OWL Visual Notation and Editor

Taglout Ramdane  
Department of Computer Science and ICTS  
University Kasdi Merbah-Ouargla  
Ouargla, Algeria  
tagloute.ramdane@gmail.com

Said Bachir  
Department of Computer Science and ICTS  
University Kasdi Merbah-Ouargla  
Ouargla, Algeria  
saidbachir@gmail.com

Abstract—In this paper a new visual notation of OWL-DL language is proposed. It is inspired from several studied notations and editors. The proposed notation is suitable for ontology visualization and more adapted for ontology visual modeling. OVE is an OWL visual editor under developed. It is based on the proposed notation of OWL-DL.

Keywords-Ontology; OWL; OVE; Visual Notation; Visual Editor

I. INTRODUCTION

For human, the use of images is still a simple way for expression and communication. Human often learn in his life through the images and think with forming images in his mind. Researches on visual programming consider the human visual perception to design systems and simplicity programming for no-specialist without the need to be expert in the techniques classical programming.

In visual programming language (VPL), there are different methods that allow users to programming visually, N.C. Shu [1] classified them in three categories: diagram based, iconic and form based languages.

Recently, many visual languages are defined in several fields. Among these fields, we find the Semantic Web. Necessary to use it in this field could be owing to the difficulty in the manipulation and the use of its languages, such as OWL (ontology web language) [2]. In several works, ontology visual tools are proposed, but the problem which remains posed, is that they are difficult to use for no-ontologist. In addition, there are some visual editors based on simple representation of OWL ontologies, but they take a long time on ontology development.

Moreover, Lack of standard visual notification to different elements of OWL (classes, individuals, properties, and their relationships).

The purpose of this work consists in two parts:

- Provide a visual representation of the ontology web language (OWL).
- Implementing a visual ontology editor simpler and easier to use in the creation and the manipulation of ontologies by expert and non-expert users, which based on the proposed notification.

In this work, we have combined diagram based, and form based representation to benefits advanced of each one and more effectiveness.

In the following, we first summarize related work in Section 2. In Section 3, we have presentation of the proposed visual notation for each OWL elements (class, property, relationship, restriction...). In Section 4, we experiment the editor OVE by different example of ontology and compare them with other tools in creation time. Finally, a conclusion and prospects on future work.

II. RELATED WORKS

Ontologies are used in several areas due to its ability to describe and represent a domain of knowledge. Recently has taken several efforts for visualization of ontology to simplicity creation and deal with it.

Among the first works that give a visual representation of OWL ontologies using UML we find [3], but cannot be sufficient.

So developers defined ODM Ontology Definition Metatmodel [4] to create mappings between standard UML metamodel and RDFS and metamodels OWL, thus creating method to use the current models UML in Ontology development. Each component of the model of ODM is annotated using the intrinsic property of UML stereotype.

After that appeared several notations of visualization OWL with different forms, for instance we have VOWL notation created by Stefan Negru and Steffen Lohmann [5].

The notation is based on four types of graphical elements:

- classes = circles (size indicates the connectivity of a class);
- individuals = sections in the circles;
- Properties = lines/arcs (with different arrow heads, line types, etc.);
- Literals, data types and values = rectangles.

Before speaking in detail about our work, we have a review to various ontology visualization tools existing.
The popular one is Protégé-OWL 1 was developed at Stanford University. What sets it apart is its used to explorer-tree view of different elements of ontology. The positive thing is the simplicity and ease of handling, but a major drawback is the absence of a detailed vision of the relations between elements.

OWLGrEd [6] is a free UML style graphical editor for OWL ontologies. It has additional features for graphical ontology exploration and development. Obstacle that finds when use this tool is difficult to distinguish between elements (concepts, properties, and instances) as they had similar graphical representations.

OWLVis2 has the ability to give the user an overview and it is due to used of node-link and tree methods representations. It has a negative point is difficult to display in the cases when ontology be large.

The conclusion is that there is not a method that is always the best. But many of the ontology tools did not take into account the no-ontologist users for its establishment of their visualization, so we tried in this work to create a visualization based on simplicity and the overall vision, as we put all element of OWL ontology in one location in order to avoid wasting time user to move from one process to another.

