Intuitive Semantic-Editing for regarding Needs of Domain-Experts

Dirk Burkhardt, Cristian Hofmann, Kawa Nazemi, Christian Stab, Matthias Breyer, Dieter W. Fellner
Fraunhofer Institute for Computer Graphics Research
Fraunhoferstrasse 5, 64283 Darmstadt, Germany
{dirk.burkhardt, cristian.hofmann, kawa.nazemi, christian.stab, matthias.breyer, d.fellner}@igd.fraunhofer.de

Abstract: Ontologies are used to represent knowledge and their semantic information from different topics, to allow users a better way to explore knowledge and find information faster, because of the data-structuring. To achieve a well filled knowledgebase, editors have to be used, to enter new and to edit existing information. But most of the existing ontology-editors are designed for experienced ontology-experts. Experts from other topic fields e.g. physicians are often novices in the area of ontology-creating, they need adequate tools, which hide the complexity of ontology-structures. In the area of e-learning experts are also teachers as well. In this paper we will present a method, how the needs of domain-experts can be regarded and so an editor can designed, which allows an editing and adding of information by users without having experiences of creating ontologies. With such an editor domain-experts are able to commit their expert-knowledge into the ontology.

Introduction

In the current media-dominated world, where actuality of information and extensive knowledge is very important, adequate methods for storing information are indispensable. Since a few years many researches were done in the part of semantic structures, which will saved in ontology-based file-formats. Digital ontologies provide the facility to store the information and bring them in relation to each other. In consequence a user of interactive systems is able to explore within this semantic information and also technical autonomously working systems are able to work with the semantically stored data.

To create, change und enrich such ontologies a lot of programs are existing, but because of the design for technical experienced people, these tools working ontology-based and provide specific functions, which will only understand by ontology-experts. Today only ontology-driven editors exists, which are not usable for non-ontology experts. For creating ontologies, which is filled with knowledge of domain-experts, other tools are necessary, which will regard the needs of domain-experts like physics or chemist. Such domain-experts often have no experiences in creating ontologies and what technical specification exist and have to be considered. This group of users need tools, which are intuitive and present the information on an abstract level. To allow this group to commit their knowledge, they need tools, which allow an easy integration of new semantics information. So also the editor must provide an abstract representation of the data, which have to be entered. The challenge for these goals are, how the ontology-based entries can be converted and presented on an abstract level and how these abstracted information after the editing-process of the domain-experts can be converted to an ontology-entry.

In this paper we describe a method, how the needs of domain-experts can be regarded, so that they can bring in their domain-knowledge for creating ontologies which comprises a lot of different topics. Especially the graphical user interface is in our focus, so that domain-experts get an easy access to knowledge and extend the semantic data for domain-specific knowledge. This method will prototypical implemented and evaluated by different domain-experts. Actually we present an enhanced version of the Semantic-Editing tool, which will be developed within the ongoing THESEUS-program (see THESEUS-program 2006). This research program has as main goal, to simplify the access to information, data and services on the internet.

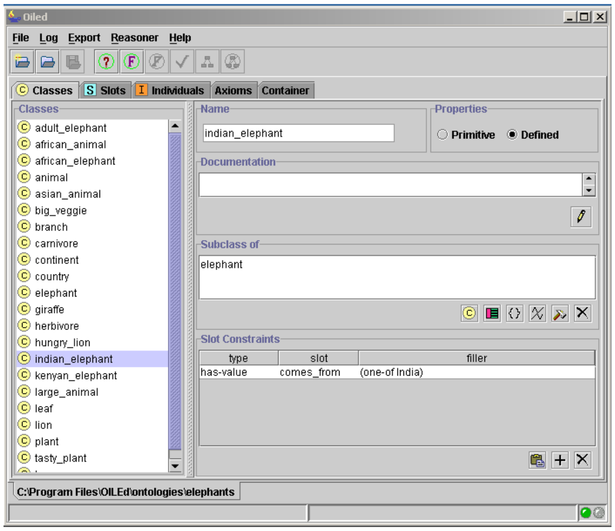
Related Works

For edit semantic data, which are defined in standardised description ontology languages like RDF (abbr.: Resource Description Framework) or OWL (abbr.: Web Ontology Language), a lot of tools exist. In this chapter an overview will given to established tools, which allow making changes in ontologies. A larger overview about exiting semantic-visualizations and editors are published in (Katifori et. al. 2007). All of the currently existing tools are primary designed for ontology-experts and support ontology-specific definition-aspects. The weight by all these tools lay on the definition of ontology-based elements, like literals with specific type definitions and inheritance of concepts and specific instantiation depending on such classes. But an ontology-novice will not understand what all these necessary definitions stand for. Also the creation of relations between instances is hard for an ontology-novice, because he only knows the semantically relation, but not the ontology-based relation-type in form of an URI (abbr.: Uniform Resource Identifier). So the complexity of ontologies, also the technical experiences, which are also needed for working with these tools, makes creating and editing of an ontology difficult for domain-experts, with missing experiences in designing ontologies.

