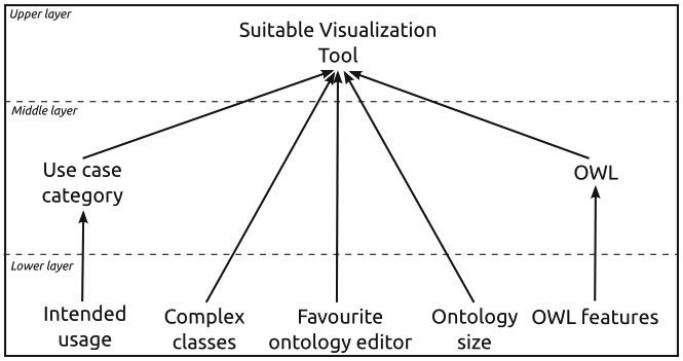


Fig. 2. Abstracted inference network for the recommendation knowledge base

4 Ontology Visualization Tools Recommender

The recommender[14] is built as a knowledge base (KB) for the NEST expert system shell [3]. NEST covers the functionality of compositional rule-based expert systems (with uncertainty handling), non-compositional (Prolog-like) expert systems, and case-based reasoning systems. NEST employs a combination of backward and forward chaining and it processes uncertainty according to the algebraic theory of Hajek [8]. In order to capture (task-specific) domain knowledge for rule-based reasoning, we employed the following apparatus of NEST:

- *Attributes and propositions.* Attributes are used to describe the features of the consulted case, and propositions are derived from the values of attributes. There are four types of attributes: binary, single nominal, multiple nominal, and numeric. *Fuzzy intervals*, for a numeric attribute, allow to express vague information such as *high body temperature*.
- *Rules* having a *condition* (disjunction of literal[15] conjunctions) and *conclusion* (list of literals) component. There are three types of rules. A *compositional* rule has its conclusion equipped with a weight expressing the degree of uncertainty of the conclusion if the condition holds with certainty. Furthermore, to evaluate the ultimate weight of a proposition, all contributions having this proposition in its conclusion are evaluated and combined. An *apriori* rule is a compositional rule without condition. Finally, a *logical* rule is a non-compositional rule without weights.

Our KB contains 8 attributes, 36 propositions, 32 compositional rules and 1 apriori rule. An abstraction of its inference network is in Fig. 2. Directed edges indicate groups of compositional rules connecting the attributes (shown as texts); the grouping is based on the corresponding propositions.[16] The KB consists of

[14] The recommender is available at `http://owl.vse.cz:8080/OVTR/`.
[15] 'Literal' is not meant here in the RDF sense but as 'attribute-value pair'.
[16] The full inference network is at `http://owl.vse.cz:8080/OntoVisualTool/`.

three layers. The top layer only contains one node, representing the *recommendation of visualization tool*. The middle layer contains two nodes, which aggregate the relevant answers from the user: *Use case category* (editing, inspection, learning and sharing) and *OWL* (the importance of particular OWL features). The bottom layer represents possible user answers to questions:

- *Complex classes*: importance of anonymous classes based on various OWL constructs such as union, complement, intersection etc.
- *Intended usage*: the nine use cases from Section 2.
- *Ontology size*: fuzzy intervals for 'small', 'medium' and 'large' ontologies.
- *OWL features*: importance of particular OWL features (object properties, interclass relationships, datatype properties and property characteristics), aiming to infer the overall importance of OWL support in the visualization.
- *Favorite ontology editor*: the user's preference for some (freely available) ontology editor: Protégé 3, Protégé 4 or Neon Toolkit.

This KB thus encompasses the main insights we gained from 'roadmapping the ontology visualization landscape' as presented in the first part of the paper.

4.1 Recommender Usage Example

We include an example of usage of the recommender. The particular source case refers to paper [17], where *Ontoviz* is used to show (as screenshots) parts of the ontology that is being described in the paper. We tried to emulate the hypothetical entering of information about the ontology into the recommender by the paper authors. Technically, this consisted in answering the above mentioned questions with Likert-scale answers (represented by numbers from the interval <-3;3> where 3 means "Certainly yes" and -3 means "Certainly no"), or with exact numbers for numeric questions, as follows: *Complex classes*: 3; *Intended usage*: Screenshots: 3; *Ontology size*: 91; *OWL features*: Object Properties: 1, Interclass Relationships: 3, Datatype Properties: 1, Property Characteristics: -1;[17] *Favorite ontology editor*: Protégé 3: 3,[18] Protégé 4: -2, Neon Toolkit: -3.

