

# On Visualization of OWL Ontologies

Serguei Krivov<sup>1,3\*</sup>, Ferdinando Villa<sup>2,3</sup>, Richard Williams<sup>4</sup>, Xindong Wu<sup>1</sup>

<sup>1</sup>Department of Computer Science, <sup>2</sup>The Botany Department,

<sup>3</sup>Gund Institute for Ecological Economics,

The University of Vermont;

<sup>4</sup>Rocky Mountain Biological Laboratory.

\*Corresponding author: E-mail: skrivov@uvm.edu Phone:(802)656-0380; FAX:(802)656-2995

Address: Gund Institute for Ecological Economics, 617 Main Street, Burlington, VT. 05405-0088 USA

July 28, 2006

## Abstract

Ontology visualization tools serve the expanding needs of knowledge engineering communities. A number of visualization frameworks for the standard ontology language OWL have already been developed. Considering information visualization in general, we propose the criteria of simplicity and completeness with which to gauge ontology visualization models. After analyzing existing OWL visualization frameworks we propose a simple visualization model for OWL-DL that is optimized according to our criteria. This visualization model is based around the underlying DL semantics of OWL ontologies; it circumvents the perplexities of RDF syntax. It has been implemented in GrOWL- graphical browser and editor for OWL ontologies. Apart from visualization model we discuss the role of visualization techniques in the interaction of user with ontologies. In the end we discuss the usage of GrOWL in Ecosystem Services Database.

**Keywords:** OWL, GrOWL, Ontology visualization, Ontology editing, Semantic networks

# 1 Introduction

Ontologies [1] are specifications of conceptualization that facilitate the sharing of information between different agents. In many Semantic Web (SW) projects, ontologies are also set to play the role of an interface between the user and the data. This increasing use of ontologies in the role of an interface makes the problem of ontology visualization highly relevant. Well designed visualization schemes and efficient visualization techniques are important for designing convenient user interfaces that provide means for browsing, editing and querying large ontologies. This chapter discusses the problem of ontology visualization, focusing on the standard ontology language OWL [2].

Ontology languages are primarily designed to represent information about categories of objects and how objects are interrelated. This is the sort of information that ontologies store. Ontology languages can also represent information about the objects themselves—this sort of information is often thought of as data. An ontology language must have a well-defined syntax, well-defined semantics, efficient reasoning support, sufficient expressive power, and convenience of expression. These requirements directed the evolution of a sequence of W3C recommendations and standards for ontology languages: RDF, then DAML+OIL [3], and now OWL [2].

OWL is a product of long evolution in Knowledge Representation techniques. The history and the evolution of ideas that led to the design of OWL are described in [4]. There are three versions, or species, of OWL. In the order of increasing expressiveness, OWL Lite was designed to support classification hierarchies and simple constraints. OWL DL is backed by a description logic formalism and so maximizes expressiveness while maintaining computational completeness and decidability of reasoning systems. Finally, OWL Full offers much greater expressive freedom at the expense of giving up the computational guarantees of OWL DL [2].

Various tools for visualization of OWL ontologies have been developed. In an effort to optimize visualization and editing of OWL ontologies we have developed a visual language for OWL-DL and implemented it in GrOWL, a visual editor and browser for OWL ontologies [5], [6]. The visual language referred here as *GrOWL visualization model* attempts to accurately visualize the underlying DL semantics of OWL ontologies, without exposing the complex OWL syntax. We intentionally limited our focus to OWL-DL, the most expressive species of OWL that is supported by reasoners. GrOWL has been implemented both as a stand alone application and as an applet. The applet version has been used in publicly available, semantically aware databases such as the Ecosystem Services Database [7],[8].