OilEd

OilEd (see OilEd 2003) is an open-source ontology-editor to create ontologies basing on the DAML+OIL format (see Davis et. al. 2003). This ontology-editor is no fully fledged ontology development environment, but it provides the most common tools to create and edit ontologies.

OilEd is an easy to use toolkit with the basic tools, but the user interface is ontology-driven, so that a user must have knowledge about ontologies. It is hard to use for people who does not have experiences in semantic date formats.



adult_elephant
african_animal
african_elephant
animal
asian_animal
big_veggie
branch
carnivore
continent
country
elephant
giraffe
herbivore
hungry_lion
indian_elephant
kenyan_elephant
large_animal
leaf
lion
plant
tasty_plant

Name: indian_elephant

Properties: ○ Primitive ● Defined

Subclass of: elephant

type	slot	filler
has-value	comes_from	(one-of India)

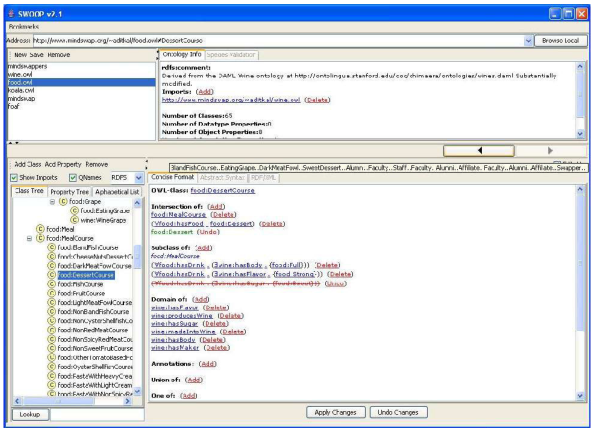
C:\Program Files\OILEd\ontologies\elephants

Figure 1. Screenshot of OilEd

SWOOP

SWOOP is the abbreviation for Semantic Web Ontology Overview and Persual (see SWOOP 2004). It is a quite simply understandable and clearly designed OWL Ontology Browser and Editor and has a similar look&feel like established web-browsers. With this tool also multiple ontologies can be administrated and combined, next to the managing of a single ontology. It is also possible to use this program as mindmap.

This editor is also primary designed for experienced ontology-experts, so that it does not contain a visualization to show the existing semantic-objects in a semantic-visualization.



rdfs:comment:
Derived from the DAML Wine ontology at http://ontolingua.stanford.edu/doc/chimaera/ontologies/wines.daml Substantially modified.
Imports: (Add)
http://www.mindswap.org/~aditkal/wine.owl (Delete)

Number of Classes:65
Number of Datatype Properties:0
Number of Object Properties:8

OWL-Class: food:DessertCourse

Intersection of: (Add)
food:MealCourse (Delete)
(∀food:hasFood . food:Dessert) (Delete)
food:Dessert (Undo)

Subclass of: (Add)
food:MealCourse
(∀food:hasDrink . (∃wine:hasBody . {food:Full})) (Delete)
(∀food:hasDrink . (∃wine:hasFlavor . {food:Strong})) (Delete)
~~(∀food:hasDrink . (∃wine:hasSugar . {food:Sweet}))~~ (Undo)

Domain of: (Add)
wine:hasFlavor (Delete)
wine:producesWine (Delete)
wine:hasSugar (Delete)
wine:madeIntoWine (Delete)
wine:hasBody (Delete)
wine:hasMaker (Delete)

Annotations: (Add)

Union of: (Add)

One of: (Add)

Figure 2. Screenshot of SWOOP

OntoTrack

OntoTrack (see Liebig and Noppens 2004) is an ontology authoring tool for ontology language OWL-light and offers a user friendly Graphical User Interface, which allows the users navigation and manipulation of large ontologies. The system based on SpaceTree (see Plaisant et. al. 2002) and is implemented in Java2D.