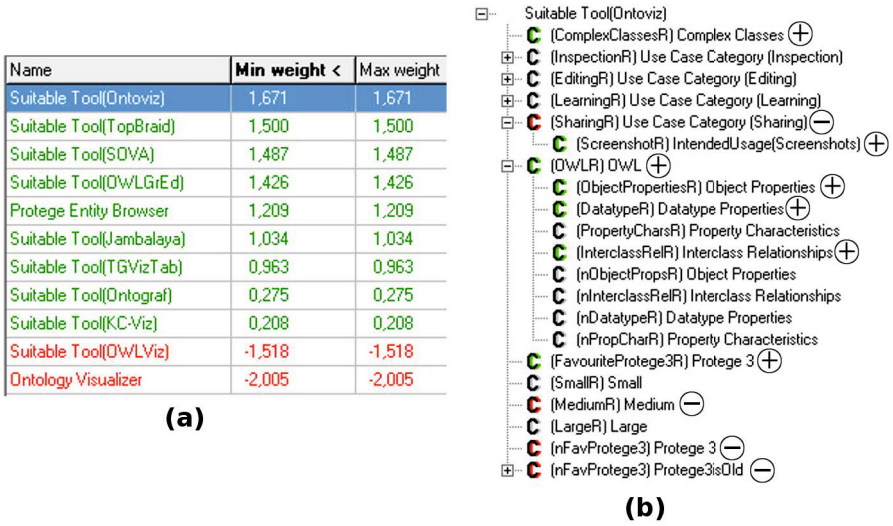
The recommendation is in Fig. 3 (a). The tool indeed recommends *Ontoviz* with the weight of 1.671 (of the maximum of 3). The next most suitable tool is *TopBraid*. Least suitable, in turn, are *Ontology Visualizer* and *OWLViz*.

The final weight of the proposition supporting *Ontoviz* has been inferred as indicated in Fig. 3 (b), where the rule hierarchy is shown. Rules marked as green by the explanation component[19] of NEST (and annotated with a circled-plus sign for the sake of B/W readability, in the screenshot only) positively contribute to the weight of their superordinate rule; rules marked as red (circled-minus sign in the screenshot) contribute negatively; rules in grey are indifferent in this

[17] The questions regarding OWL properties have been answered with lower weights since there are only 7 properties in the ontology. Property Characteristics are not mentioned in the paper, so we consider them as unimportant.

[18] Protégé 3 is mentioned in the paper as the tool used for the ontology development.

[19] Currently only in the desktop version, `http://sorry.vse.cz/~berka/NEST/`.



Name	Min weight <	Max weight
Suitable Tool(Ontoviz)	1,671	1,671
Suitable Tool(TopBraid)	1,500	1,500
Suitable Tool(SOVA)	1,487	1,487
Suitable Tool(OWLGrEd)	1,426	1,426
Protege Entity Browser	1,209	1,209
Suitable Tool(Jambalaya)	1,034	1,034
Suitable Tool(TGVizTab)	0,963	0,963
Suitable Tool(Ontograf)	0,275	0,275
Suitable Tool(KC-Viz)	0,208	0,208
Suitable Tool(OWLViz)	-1,518	-1,518
Ontology Visualizer	-2,005	-2,005

Fig. 3. (a) The recommendation results. (b) The inference explanation.

particular inference. *Ontoviz* scores high in this case since it visualizes all the required types of OWL entities including complex classes and it is a plugin for Protégé 3, which is used (i.e. preferred) by the authors of the paper. The score is lowered by the "SharingR" rule, as Ontoviz is not very suitable for the *Sharing* use case category, by the "MediumR" rule, as *Ontoviz* can only clearly display a small number of entities, and by the two rules at the bottom of the list, which are built-in rules slightly lowering the score of Protégé 3 plugins, as it has been meanwhile replaced by the newer version (4) and is not supported anymore.