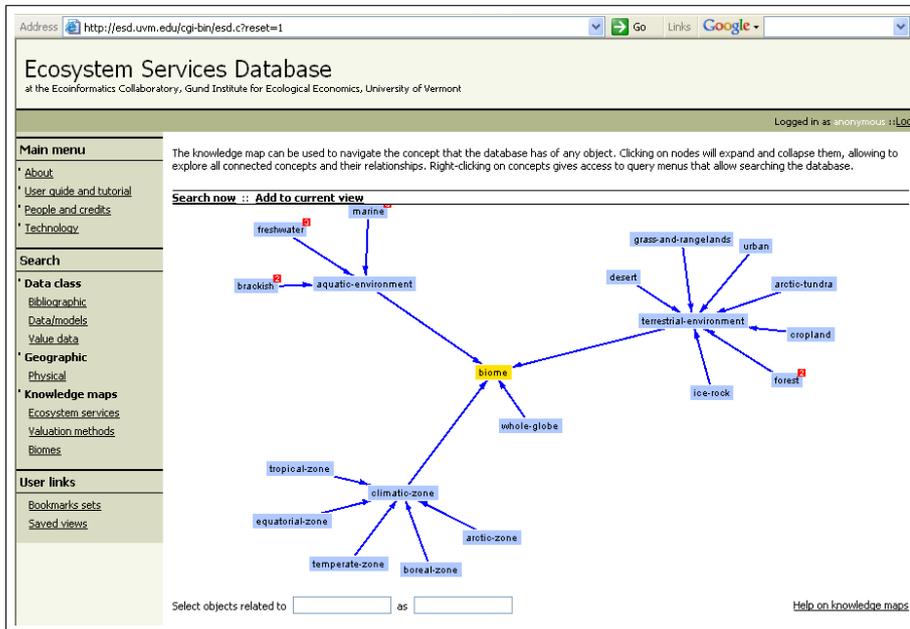


Fig. 1 GrOWL applet running in the Ecosystem Services Database, a web-based application, showing a simple biome ontology for browsing and query.

In this chapter, we discuss the GrOWL visualization model and related visualization techniques. In section 2, we define our performance criteria for OWL visualization models and tools, provide a brief review existing models and tools, and describe the design choices that led us to GrOWL. In section 3, we introduce the visualization model for OWL. In section 4 we describe the current implementation of GrOWL and its applications. The conclusion contains a summary of the results and a discussion of future work.

## 2 The Analysis of Visualization Models

Numerous examples of ontology visualization have been presented in the literature [9]. However the questions about visualization of ontologies were not yet considered from a theoretical perspective. This is not surprising, as information visualization is still emerging as a new field within computer science. A variety of visualization designs for both structured and unstructured data has been proposed [10]. Ontology visualization tools can also take advantage of well designed software libraries, such as Prefuse [11], that reflect a decade of experience in information visualization. These generic visualization techniques will be discussed in section 4, while in this section we consider those specifically related to ontologies.

Chen [10] argues that information visualization is in need of a generic theory that can help designers to evaluate and compare the visualization techniques. An interesting effort made in this direction is the work on semiotic morphism by Goguen [10];[12], who suggested a rule of thumb that allows the evaluation of the quality of a visualization. The quality could be measured by what is preserved and how it is preserved; it is more important to preserve structure than content when a trade-off is forced. We refer to this rule as *Goguen's criterion*. Our discussion

of visualization frameworks also adopts a *minimum complexity criterion* as an informal rule of thumb. Based on these criteria, we consider acceptably efficient a visualization model that can *provide a readable rendering of all or almost all elements of Roger L. Costello's camera ontology [13] and other similar-sized ontologies on a 640-by-800 pixel canvas.*

## 2.1 OWL Semantics

Before reviewing OWL visualization framework we briefly describe the underlying semantic structure that has to be visualized. The OWL formalism was designed to harness the power of sound, complete and decidable Description Logic (DL) systems. OWL-DL and OWL-Lite are two major species of OWL that have a clean DL semantics.

**OWL-DL knowledge base (KB)** DL languages are built up of two kinds of primitive symbols, *concepts* interpreted as unary predicate symbols and *roles*, interpreted as binary predicate symbols; the latter are used to express relationships between concepts. The concepts are of two kinds - atomic concepts and concept expressions. Concept expressions are formed using boolean operations on concepts and role restrictions. There are several types of role restrictions. For example, a concept written  $\exists hasChild.Male$  denotes all the individuals having at least one male child. The concept  $\forall hasChild.Male$  denotes the set of individuals whose children are all male. The concept  $Male \sqcap \exists hasChild.Parent$  may serve as a definition of the concept "grandfather" because it denotes the set of all male individuals who have at least one child who is also a parent. The concept  $\geq 6 hasChild.Person$  denotes a set of individuals who have at least 6 children and may serve as a definition of the concept "prodigious parent". The fundamental axiom between concepts is the subsumption relation (subclass-of relation) denoted by symbol  $\sqsubseteq$ , e.g.  $Human \sqsubseteq Animal$ . The fundamental relation between an individual and a class is the *instance - of* relation usually denoted by colon, e.g.  $John : Person$ . Concept expressions are built from concept expressions using several kinds of constructors such as intersection  $\sqcap$ , union  $\sqcup$ , complement  $\neg$ , "all values from" restriction  $\forall R.C$ , "exist value" restriction  $\exists R.C$ , cardinality restrictions  $\leq nR$  and  $\geq nR$  and qualified cardinality restrictions  $\leq nR.C$  and  $\geq nR.C$ .