OntoTrack is an intuitive useable tool, to explore and manipulate ontologies and it is also useable for less experienced users in ontology-modelling. OntoTrack provide only one visualization-graph, which allows no adequate adaption to users perceptions of individual aspects.

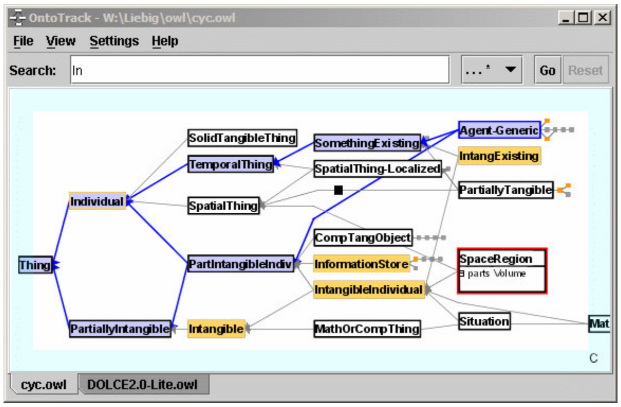


Figure 3. Screenshot of OntoTrack

Protégé

Protégé (Noy et. al. 2000) is one the most established ontology-editor for creating and manipulating ontologies. This program can be extended with several plugins to provide further features and functionalities. Some interesting plugins are *TGVizTab* and *Jambalaya*, which extending the Protégé tool with user friendly visualizations.

Protégé provides a lot of functions and by the use of plugins, it also can provide easy to use graphical interfaces also to non-ontology experienced users. For manipulation of semantic-data, an understanding for ontology is necessary. Currently is no plugin existing, which allows editing on abstract level, so that non-ontology experts cannot use this tool without risking mistakes in the ontology.

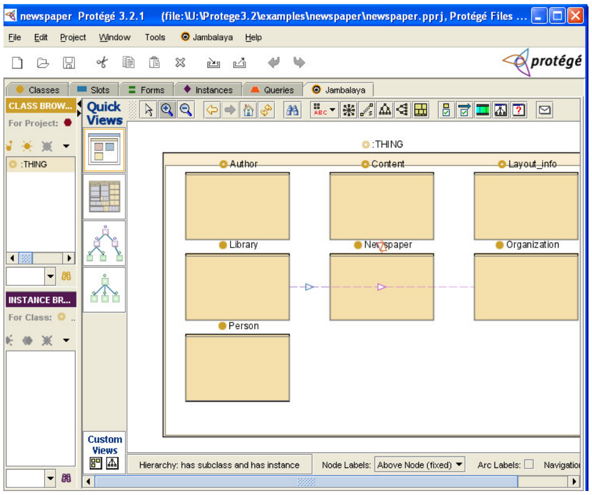

newspaper Protégé 3.2.1 (file:\U:\Protege3.2\examples\newspaper\newspaper.pprj, Protégé Files ...
File Edit Project Window Tools Jambalaya Help
protégé
Classes
Slots
Forms
Instances
Queries
Jambalaya
CLASS BROW...
For Project:
:THING
INSTANCE BR...
For Class:
Quick Views
Custom Views
:THING
Author
Content
Layout_info
Library
Newspaper
Organization
Person
Hierarchy: has subclass and has instance
Node Labels: Above Node (fixed)
Arc Labels:
Navigatio

Figure 4. Screenshot of Protégé with Jambalaya-Plugin

Concept of an intuitive Semantic-Editing System

In the related works some selected ontology-editors were presented, which are established in the scientific community. These systems basing on the idea to realize a tool for users, whose are ontology-experts and have experiences in designing ontologies and ontology-structures. Instead of these approaches, we created a concept, to regard the knowledge of domain-experts e.g. chemists or physicist. But often these experts are novices in ontologies and how ontologies have to be structured and how they get modeled.

In this chapter we present an approach, how domain-experts can bring in their knowledge, without knowing how to create, model and construct ontologies, so the complexity will be hidden to these domain-experts. We also regard the individual perceptions of the experts, so that the experts get an individualized editing environment.

Specification of the Editing-Process

During the conceptualization we define at first the default lifecycle of the Editing-Process. This is necessary, because the knowledge about the process makes it easy to determine, which tools the users need in every phase of process. We split the Editing-Process into 4 sequential phases (see Figure 5).