5 Evaluation

As the evaluation of both the overall analysis and the recommender requires human expertise,[20] we prepared a web-based *anonymous questionnaire*,[21] sent invitations to participate in it to several relevant mailing lists, and also asked several people from the area of ontology engineering, including authors of the surveyed visualization tools, directly. The respondents were asked about their level of expertise in ontologies and their experience with ontology visualization tools: which they used, in what use case (out of the 9 use cases described in Section 2) and whether they were satisfied with it. Then, they were asked whether they agree with the categorization of their use case, with the weak and strong aspects of the visualization tool as inferred from our analysis and with our categorization system itself. Finally, a consultation with the recommender was offered

[20] Yet, we also made a literature-based evaluation, available at `http://bit.ly/1phLBdm` along with additional details about the evaluation described in this section.

[21] Available at `http://owl.vse.cz:8080/OVQuestionnaire/`

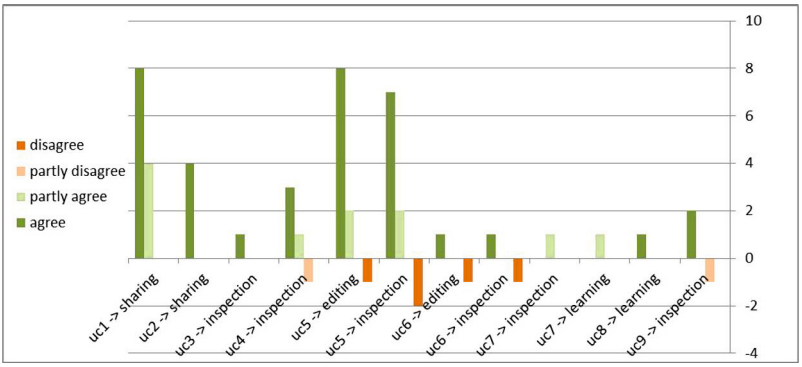


Fig. 4. Counts of respondent opinions to each use case → category mapping

and if they ran through it, they were asked to express their satisfaction with the resulting recommendation. During the consultation, the respondents could describe either a real visualization scenario or a hypothetical one. We gathered answers from 32 respondents, out of which 3 skipped the consultation.

Agreement with the Categorization System. When introduced to a brief description of the use case categories (as described in Section 2), 13 respondents answered that the categorization makes "perfect sense", the same number of respondents stated that it makes "more or less sense", 5 replied that it "does not make much sense", and no one chose "does not make sense at all"; 1 respondent did not answer this question. To sum up, about 84% of respondents partly or fully agreed with the categorization.

Agreement with the Categorizations of Use Cases. Figure 4 shows the counts of respondent opinions about specific use case mappings to category (partial and full disagreement counts are shown as negative values). Each respondent could describe up to three different use cases. We gathered 40 opinions. Agreement clearly prevailed, with the exception of *uc6*, *Adapting an existing ontology*, where one respondent agreed and one disagreed with the mappings to both editing and inspection.

Agreement with Strong/Weak Aspects of the Tools. When the respondent stated s/he has experience about some visualization tool, s/he was asked about his/her opinions about each of the strong/weak aspects of the tool as shown in the right part of Table 4. We mapped the answers to numerical values as follows: "agree" → "1", "partly agree" → "0.5", "partly disagree" → "-0.5", and "disagree" → "-1". The average of the numerical values of all 141 answers from 27 respondents was approx. 0.34. If we look at the average agreement on each aspect of each tool separately, only the weak and very weak aspects of SOVA and OwlGrEd had averages below 0 (i.e. the respondents rather disagreed).

Satisfaction with the Consultation. 29 respondents ran the consultation, of which 14 (48%) were satisfied and 7 (24%) partly satisfied with the resulting recommendation, i.e. 72% of respondents were partly or fully satisfied. In 23 (79%) cases, KC-Viz was recommended as the most suitable tool. Such a high percentage of one tool was mainly caused by the fact that most of the respondents (approx. 66%) entered an ontology size larger than 120 entities, which is considered by the recommender as 'large', and KC-Viz is considered as the most suitable tool for large ontologies. These results suggest that we should reconsider the rules in the KB related to the size of the ontology (the risk of clutter when a larger ontology is visualized might be over-estimated for some tools).