A DL knowledge base could be represented as a set of statements of the form:  $C \sqsubseteq D$  ( $C$  is subclass of  $D$ ),  $a : C$  ( $a$  is an instance of  $C$ ),  $(a, b) : R$  (*role assertion* stating that  $b$  is a value of property  $R$  for individual  $a$ ); here  $C, D$  are concepts definitions and  $a, b$  individual names. The statements of the form  $C \sqsubseteq D$  are called terminological. The statements of the form  $a : C$  and  $(a, b) : R$  are called assertional. All terminological statements form *Tbox* of the knowledge base, while assertional statements form the *Abox*. Expressive description logics support role subsumption, or role hierarchies axioms  $R \sqsubseteq S$  ( $R$  is subrole of  $S$ ). Different systems of DL are formed by selecting different sets of class constructors and axiom types.

OWL-DL is reducible to logic *SHOIN(D)* [14] that allows boolean operations on concept expression, existential and universal role restrictions, cardinality restrictions, role hierarchies, transitively closed roles, inverse and functional role axioms and concrete domains. To maintain compatibility with earlier SW standards, OWL is encoded within the Resource Definition Framework (RDF) syntax. In the case of OWL-DL, the way from the RDF-based syntax of OWL to its DL semantics is very hard and thorny (see [4]). To hide the complexity of RDF syntax and to simplify the presentation of OWL-DL ontologies and make them readable to human, OWL Abstract Syntax was created. This syntax closely follows the DL semantics of OWL.

There are many possible avenues for visualizing an OWL knowledge base. Oftentimes, the hierarchy of named classes serves as a base of visualization. However, since there is multiple inheritance, the translation of the class hierarchy into a tree is not straightforward. In addition, focusing on named classes doesn't address the representation of boolean operations and property restrictions. They represent essential structures of OWL ontology and therefore according to

*Goguen's criterion* it is essential to represent them in the graph as well. Moreover, it is essential to provide visualization methods for both the TBox and the ABox. We suggest that OWL visualization tools should support at least separate views of class definitions, the named class hierarchy, and the whole ontology. The following subsection analyzes existing approaches to visualization. It appears that most of them realize the essential goals of visualization only partially.

## 2.2 OWL Visualization Models

There are several implemented and designed visualization frameworks that are used or could be potentially used with OWL. To simplify our analysis we divided them into 5 categories.

**1. Tree-based visualization models** Some visualization frameworks were designed mainly for navigating an ontology's class hierarchy. Protege [15] plug-in OWLViz [16] is an example that belongs to this category. Several tools of this kind were described in [9]. Although such tools are useful, they do not represent visually all the elements of ontology's meaning and would not get high score according to *Goguen's criterion*.

**2. Table-based visualization models** Some visualization frameworks have been inspired by UML modeling of object oriented languages. The ezOWL [17] tool is a visualization and editing tool that provides a table based rendering of OWL ontologies, where classes are described as tables of properties. The OntoTrack tool [18] also belongs to this category. OntoTrack uses sophisticated layout and navigation techniques to optimize browsing and editing large ontologies, however at the moment it supports only a subset of OWL-Lite. The main issue with the table based frameworks is the redundancy in representation of properties, since they are listed for a class and all its subclasses and as a result the visualization process requires a lot of space on the canvas. This wasted canvas space is especially problematic when visualizing large ontologies. Nevertheless, tables of properties are very useful, since they clearly describe what properties a class must have or can have. We decided that it is more appropriate to use them as a secondary visualization aid, and so in GrOWL, the tables of properties are presented in special window only for the currently selected class. We consider this to be a better use of table based visualization.

**3. RDF-based visualization models** Since OWL is expressed in XML/RDF syntax, the frameworks for RDF visualization theoretically may be used for OWL ontologies. There are several frameworks that are based around visualization of RDF graphs. One example of such a framework is IsaViz [19]. Its strong point is support for graph stylesheets that allow the user to modify the way the graph is presented. This tool uses AT&T graphviz/dot program for making the layout and it is unlikely that it can compete with tools that use modern graph layout libraries. Perhaps the most interesting RDF visualization framework is RDF-Gravity [20]. This tool has an advanced filter mechanism. Filters allow a user to hide or view specific edges based on type or to hide or view specific particular instances of nodes or edges. RDF-Gravity has a query backend and allows the generation of views from queries.