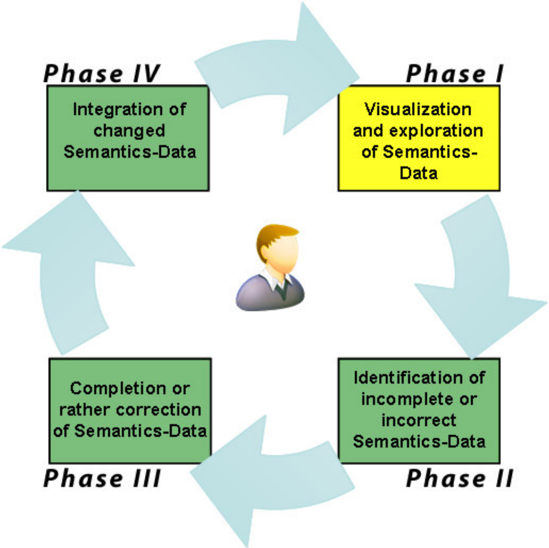

Phase IV
Integration of changed Semantics-Data
Phase I
Visualization and exploration of Semantics-Data
Identification of incomplete or incorrect Semantics-Data
Phase II
Completion or rather correction of Semantics-Data
Phase III

Figure 5. Lifecycle of the Editing-Process

In the first phase, the user explores through the semantics-data. This phase is not directly a part of editing, but it is the initial point, to start the editing-process. In this phase, the users can interact through the semantics and can identify mistakes in the data.

In the second phase, the user locates the errors and wants to correct it or wants to complete the existing semantics. Up to this phase, the "real" editing-process begins. During this phase he needs tools, where he can concrete select /define the mistaken or rather missing date-elements.

In the third phase, the user corrects the date or rather completes the existing data. In this phase the user needs adequate tools to correct or rather complete existing data. This phase is critical, because in this phase the usage must be easy to understand for the user (also for ontology-novices) on the one hand, and syntactically (especially type-conform with the underlying data-format e.g. an ontology) correct for the later integration into the exiting semantic-structure on the other hand. The challenge for this phase is, how the data can be converted on an abstract level and later on can reconverted into the ontology-format.

In the fourth phase, the made changes were converted to the format of the underlying data-form (e.g. an ontology) and will overtaken. After the integration, the visualization will be updated, to show the changed and added elements, too. If an ontology-management system is used, the changes have to be evaluated by the ontology-management too. If there were errors, than the system can ignore the made changes or repeat since phase 3.

SemaVis-Framework as Base

In the past we created the SemaVis-Framework (Nazemi et. al. 2009), which regards the individual aspects of the users, to visualize semantics-data (in type of information) personalized to users.

To achieve this personalization, different visualizations were made, which are differently qualified to visualize information e.g. tree-based visualization like SeMap (Nazemi et. al. 2009), to show hierarchical data-structures. All of these visualizations are adaptable by predefined attributes, which allows changes on the layout and presentation-primitives. We also use a self-developed role-based user-interaction module, to personalize the visualizations to the users' perceptions.

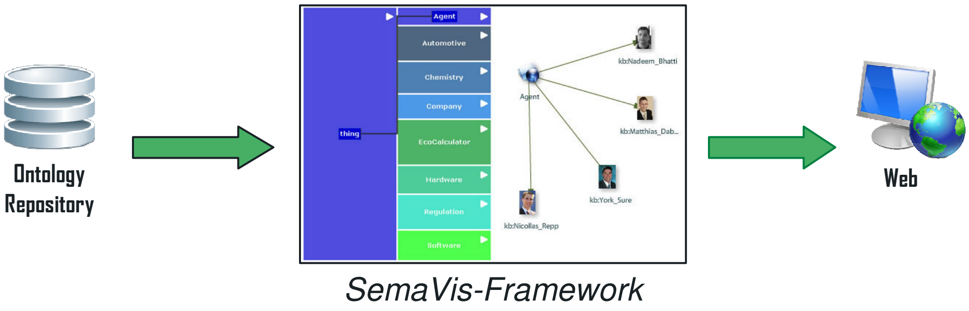


Figure 6. Simplified Concept of the SemaVis-Framework

The framework has a modular structure, specified by clear defined interfaces, so that further modules and components can easily be added. The framework can be configured at startup per configuration file and during the runtime – this allows data-changes from external components.

Semantic-Editing as Extension of the SemaVis-Framework

To provide the adaptation aspects next to the editing-functionality, the SemaVis-Framework (Nazemi et. al. 2009) is extended by the Editing-Component (see Figure 7). We will add features, so that the graphical user interface will consists of two layers. On the bottom-layer the visualizations from the SemaVis-Framework will be presented. The top-layer is used by the Editing-Components.