6 Related Work

We are unaware of a recent analysis with as large coverage of visualization tools as in this paper, nor of an implemented system for tool recommendation. Our questionnaire was inspired by a survey [6] done several years ago aimed at discovering which visualization tools are used by whom and for what tasks. The options offered to the respondents regarding the usage of the tool partially agree with our use case categories: "Check for inconsistencies or errors" is our *Inspection*, "Present reports to others" is *Sharing* and "Help with understanding new ontologies" is *Learning*. We consider the remaining three options either too specific ("Show hidden relationships" and "Show areas of interest") or too general ("Assist with navigating information space"). [12] describes the results of a comparative evaluation of (Protégé) Entity Browser, Jambalaya, TGViz and Ontoviz done with a group of test users. Time needed to perform a set of several predefined tasks in each tool by each user was measured and the users were also asked to fill in a questionnaire after using the tools. The users achieved the best performance with Entity Browser, which also received the best score regarding questions about perceived effectiveness. The tasks were aimed at finding specific information about instances, which would be classified as *Learning*. Our results agree with those from [12] in that Entity Browser is slightly better for *Learning* than Jambalaya and that Ontoviz is the worst of the four tools. The difference is in the evaluation of TGVizTab, which performed worse than Entity Browser in [12] but has a higher suitability score for *Learning* in our study. However, [12] compares the tools using a single ontology and a specific use case, while our analysis is aimed at comparing the tools regarding different ontologies and use cases. A subsequent paper by the same group, [13], defines a categorization of visualization methods and tasks, and includes a thorough (but ageing) survey of ontology visualization tools. An approach similar to [12] has been chosen in [22] for a comparative evaluation of Ontograf, OWL2Query and DLQuery: a group of users was asked to perform a set of predefined tasks aimed at finding information about instances and the time to complete each task was measured. A review of Protégé Entity Browser, Ontoviz, OntoSphere, Jambalaya and TGVizTab is available in [20]. The review includes a comparison of features and types of OWL entities supported by each visualization tool similar to our

Table 2 and Table 3. While it is less detailed and lacks evaluation of the quality of implementation of the features, it includes a review of support of 'Animated Transactions',[22] which we considered unimportant. Finally, [5] proposes rough guidelines for visualization tool design.

7 Conclusions and Future Work

We overviewed and analyzed the current ontology visualization tools by taking into account newly formed (aggregated) *use case categories*, *visualization tool features*, *language constructs*, and *scalability* with respect to ontology size. For selecting a suitable visualizer we designed a simple *recommender*. We also performed a *questionnaire-based evaluation* of the recommender and of our analytical findings. The respondents generally agreed with the proposed use case categorization and with the strong and weak aspects we identified for the tools, and they were mostly satisfied with the recommender suggestions.

In the future we plan to improve the recommender according to the feedback from the questionnaire, in particular, the rules concerning the size of the ontology to be visualized. Moreover, since our analysis of the ability of the tools to display large ontologies is rather subjective, we plan to arrange for a more exact assessment in this respect. We will also continuously monitor the visualization 'landscape' so as to include new tools, such as VOWL[23] and OLSViz,[24] and reflect them in the KB. Furthermore, a next release of the KB will directly consume input from automatic analysis of ontology structure.[25] Finally, we are aware that our treatment of visualization use cases is so far only based on experience in the semantic web area. Insights from the broader human-computer interaction research, such as [23], should also be exploited.

Acknowledgement. The research is supported by VŠE IGA grant F4/34/2014 (IG407024). Ondřej Zamazal is supported by the CSF grant 14-14076P, "COSOL – Categorization of Ontologies in Support of Ontology Life Cycle". We thank to Vladimír Laš for help with setting up our KB for the web variant of NEST, and to the anonymous respondents of the survey.

References

1. Alani, H.: TGVizTab: An ontology visualisation extension for Protege. In: K-CAP 2003 Workshop on Visualization Information in Knowledge Engineering (2003)
2. Bardzins, J., et al.: OWLGrEd: A UML Style Graphical Editor for OWL. In: Ontology Repositories and Editors for the Semantic Web, Hersonissos (2010)

[22] Animation of the transition between different states of the visualization.
[23] `http://vowl.visualdataweb.org/`
[24] `http://ols.wordvis.com/`
[25] A preliminary implementation of such an analysis is at `http://owl.vse.cz:8080/MetricsExploration/`, but not yet integrated with the recommender.