Both IsaViz and RDF-Gravity render literally the RDF structure of the file without honoring the OWL specific constructs. They inherit the verbosity of RDF and as a consequence OWL ontologies are difficult to read in these tools. Although developers could learn a lot from RDF visualization tools such as RDF-Gravity, these tools are not particularly suitable for the rendering of OWL ontologies.

**4. UML-based visualization models** VisioOWL [21] is an MS Visio based visualization tool for OWL. There are efforts to build standard UML based presentation for OWL. A metamodel for the purpose of defining ontologies, called Ontology Definition Metamodel (ODM), has recently been requested by the Object Management Group [22], with specific focus on the OWL-DL language. To answer this request several proposals were submitted. Although some

of these metamodels have special symbols for OWL elements and unlike RDF visualizers provide a readable presentation of OWL ontologies, they are still not optimized compared to the metamodel presented in [23] that comes very close to the graphic mapping presented here.

The intention behind UML based OWL visualizers is to reuse the power of already developed advanced UML editors, such as Visio. Although this works well for the users who already have UML editors, it may be not so attractive for those who don't. Good UML editors are often very expensive, they have features that may be not useful for OWL visualization and do not support the ones that are needed. For example, experimenting with GrOWL we found that it is often convenient to use the visual representation along with traditional navigation methods, such as the class hierarchy tree. Adding such features to a UML editor may not be easy. There are some important elements of ontology managements which are not easy to incorporate into commercial UML editors. These include the connection to a DL reasoner or a database backend, and query support. Although UML editors have advanced layout engines, none of them seems to support dynamic layout that allows recentering the graph on the fly, showing only a specific locality of a selected node (e.g. a class definition).

**5. DL-based visualization frameworks** An OWL visualization can try to accurately visualize the XML/RDFS syntax of OWL ontology as VisioOWL does. However, it is also possible to center the visualization around the OWL Abstract Syntax or DL semantics of OWL. The difference in clarity of these two types of visualization is analogous to the difference between XML/RDFS syntax of OWL and OWL Abstract Syntax. Since the first one is extremely verbose and the later was designed specifically for presentation, centering visualization around OWL Abstract Syntax has clear advantages and has far better performance according to the *minimum complexity criterion*. The advantage comes at the price of generality; DL based visualization frameworks can only support the DL based semantics of OWL-Lite and OWL-DL. Although exclusion of OWL-Full may be somewhat regrettable, it can be justified since applications of OWL-Full are not common.

There is an intimate connection between semantic networks and DLs. In fact DLs were invented in an effort to provide precise semantics for semantic networks[24]. There is an earlier proposal for visual notations based around DL CLASSIC [25]. Although DLs have been used in Semantic Web languages, there are surprisingly few tools that belong to this category, and apparently GrOWL is the only one. Out of the UML based models only [23] is likely to belong here, however it has not been yet materialized in any implementation.

### 3 A Visualization Model for OWL

The OWL visualization model presented here targets all essential components of an OWL knowledge base. Both the TBox and ABox parts are represented in graphical form. The TBox model represents properties, property restrictions and boolean operations.

**Graphical Coding** From Conceptual Graphs [26], [27] we learned to use shapes to clearly differentiate between logical categories. The first principle behind GrOWL visualization model is the use of the color, shading and shape of nodes to encode properties of the basic language constructs. Table 1 describes the graphical coding scheme of GrOWL visualization model.

**ABox Mapping** We assume that the reader is familiar with DL semantics and Abstract Syntax of OWL [28]. We also assume familiarity with basic DL terminology.

Consider the following ABox:

```
Individual(JohnSmith  
  type(academicStaffMember)
```

Table 1: Graphical Coding Scheme of SWVL-OWL

Node shape	Color 1 (Blue)	Color 2 (Brown)
Rectangular with shaded background	Classes	Data types
Rectangular with white background	Individuals	Data values
Oval with shaded background	Properties and Property Restrictions	Data Properties and Data Property Restrictions
Oval with white background	Property Value pairs, Value Restrictions	Data Property Value Pairs, Data Value Restrictions

```
value(teaches Java)
value(teaches ArtificialIntelligence)
value(age "32"^^xsd:integer)
```

The graphical mapping of this ABox is shown in Fig. 2.