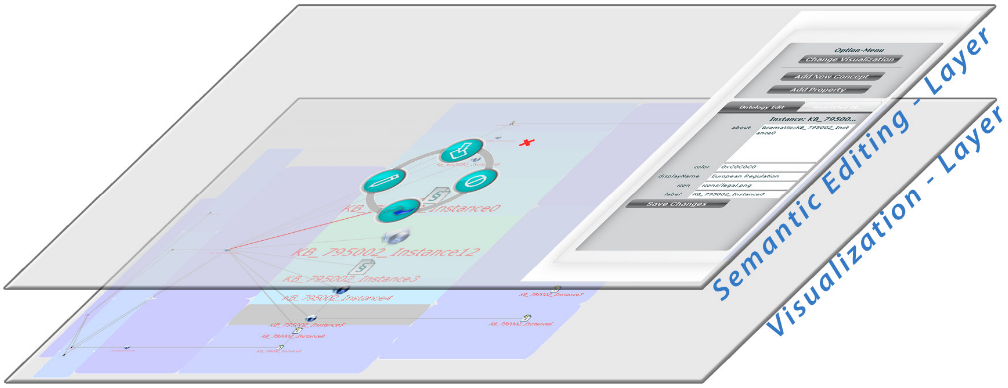
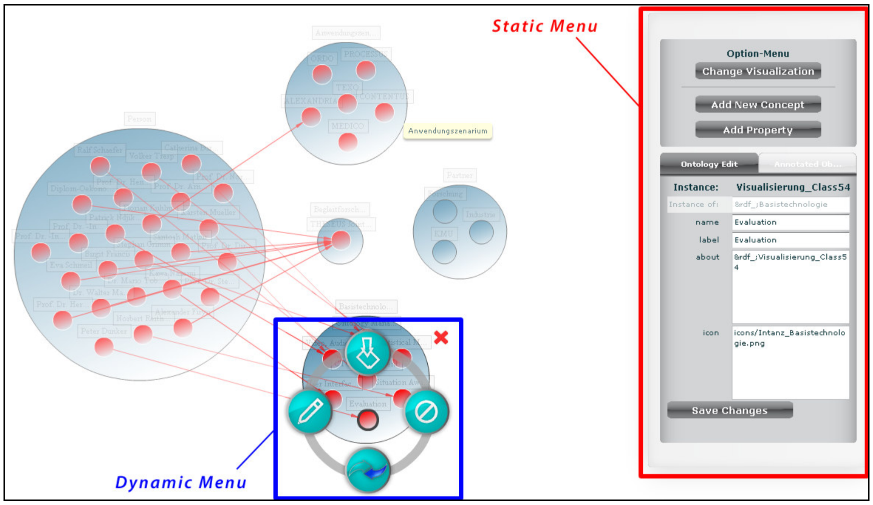


Figure 7. Layerstructure of the SemaVis-Framework and the Editing-Component

The Editing-Components consists of two different visual component types (see Figure 8). The dynamic component is used for the menu (focus-menu) etc. around presented semantics-objects e.g. instances or relations, the place for the menu is differently – it depends on the used layout-algorithm and where the semantic object is placed on the visualization-layer. To regard the aspect of the situate clearness (see Dahm 2005) we are using a pie-menu in the focus of the interaction, during the editing within the semantics data. A pie-menu is useful, because it allows a fast access to needed functions and so it is useful if a fast work-process is desired (Hopkins 2003). The concept for the pie-menu is presented in Figure 9. The static components are components, which are placed on a fix position. The static components are used for the editing of semantic-objects e.g. changing values of literals.



Instance: Visualisierung_Class54

Instance of: &rdf_;Basistechnologie

name: Evaluation

label: Evaluation

about: &rdf_;Visualisierung_Class54

icon: icons/Intanz_Basistechnologie.png

Figure 8. The visual Semantic-Editing components on a visualization example

To achieve, that domain-experts does not have to know something about ontologies or ontology-languages, the data will be presented on an abstract level. The visualizations from the SemaVis-Framework show the semantic data on an aspect-dependent and easy to understand way. The semantic specific information like the concepts of the visualized instances are shown as sufficient information, to categorize the visualized information. During the editing-process the users get only tools, which are needed in the actual context. If the user wants to change the information of an instance, he will get only tools, which allow him to create a new instance from same concept, change literals from an instance, delete the instance and to create a new relation between another instance. Furthermore interaction metaphors are implemented, which are known from modern operating systems, like drag&drop.