III. VISUAL NOTATION OF OWL ONTOLOGIES

In this section, we present a visual notation of OWL elements. The proposed notation is designed to be used in ontology visual editor. Some elements of our notation are inspired from many visual representations and visual editors of OWL such as protégé-OWL, VOWL, GrOWL [7], OWLGrEd …etc. The other elements of our notation such as (extended class, individual, property, extended expression …), are designed to decrease the disadvantages noticed on the other visual representations and visual editors.

A. Representation of OWL Elements

In TABLE I we present the visual representation of OWL-DL elements. OWL-DL is the sublanguage most used to describe ontologies.

<table>
<thead>
<tr>
<th>OWL Element</th>
<th>Visual Notation</th>
</tr>
</thead>
<tbody>
<tr>
<td>owl:Class</td>
<td>Class Name</td>
</tr>
<tr>
<td>owl:DatatypeProperty</td>
<td></td>
</tr>
<tr>
<td>owl:Individual</td>
<td></td>
</tr>
</tbody>
</table>

1 http://protege.stanford.edu/
2 http://protegewiki.stanford.edu/wiki/OWLVis
In this table, some elements have two representations (compact and extend), such as classes, object properties and expressions. The compact representation is shown as black box, while the extended representation shows all content of OWL elements. For example, in the extended form of OWL Class we can view annotations, datatype properties and individuals.

Each datatype property is defined by its name, type and multiplicity (Func: functional / Mult: multiple). Datatype property can be also characterized by a list of restrictions that are represented as mono-level tree.

Object property is represented as oriented link and a blue rectangle in the middle. Into the blue rectangle, all parts of the object property can be defined: characteristics, annotations and restrictions. Characteristics means (Func: Functionnal, InvF: Invers Functionnal, Sym: Symetric and Trans: Transitive).

The relations rdfs:subClassOf, owl:equivalentClass and owl:disjointWith connect two classes, or a class with a list of expressions. In our visual notation, object properties can be connected by three types of relation: rdfs:subPropertyOf, owl:equivalentProperty and owl:inverseOf.

A list of expressions is described by a rounded rectangle. Its extended representation shows a set of expressions related to the class with which it is connected. The visual notation describes the simple expressions as well as the complex ones. The simple expression is represented by mono-level tree, while the complex one is represented by multi-level tree. Figure 1 shows an example of complex expression related to Class_1 by owl:equivalentClass relation.

Example:

Class_1 = Class_2 or (Class_3 and Class_4) or not Class_5

Restrictions determine exact, min and max cardinality of object property. They can also determine the value(s) of object property range (owl:hasValue, owl:allValuesFrom and owl:someValuesFrom).

In Figure 2, we present (with our visual notation) a part of family ontology that describes family members and relationships.

B. Novelties and Advantages

Unlike the extent visual representations of OWL, our representation is designed to be used in a visual editor of OWL ontologies. It ensures simple visualization, fast access and easy manipulation to ontology elements.

As is shown in TABLE I, some elements are designed to be extendable such as (class, expression, and property). This idea aims to reduce the space used to visualize and edit ontology elements. So, initially the user sees each element of the ontology as simple box. Once he desires edit an element or view its details, he has to extend it. For example, we used a form that includes lists and trees, to show all details of an extended class. But, in the other visual notations, a class is represented by a rectangle, labeled by its name. Each datatype property or individual must be shown as separated entity. That requires more space and causes a low access to ontology elements.
Other novelty, object property is a form that links two classes. The form includes all details of object property (characteristics, annotations and restrictions). In the other notations, object property is defined by oriented arrow that is labeled with some characteristics. The advantage of our representation is the ability to define visually direct relations between object properties (\texttt{rdfs:subPropertyOf}, \texttt{owl:inversOf} and \texttt{owl:equivalentProperty}). Another advantage is the possibility of defining restrictions visually into the object property form.

A list of expressions is a rounded rectangle linked with OWL class by (\texttt{rdfs:subClassOf}, \texttt{owl:equivalentClass} or \texttt{owl:disjointWith}) relation. An expression is represented by vertical multi-level tree that can be developed and reduced. Moreover, this representation reduces the number of used operators. For example, if a sub-expression contains two or more union operators, we can use one union node linked to its children (terms). But, in some visual notations, each expression is represented separately by a binary tree that requires more operators and needs more workspace.