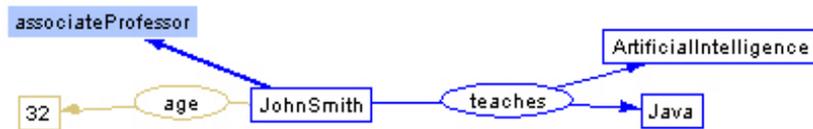


Fig.2. Graphic idioms for assertions about individuals.

The following two diagrams provide simple examples that illustrate idioms for axioms `SameIndividual` and `DifferentIndividuals`.

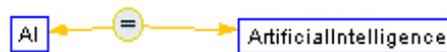


Fig. 3. Representation of axiom  
`SameIndividual(AI  
 ArtificialIntelligence)`



Fig.4.Representation of axiom  
`DifferentIndividuals(JohnWSmith  
 JohnSmith)`

**TBox Mapping** The TBox mapping in GrOWL visualization model was inspired by the domain maps introduced in [29]. The mapping is defined by two mutually recursive functions. The first function is the structural mapping  $G$  that maps every definition of class  $C$  from the OWL knowledge base into a graph  $G(C)$ . Only one node of the graph  $G(C)$  represents the mapped class  $C$ . Thus, the second function is the base node mapping  $BN$  that draws correspondence between a class definition and the single node  $BN(C)$  of graph  $G(C)$ . The node  $BN(C)$  (base node of  $C$ ) represents class  $C$  in that sense that every arrow representing subclass-of and equal-to relations of a class is attached to the base node of this class. Thus, for any pair of either named or anonymous classes  $C1$  and  $C2$  the following holds:

$G$  maps  $C1 \sqsubseteq C2$  ( $C1$  is subclass of  $C2$ ) to the diagram in Fig. 5.



Fig.5. Graph  $G(C1 \sqsubseteq C2)$

$G$  maps  $C1 = C2$  (which is equivalent to  $C1 \sqsubseteq C2$  and  $C2 \sqsubseteq C1$ ) to the diagram in Fig. 6.



Fig.6. Graph  $G(C1 = C2)$

$G$  maps ABox expression  $a : C1$  ( $a$  is instance of  $C1$ ) to the diagram in Fig. 7.



Fig.7. Graph  $G(a : C1)$

In the Figures 5-7 and in Table 2, the node labeled as  $BN(C1)$  is a base node of class  $C1$ , and the node labeled  $BN(C2)$  is a base node of class  $C2$ . They do not have to be shaded squares, but may be of any shape permissible for base nodes shown in the third column in the Table 2. Table 2 describes the recursive mapping of OWL class constructors into a graph that constitutes the core of GrOWL visualization model. This mapping is created by functions  $G$  and  $BN$  described in second and the third column of the table respectively. Data properties and data property restrictions are not shown in the table since their mapping is identical to the mapping of respective relations pertaining to individuals, and the difference is only in color.

Table 2: Recursive mapping of DL class constructors.

Definition of class $C$	The diagram $G(C)$	Base node $BN(C)$
Named Class $C$		$C$
Intersection $C_1 \sqcap C_2$		$\sqcap$
Union $C_1 \sqcup C_2$		$\sqcup$
Complement $\neg C_1$		$\neg$
Enumeration $\{o_1, o_2\}$		$\{E\}$
Exist Restriction $\exists R.C_1$		$\exists R$
For all Restriction $\forall R.C_1$		$\forall R$
Number Restriction $\geq nR$	Eg. 	$\geq nR$
Number Restriction $\leq nR$	Eg. 	$\leq nR$
Value Restriction $R:o$		$R$

Figure 6 describes the mapping of “subclass of” axiom; the mapping of the remaining OWL class axioms is described in the Fig. 8.



Fig.8. Mapping of OWL class axioms  $\text{EquivalentClasses}(C_1 C_2 C_3)$  and  $\text{DisjointClasses}(C_1 C_2 C_3)$

Structural mapping follows the recursive definitions of the OWL semantic constructs, and therefore every TBox construct receives a graphic representation. Structural mapping is one-to-one. Although the rendering of the arrows representing subclass-of and instance-of axiom is identical, one cannot be mistaken for another since subclass-of arrow always connect two classes.