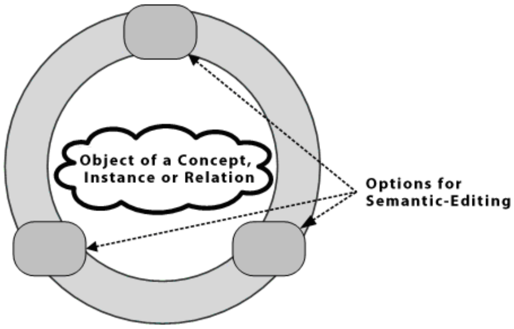

Object of a Concept, Instance or Relation
Options for Semantic-Editing

Figure 9. Visual Concept for the Focus-Menu around visual semantic objects

The Editing-Process with SemaVis-Editing

In the concept of the SemaVis-Editing we regard the aspects of the previous defined semantic-editing workflow. For realizing these aspects, we only provide tools, which are relevant in the actual phase. This helps the users, because the cognitive load will reduced, so that the users can focus on the actual task.

In phase 1, the users only see the static component next to the visualizations, in which he will later able to change data. In this step this component is empty, so that the user cannot be confused. In this phase only the SemaVis-Framework is active with its visualizations.

Up to phase 2 the editing-process, where the goal of changing information will achieve, "really" started. The user has to select the semantic object e.g. an instance or relation. In our prototypical implementation the selections will achieved by a mouse double-click on the object. After that, the user get a pie-menu with the available functions, he can do with this semantic object e.g. deletion of an instance or editing of an instance. The pie-menu is a dynamic component, which is shown around the semantic object. This is an efficient approach, because the available objects are next to object and the actual mouse-position, so that within a short time the function can be choosen.

In phase 3, the users can set information, especially the literals of existing instances. The information which can be changed is presented in the static component. If the user wanted to delete an object, he only gets a request, if he really wants to remove the object. The editable elements, like the literals, were presented on an abstract form. Currently we only reduce the literal name from the property-URI to the name of the property (e.g. by a defined label), without the URI of the instance. By accepting the changes, this phase is done.

The last phase is phase 4 where the changed information, which are still in an abstract form, were converted to an ontology conform syntax. Conversion does also include a casting in ontology specific data formats like string or integer. After the conversion, the changes are stored in the ontology. If this phase is done, the editing-process finished.

Results of the SemaVis-Editing-Framework

In a first prototype we implement all the told aspects. The two layers, the bottom one for the visualizations and the top one for the editing components, are working completely independent from each other. This means, that in principle the editing-layer will work correctly with arbitrary visualizations. This aspect was achieved because of the global event-broker, which is part of the SemaVis-Framework. This event-broker delivers status-messages from the visualizations, like which semantic object was selected with the mouse and so on.

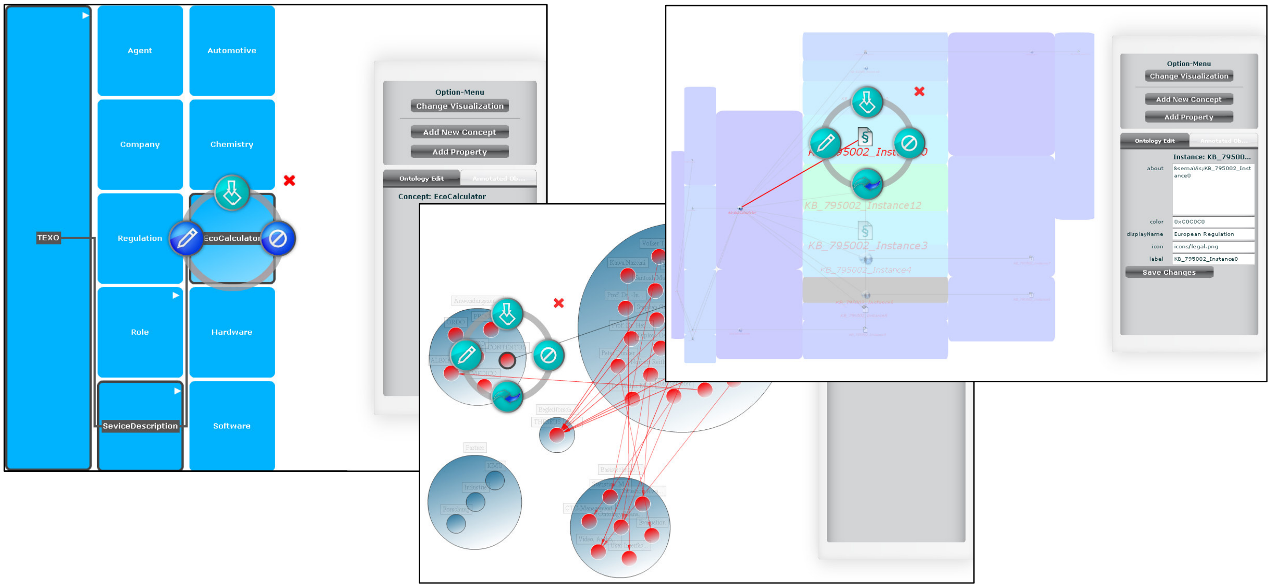
In the actual version, the following features are possible:

- Create instances from a selected concept
- Delete instances
- Change literals (and also the name) of instances
- Move existing instances into other concepts

- Create relations between instances
- Change the type (name) of relations
- Delete relations

Also implemented, but not designed for domain-experts, are functions which allow creating and changing of ontology-schemata. But for these functions it is essential to have knowledge and experiences in creating ontology-structures!

- Create new sub-concepts
- Change names of existing concepts
- Move existing concepts into other concepts
- Delete concepts



Instance: KB_79500...

about: &semaVis;KB_795002_Instance0

color: 0xC0C0C0

displayName: European Regulation

icon: icons/legal.png

label: KB_795002_Instance0

Figure 10. Example Screenshots of the SemaVis-Editing-Framework (the Editing-Menus on different Semantic-Visualizations – Visualizations from left to right: SeMap, SemaSpace, SemaZoom)

In Figure 10 some Screenshots are presented, where the editing-components are integrated and used. Because of the flexibility of the SemaVis-Framework, also further visualization can be integrated, which will also be supported by the editing-components, if they will use the event-broker and commit their interaction status.

Evaluation

The Semantic Editing is currently under further development and functionalities will extended. At this time, we only made a small evaluation study, to get a first impression, if the concept brings a benefit to the users. An end-user acceptance test will be made to later times.

To evaluate the conceptualized and prototypical implemented editor, we use the evaluation method of Performance Measures (see Nielsen 1993). We have two groups, in the first group are the developers of the Semantic-Editing, who have a lot of experience in working with this editor and the other group consist of users, who have less or never worked with this editor and are non-ontology experts. Before the evaluation starts, every user gets the possibility to try the editor and to understand how it works.

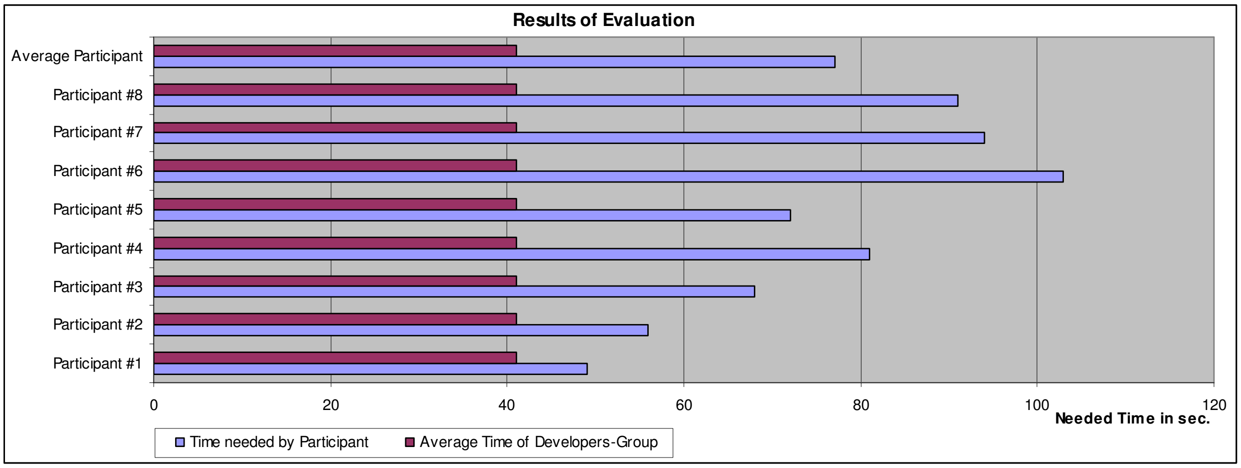
In the evaluation the users get task, which they have to solve. The tasks are to correct and to complete the existing data. Every user gets the same tasks. At the end, the average time for solving all tasks of both groups will be compared. As data we take an example ontology, which contains semantics-data about the THESEUS research program (see THESEUS-program 2006). The tasks every user has to do are:

1. Create a new Instance within a specific concept and give a name.
2. Change the name to another one.
3. Add a new specific Literal.
4. Create a new Relation from the added Instance to another given Instance from another Concept.