In visual classes, individuals are listed without viewing their details. Once an individual is selected, all its datatypes and object properties values are viewed in an independent form.

Generally, our visual notation is intended, not only for the visualization of ontologies, but also for the use in ontology visual editor. It is a combination between the diagram based tools (OWLGrEd, GroOWL, and Visual OntologyModeler) and the form based tools (protégé-OWL, TopBraid Compozer). Diagrams provide a global view of ontology and show all relations between its elements. Forms allow rapid access and determination of details of any element.

IV. IMPLEMENTATION OF ONTOLOGY VISUAL EDITOR

In this section we present the prototype of Ontology Visual Editor (OVE). This editor is based on OWL visual notation proposed in this paper. It is still under development, so, all components are developed except the expression component.

A. Presentation of OVE

OVE is developed on Java using JDK 1.6 and Eclipse environment. We also used the API of Protégé-OWL to develop OWL ontologies in the Java code. JGraphx is a Java based graph library; it is used to develop diagrams.

OVE ontologies have two layers (see Figure 3):

- OWL layer: its elements are based on Protégé-OWL API components.
- Visual layer: its elements are based on JavaBean and JGraphx components.

![Figure 3. Example of a figure caption](http://www.topquadrant.com/products/TB_Composer.html)
The main window of OVE is presented in Figure 4. To add new relation from a class, the user must choose the type of relation and drag the mouse from the center the first class to the second class.

In OVE, classes and object properties contain the majority of Java Swing components. For this reason, the editor became powerful and easy to use.

**B. Experimentation**

The objective of this experimentation is to compare OVE with some popular ontology editors, on the times of creation. For this experimentation we chose Protégé-OWL, OWL-VizMod and OWLGrEd editors.

The four editors are used to create three types of ontologies (TABLE II). Onto1 is a simple ontology, Onto2 is medium and Onto3 is greater.

<table>
<thead>
<tr>
<th>Ontology</th>
<th>Classes</th>
<th>Object properties</th>
<th>Datatype properties</th>
<th>Individuals</th>
</tr>
</thead>
<tbody>
<tr>
<td>Onto1</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>9</td>
</tr>
<tr>
<td>Onto2</td>
<td>5</td>
<td>3</td>
<td>2</td>
<td>21</td>
</tr>
<tr>
<td>Onto3</td>
<td>14</td>
<td>20</td>
<td>20</td>
<td>30</td>
</tr>
</tbody>
</table>

**C. Results**

The results shown in Figure 5, present the time calculated during the creation of each ontology by the four editors.

**D. Discussion**

Figure 5 shows the time calculated for each ontology by Protégé-OWL, OWL-VizMod, OWLGrEd and OVE. The presented results demonstrate that ontologies are created in less time by OVE and Protégé-OWL. So, OVE keeps the rapidity advantage of Protégé-OWL. In the same time OVE acquires the visualization advantages of the other editors.

After the use of many ontology editors, we noticed that ontology editors with good visualization are more difficult and take more time. But OVE provides good visualization with rapid access, easy use and less creation time.
V. CONCLUSION AND FUTURE WORK

In conclusion, in this paper we proposed an OWL visual notation. This notation is suitable for ontology creation and visualization. So, we developed OVE that is based on the proposed notation. To get advantages from many popular ontology editors, we studied their characteristics. We distinguished three families of editors: form based (Protégé-OWL, TopBraid Composer …), diagram based (HOZO, OWLGrEd, GrOWL …) and surface based (OWL-VizMod).

OVE is a combination between form based and diagram based editors. With this combination OVE has got advantages of the two categories of editors. So, OVE provides good visualization, rapid access and less time of creation. In this paper, the time was the only criterion of evaluation. But there is other criteria of evaluation, that can be employed in the comparison between different editors. For example we can interrogate user about simplicity, interactivity and accessibility.

In future work, we will complete the development of OVE and we will test some editors with many types of ontology, taking into consideration users’ opinion.

REFERENCES