**Properties** Most of the GrOWL visualization model idioms could be obtained from the structural mapping of a representation of an OWL file as a DL knowledge base. Property declarations constitute an exception. The DL semantics of property declarations is complex and therefore we have introduced special idioms for this case (see Figure 9). The introduction of special symbols as a substitute for complex graphs that express the DL semantics of property declarations does not break the unambiguous character of the GrOWL visualization model mapping.

A property is a binary relation. Fig. 9 shows a graphic idiom for a simple specification of an object property: The global restrictions on properties, such as specifications that a property is symmetric, functional, transitive, or any allowed combinations of these, are provided on a separate property pane. The relations of properties to other properties such as subproperty and `inverseOf` relations are depicted graphically in a separate RBox view that displays the property hierarchy. For example, consider the following property specification:

```
ObjectProperty( teaches super(involvedIn)
```

```

domain(academicStaffMember)
range(course)
inverseOf(isTaughtBy))

```

This specification of an object property will generate two separate diagrams as shown in Fig. 9.

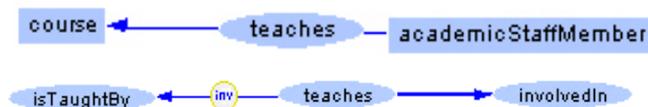


Fig.9. The separate diagrams generated by the object property specification.

The second diagram in Figure 9 appears in the RBox view only. The separation of TBox, ABox and RBox views is especially convenient for viewing large ontologies. Note that the subproperty relation could be specified with a subproperty axiom separately from the property declaration. For example, the relation between *teaches* and *involvedIn* could be specified with the subproperty axiom *SubProperty(teaches, involvedIn)*. The idiom for datatype properties is analogous to idiom for object properties, except that it is rendered in a different color.

## 4 Current implementations and applications of GrOWL

The current prototype of GrOWL is based on the Prefuse library[11], which supports a wide variety of layout algorithms, most of which are used in the GrOWL implementation. GrOWL is open source and is available at the download page of the UVM Ecoinformatics Collaboratory (<http://ecoinformatics.uvm.edu>). GrOWL's visualization algorithms include animated force directed layout, interactive locality restricted viewing and selective filtering to simplify the display of large ontologies by selectively reducing detail. The filters provide a mechanism for restricting the view to only class definition, the subclasses, the superclasses, or all instances associated with a selected node. The current GrOWL prototype easily handles large ontologies such as the 1620 KB FungalWeb Ontology: the only visible problem in such cases is a rather long load time. Use of a database backend for storage of the graph will further improve performance and resolve the problem with the load time. Future development plans include this improvement, already supported by the Prefuse library.

At present, GrOWL is being used for two main purposes. The first is to allow non-technical users to browse the structure of the knowledge stored in web-accessible, semantically aware database applications. Used as an applet enriched with a JavaScript communication layer, GrOWL also allows performing assisted queries using a graphical interface. Specifically, GrOWL has been used in the Ecosystem Services Database (ESD) [7],[8] a data and analysis portal to assist the informed estimation of the economic values of ecosystem services [30]. ESD extensively uses OWL format for the description of dynamic models as well as composite datasets with extensive metadata. Users of the ESD can use GrOWL to located a concept of interest, following relationships and with intuitive access to annotation properties. The right-click menu in the applet gives access to a set of queries that will locate instances of the selected concept or use the concept to restrict queries being defined. The representation of instances offered by GrOWL is also used in the ESD to document the objects retrieved by a user query.

The second important application of GrOWL is to enable collaborative development of ontologies in a workshop context, where not all participants are versed in the concepts and methods of knowledge representation. We have found that the graphical paradigm enabled by GrOWL often resonates better with a non-technical audience than tree-based display such as the ones offered by common ontology development environments. In addition to using GrOWL regularly in classes and presentations, the GrOWL prototype is being evaluated in the context of the SEEK project [31] to provide ontology visualization and editing in large-scale, semantically annotated scientific workflow environments.

## 5 Conclusion

In this article we have discussed a visualization model for OWL and the visualization techniques designed in the GrOWL prototype. A comparison of the GrOWL visualization model with related approaches has been made in section 2. We argue that focusing the visualization around the DL semantics of OWL has clear advantages of simplicity and clarity over other approaches to OWL visualization. The structural mapping  $G$  that constitutes the core of our visualization model naturally follows the recursive definition of DL concept expressions. We have carefully considered the possible alternatives and we have not found another more simple and elegant mapping of this sort.