The evaluation was finished with good results. In average, the users only need 36 seconds (less than twice of average-time of the developers-group) more to complete the given tasks than the developers-group (reference-group), who are experienced ontology-experts, in average need. Overall the usability can be summed as good, especially because every user understand how to handle these tools und they know the sense of what and why they should do the different tasks.

Developers	
	Needed Time in sec.
Developer #1	38
Developer #2	43
Average	**41 (Reference time)**

Normal Users		
	Needed Time in sec.	*Rel. to Reference-time*
Participant #1	49	+20%
Participant #2	56	+37%
Participant #3	68	+66%
Participant #4	81	+98%
Participant #5	72	+76%
Participant #6	103	+151%
Participant #7	94	+129%
Participant #8	91	+121%
Average	**77**	**+87%**



Discussion and Future Work

In this prototypical implemented system, the literals are only simplified by reducing the property-URI only to the property name, so that the base, the instance-URI, will be hidden. But for more complex data and especially if also values from other types than strings will be allowed, another solution is necessary. In further work it is command to develop a general valid approach, to achieve literals as abstract as possible.

Another aspect is to offer an active support during the semantic-editing process. An innovative approach is the combination of automatic methods for generating semantics and manual acceptations of the domain-expert. This approach helps domain-expert to regard further possible relations, which the expert does not have in mind. This is a challenge on which we work in the next time.

Conclusion

In this paper we described a method, how needs of domain-experts can be regarded to allow them to commit their domain-expertise into ontologies on an intuitive way. This is important, to create a knowledge-base, which can more efficiently used to explore information and finding needed information within a short time. The described method primary concerns on the design of the graphical user interface, which allows an abstract form of information visualization and an abstract form to edit existing information. It is also possible to enrich the ontology by entering new information on an abstract form too, which is more intuitive to use for ontology-novices than established ontology-based editors, which are designed for ontology-experts.

References

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

Dahm, Markus (2005). Grundlagen der Mensch-Computer-Interaktion. Pearson Studium.

Davies, J., Fensel, D. and Harmelen, F. van. (2003). Towards the Semantic Web, John Wiley & Sons.

Hopkins, Don (2003). Why Pie Menus Aren't Ubiquitous. Infovis.net. http://www.infovis.net/printMag.php?num=125&lang=2.

Katifori, A., Halatsis, C., Lepouras, G., Vassilakis, C., and Giannopoulou, E. (2007). Ontology visualization methods—a survey. *ACM Comput. Surv.* Vol. 39, 4.

Liebig T. and Noppens, O. (2004). OntoTrack: Combining Browsing and Editing with Reasoning and Explaining for OWL Lite Ontologies,. In Proceedings of the 3rd International Semantic Web Conference ISWC 2004, Hiroshima, Japan.

Nazemi, K., Breyer, M., Hornung, C. (2009): SeMap: A Concept for the Visualization of Semantics as Maps In: Proceedings of HCI-International 2009.

Nazemi, K., Burkhardt, D., Hofmann, C., Breyer, M., Stab, C. (2009): Deliverable: D.CTC.5.7.1: Design und Conceptualization Semantic Visualization Framework, THESEUS-Program

Nielsen, J. (1993): Usability Engineering. Morgan Kaufmann, San Francisco.

Noy, N. F., Fergerson, R. W., and Musen, M. A. (2000). The Knowledge Model of Protégé-2000: Combining Interoperability and Flexibility. In Proceedings of the 12th European Workshop on Knowledge Acquisition, Modeling and Management. R. Dieng and O. Corby, Eds. Lecture Notes In Computer Science, vol. 1937. pp. 17-32, Springer-Verlag, London.

OilEd (2003). http://oiled.man.ac.uk/.

Plaisant, C., Grosjean, J., Bederson, B. B. (2002). SpaceTree: Supporting Exploration in Large Node Link Tree, Design Evolution and Empirical Evaluation. In Proceedings of IEEE Symposium on Information Visualization, pp. 57-64, Boston.

SWOOP (2004). Semantic Web Ontology Overview and Perual. http://www.mindswap.org/2004/SWOOP/.

THESEUS-program (2006). http://theseus-programm.de.