3. Berka, P.: NEST: A Compositional Approach to Rule-Based and Case-Based Reasoning. In: Advances in Artificial Intelligence, 15 p. (2011), http://www.hindawi.com/journals/aai/2011/374250/
4. Brockmans, S., Volz, R., Eberhart, A., Löffler, P.: Visual modeling of OWL DL ontologies using UML. In: McIlraith, S.A., Plexousakis, D., van Harmelen, F. (eds.) ISWC 2004. LNCS, vol. 3298, pp. 198–213. Springer, Heidelberg (2004)
5. Da Silva, I.C.S., Freitas, C.M.D.S., Santucci, G.: An integrated approach for evaluating the visualization of intensional and extensional levels of ontologies. In: Proceedings of the 2012 BELIV Workshop: Beyond Time and Errors-Novel Evaluation Methods for Visualization, p. 2. ACM (2012)
6. Ernst, N.A., Storey, M.-A.: A Preliminary Analysis of Visualization Requirements in Knowledge Engineering Tools. University of Victoria (2003)
7. Freitas, C.M.D.S., et al.: On evaluating information visualization techniques. In: Proceedings of the Working Conference on Advanced Visual Interfaces, pp. 373–374. ACM, New York (2002)
8. Hajek, P.: Combining functions for certainty degrees in consulting systems. International Journal of Man-Machine Studies 22(1), 59–76 (1985)
9. Herman, I., Melanon, G., Marshal, M.S.: Graph visualization and navigation in information visualization: A survey. IEEE Transactions on Visualization and Computer Graphics 6(1), 24–43 (2000)
10. Hop, W., et al.: Using Hierarchical Edge Bundles to visualize complex ontologies in GLOW. In: Proceedings of the 27th Annual ACM Symposium on Applied Computing, pp. 304–311. ACM (2012)
11. Howse, J., Stapleton, G., Taylor, K., Chapman, P.: Visualizing ontologies: A case study. In: Aroyo, L., Welty, C., Alani, H., Taylor, J., Bernstein, A., Kagal, L., Noy, N., Blomqvist, E. (eds.) ISWC 2011, Part I. LNCS, vol. 7031, pp. 257–272. Springer, Heidelberg (2011)
12. Katifori, A., et al.: A comparative study of four ontology visualization techniques in protege: Experiment setup and preliminary results. In: IEEE Information Visualization 2006, pp. 417–423 (2006)
13. Katifori, A., et al.: Ontology visualization methods – a survey. ACM Computing Surveys (CSUR) 39(4), 10 (2007)
14. Kendall, E.F., Bell, R., Burkhart, R., Dutra, M., Wallace, E.K.: Towards a Graphical Notation for OWL 2. In: OWLED 2009 (2009)
15. Motta, E., Mulholland, P., Peroni, S., d'Aquin, M., Gomez-Perez, J.M., Mendez, V., Zablith, F.: A novel approach to visualizing and navigating ontologies. In: Aroyo, L., Welty, C., Alani, H., Taylor, J., Bernstein, A., Kagal, L., Noy, N., Blomqvist, E. (eds.) ISWC 2011, Part I. LNCS, vol. 7031, pp. 470–486. Springer, Heidelberg (2011)
16. Parreiras, F.S., Walter, T., Gröner, G.: Visualizing ontologies with UML-like notation. In: Ontology-Driven Software Engineering. ACM (2010)
17. Rene Robin, C.R., Uma, G.V.: Development of educational ontology for software risk analysis. In: Proceedings of the 2011 International Conference on Communication, Computing & Security, ICCCS 2011, pp. 610-615. ACM (2011)
18. Sarkar, M., Brown, M.H.: Graphical fisheye views of graphs. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 83–91. ACM (June 1992)
19. Shneiderman, B.: The eyes have it: A task by data type taxonomy for information visualizations. In: Proceedings of the IEEE Symposium on Visual Languages, pp. 336–343. IEEE (1996)

20. Sivakumar, R., Arivoli, P.V.: Ontology Visualization Protege Tools: A Review. International Journal of Advanced Information Technology 1(4) (2011), http://airccse.org/journal/IJAIT/papers/0811ijait01.pdf
21. Storey, M., et al.: Jambalaya: Interactive visualization to enhance ontology authoring and knowledge acquisition in Protégé. In: Workshop on Interactive Tools for Knowledge Capture, K-CAP-2001 (2001)
22. Swaminathan, V., Sivakumar, R.: A Comparative Study of Recent Ontology Visualization Tools With a Case of Diabetes Data. International Journal of Research in Computer Science 2(3), 31 (2012)
23. Weidong, H. (ed.): Handbook of Human Centric Visualization. Springer (2014)