We have used the following tentative criteria for the performance of OWL visualization frameworks:

1. Sufficient completeness and simplicity to provide a readable rendering of all or almost all elements of Roger L. Costello’s camera ontology [13] and other similar-sized ontologies on a 640 by 800 canvas.
2. Support for separate views of the class definitions, the named class hierarchy, and the whole ontology

Unlike the other visualization frameworks we tested, the GrOWL implementation performs well according to both criteria. The good performance of GrOWL on the first criterion is a result of the visualization model’s affinity with DL semantics.

An important objective for GrOWL was to make ontology browsing and editing more intuitive for non-technical users, limiting exposure to the complexities of DL and forcing good design practices through the workflow supported by the interface. This objective was only partially realized. Although the visualization model in GrOWL is relatively simple, some knowledge of DL is a prerequisite on the part of the user. In order to read ontologies in GrOWL, the user at least has to be familiar with boolean operators  $\sqcap, \sqcup, \neg$  and understands the meaning of quantors  $\exists$  and  $\forall$ . As a result, GrOWL has been most appealing to the users with some background in DLs. A more verbose model perhaps would give some advantages to users who are not familiar with DLs, but such advantages would be short lived. As soon as the user becomes familiar with OWL and the visualization model, the simplicity becomes much more valuable especially during the visualization and editing of large ontologies. Approaches to improve the appeal to less technical users while maintaining minimum complexity criteria are still being evaluated, and will evolve as GrOWL is exposed to more users and problem areas.

GrOWL is a key component of a larger strategy being pursued at the UVM Ecoinformatics Collaboratory to bring collaborative knowledge modelling to mixed, delocalized audiences including entirely non-technical members. This project was started as an answer to the needs of several communities in the ecological and agricultural field, in need of more intuitive and efficient ways to collaboratively develop ontologies to annotate and mediate independently developed data and models. In particular, the ThinkCap knowledge portal infrastructure

(<http://ecoinformatics.uvm.edu/technologies/thinkcap.html>), a web application that provides user interfaces over a remote, multi-ontology knowledge base, is being developed to allow remote users of diverse disciplines and technical levels to develop shared conceptualization that are automatically formalized into OWL or RDFS ontologies. In ThinkCap, users will be able to choose the mode of interaction that best suits their expertise. The entry level will be a Google-like search for concepts, using a sophisticated text search that indexes concept descriptions as well as related web resources or documents. Concepts that are found can be explored in several ways, including GrOWL-enabled graphical concept maps. Concepts that are not found can be submitted for inclusion, in more or less formal ways according to the technical level of the user, with the asynchronous involvement of a knowledge engineer. GrOWL will be instrumental in defining and implementing the ThinkCap browsing and editing paradigm for intermediate, advanced, and administrator users.

## References

- [1] Klein, M.C.A., Broekstra, J., Fensel, D., van Harmelen, F., Horrocks, I.: Ontologies and schema languages on the web. In: *Spinning the Semantic Web*, MIT Press (2003)
- [2] Bechhofer, S., Harmelen, F.V., Hendler, J., Horrocks, I., McGuinness, D.L., Patel-Schneider, P.F., Stein, L.A.: OWL web ontology language reference. (M. Dean, and G. Schreiber (eds.) W3C Recommendation 10 February 2004)
- [3] Horrocks, I.: DAML+OIL: a description logic for the semantic web. *Bull. of the IEEE Computer Society Technical Committee on Data Engineering* **25** (2002) 4–9
- [4] Horrocks, I., Patel-Schneider, P.F., van Harmelen, F.: From SHIQ and RDF to OWL: The making of a web ontology language. *J. of Web Semantics* **1** (2003) 7–26
- [5] Krivov, S., Villa, F.: Towards the paradigm of an ontology-based user interface: From simple knowledge maps to graphical query language. In: In: Ludascher, B., Raschid, L., (Eds) *Data Integration in Life Sciences*, Springer Lecture Notes in Bioinformatics **3615**, (2005)
- [6] Krivov, S.: GrOWL website. URL: <http://ecoinformatics.uvm.edu/dmaps/growl/> (2004)
- [7] Villa, F., M.A.Wilson, DeGroot, R., Farber, S., Costanza, R., Boumans, R.: Design of an integrated knowledge base to support ecosystem services valuation. *Ecological Economics*, **41** (2002) 445–456
- [8] : Ecosystem services database. (URL: <http://esd.uvm.edu>)
- [9] Geroimenko, V., Chen, C., eds.: *Visualizing the Semantic Web*. Springer (2003)
- [10] Chen, C.: Information visualization versus the semantic web. In: *Visualizing the Semantic Web*. Springer Verlag London (2003)
- [11] Heer, J., Card, S.K., Landay, J.A.: *prefuse: a toolkit for interactive information visualization*. In: *CHI '05: Proceeding of the SIGCHI Conference on Human Factors in Computing Systems*, Portland, Oregon, USA, New York, NY, USA, ACM Press (2005) 421–430
- [12] Goguen, J., Harrell, D.F.: *Information visualization and semiotic morphisms* (2003)

- [13] Costello, R.L., Jacobs, D.B.: Camera ontology. website: <http://www.xfront.com/camera/index.htm> (2003)
- [14] Horrocks, I., Patel-Schneider, P.: Reducing owl entailment to description logic satisfiability (2003)
- [15] Grosso, W.E., Eriksson, H., Ferguson, R.W., Gennari, J.H., Tu, S.W., Musen, M.A.: Knowledge modelling at the millenium (the design and evolution of prottegte-2000). In: Proceedings of Knowledge Acquisition Workshop (KAW'99). (1999)
- [16] : OWLViz. (OWLViz, URL, <http://www.co-ode.org/downloads/owlviz/>)
- [17] : EzOWL. (EzOWL URL: <http://iweb.etri.re.kr/ezowl/index.html>)
- [18] Liebig, T., Noppens, O., Track, O.: Fast browsing and easy editing of large ontologies. In: Proceedings of the Second International Workshop on Evaluation of Ontology-Based Tools. (2003)
- [19] : IsaViz: A visual authoring tool for RDF. URL: <http://www.w3.org/2001/11/IsaViz/> (2004)
- [20] : RDF gravity (RDF graph visualization tool). (URL: <http://semweb.salzburgresearch.at/apps/rdf-gravity/>)
- [21] : VisioOWL. (URL: <http://web.tampabay.rr.com/flynn/VisioOWL/VisioOWL.htm>)
- [22] Group, O.M.: Ontology definition metamodel - request for proposal. URL: <http://www.omg.org/docs/ontology/03-03-01.rtf> (2003)
- [23] Brockmans, S., Volz, R., Eberhart, A., Löffler, P.: Visual modeling of OWL DL ontologies using UML. In: 3rd International Semantic Web Conference (ISWC2004), Hiroshima, Japan. (7-11 Nov. 2004)
- [24] Brachman, R.J.: On the epistemological status of semantic networks. In Findler, N.V., ed.: Associative Networks: Representation and Use of Knowledge by Computers. Academic Press, Orlando (1979) 3–50
- [25] Gaines, B.R.: An interactive visual language for term subsumption languages. In: J. Mylopoulos and R. Reiter, Editors, Proc. Of 12th Int. Joint Conf. On Art. Int., Sydney, Australia, Morgan Kaufmann. (1991) 817–823
- [26] Sowa, J.F.: Conceptual graphs summary. In: Conceptual Structures: Current Research and Practice, (T. E. Nagle, J. A. Nagle, L. L. Gerholz, P. W. Eklund, Eds.). Ellis Horwood, New York (1992) 3–51
- [27] Sowa, J.: Conceptual structures: Information processing in mind and machine. The Systems Programming Series. Addison-Wesley Publishing Company (1984)
- [28] Patel-Schneider, P.F., Hayes, P., Horrocks, I.: OWL web ontology language semantics and abstract syntax, W3C recommendation. URL: <http://www.w3.org/TR/owl-semantics/> (2004)
- [29] Ludäscher, B., Gupta, A., Martone, M.E.: Model-based mediation with domain maps. In: 17th Intl. Conf. on Data Engineering ICDE, Heidelberg, Germany, IEEE Computer Society (2001)

- [30] Costanza, R., D'Arge, R., DeGroot, R., Farber, S., Grasso, M., Hannon, B., Limburg, K., Naeem, S., O'Neill, R., Paruelo, J., Raskin, R., Sutton, P., VanDenBelt, M.: The value of the world's ecosystem services and natural capital. *Nature* **387** (1997) 254–260
- [31] : SEEK- science environment for ecological knowledge. (Url: <http://seek.ecoinformatics.org>)