REDUCING THE COMPLEXITIES IN THE COGNITION OF ONTOLOGY KNOWLEDGE REPRESENTATION

R. Sivakumar[1] and P.V. Arivoli[2]

[1] Associate Professor, Department of Computer Science, A.V.V.M. Sri Pushpam College, Bharathidasan University, Trichirappalli, India
rskumar.avvmspc@gmail.com
[2]Project Fellow, Department of Computer Science, A.V.V.M. Sri Pushpam College, Bharathidasan University, Trichirappalli, India
pva.tvr@gmail.com

ABSTRACT

Cognitive assistance in knowledge engineering is a growing concern and information visualization is a very useful means to address this. This paper identifies some requirements for ontology visualization tools offering cognitive assistance and presents solutions with simple knowledge representations. This paper also identifies some of its features and describes areas need to be improved for effective visualization.

KEYWORDS

Cognitive science, Protégé tools, Knowledge engineering, Ontologies.

1. INTRODUCTION

Knowledge capture and knowledge representation are the hot emerging areas of research in the discipline of information technology [1]. Ontology development and semantic web play the vital role to support the research in knowledge representations. In recent years, number of ontology tools have been designed and implemented with the support of visualization. The area of cognitive assistance much requires of visualization techniques for its improvement in performance. An ontology is a conceptualization of a domain into machine readable format [2]. They are becoming increasingly popular modelling schemas for knowledge representation services and applications.

A number of visualization techniques have also been described over the years such as spanning tree layouts, tree-maps [3], fisheye views [4], hyperbolic [5], and 3D hyperbolic layouts [6], aiming to help comprehend and analyze complex information structures. Preference of visualization models vary according to users needs and query context [7]. It is also dependent on the type and extent of the visualized network. One can get benefit by the use of combining the integrated visualization of different types [8].

Is it easier to create proper ontologies? Definitely it is a complex task. To break this complexity, it requires enhanced ability to perceive and estimate created ontology. There exists a number of ways to achieve this enhanced ability to represent ontologies via tree or a graph [9]. The complexity of human interpretation with tree structures increases when a class with many parents appear multiple times in the hierarchy. On the other hand, graph based visualization seemed to be a better choice.

Protégé tools like Jambalaya [10] and OWLviz [11] were implemented as plug-in using graph based visualizations. However the cognitive ability of visualization of various properties via edges with labels is poor due to overlapping of those labels. The main reason here is the representation of classes with big square symbols that requires a lot of display space. In Jambalaya, the visualization is based on a hierarchy and when it gets degenerated, the human cognitive ability becomes complex due to the fact of its inability to represent or display all relations at once. GrOWL [12] is an another solution for problem with Jambalaya but the problem here is its heavy library dependencies. Hence in this paper, a set of notations for ontology visualization is proposed to overcome the reported difficulties and the ontologies stored using OWL DL dialect of OWL language and accessed by including java based OWL Application Process Interface. The purpose of this interface is to reduce the heavy library dependencies.

The remaining part of the paper is organized as follows: Section 2 presents and overview of knowledge representations in various plug-ins of protégé tool. Next, section 3 proposes a simplified version of notations to represent classes, properties, individuals and data types in order to improve the cognitive ability of the users of the visualized ontologies. Finally section 4 concludes with future direction in the research.

2. REPORT OF EXISTING VISUALIZATION METHODS

A group of the protégé ontology visualization methods were selected for this study because it offers a range of different characteristics. Furthermore, protégé is a very widely used ontology tool and its open source environment provides many possibilities for further improvement or extension of additional functionalities in the form of plug-ins.

2.1. Class browser

It [13] is a simple visualization technique that offers a windows explorer-like view of the ontology. In this view, the taxonomy of the ontology is represented as a tree. Here the class hierarchy is displayed as follows: the lower level nodes are organized as a list and placed under their parent. Since it supports multiple inheritance property, classes with more than one parents appear under all their parents. The lists of child nodes may be expanded on demand by clicking on their parent. The instances of a selected class are displayed in a separate pane to the right of the class browser.[Figure. 2.1]

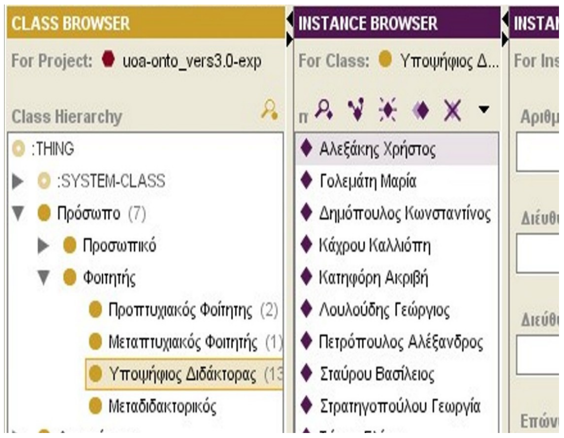


Figure 2.1. The Protégé Class Browser.

2.2. Jambalaya

Jambalaya [14] is visualization plug-in for the protégé ontology tool that uses simple hierarchical multi-perspective 2D visualization technique. It uses a nested graph view and the concept of interchangeable views, combined with interchangeable views, combined with geometric, fisheye and semantic zooming. Here sub classes are nested inside parent classes. The nested nodes are used to represent the inherited relations between the classes. Nested nodes are also used to represent instances with their respective classes in the graph. [Figure. 2.2].

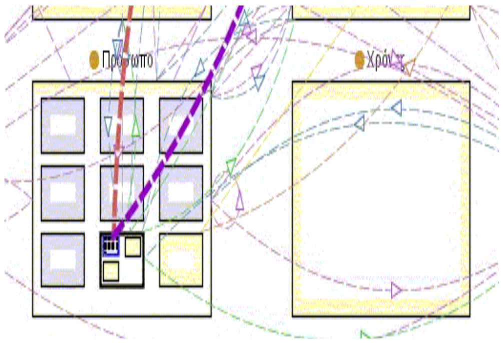


Figure 2.2. The Jambalaya tab in Protégé.

2.3. TGViz Tab

It [15] is also known as touch graph visualization Tab. It uses a spring-layout technique. Here nodes repel one another whereas the edges (represent links) attract them. This displays nodes with similar meaning appear close to one another. Figure.2.3 shows the interface of the TGVizTab. It uses three structures to represent ontologies. It displays classes and instances as nodes with different colours. Links with labels are used to represent relations. The is-a links are denoted as 'sub' links on the other hand role links use a label with the name of the relation they represent. The size of the graph parts may be altered. The instances of a selected class may also be presented in the instance browser on the left.

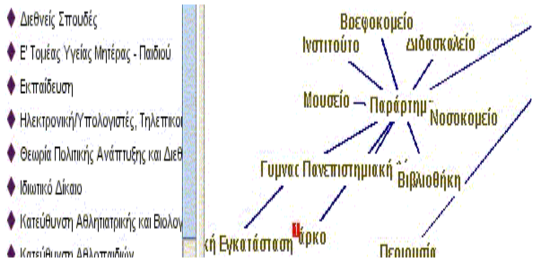


Figure 2.3. The Protégé TGVizTab.

2.4. OntoViz

It [16] is also a protégé visualization plug-in. It uses a very simple 2D graph visualization method. Here the ontology taxonomy is a 2D graph. It has the capability for each class to present, apart from name, its attributes slots and inheritance and role relations.. Different colors are used to display instances. Zooming is possible with right click option. [Figure. 2.4].

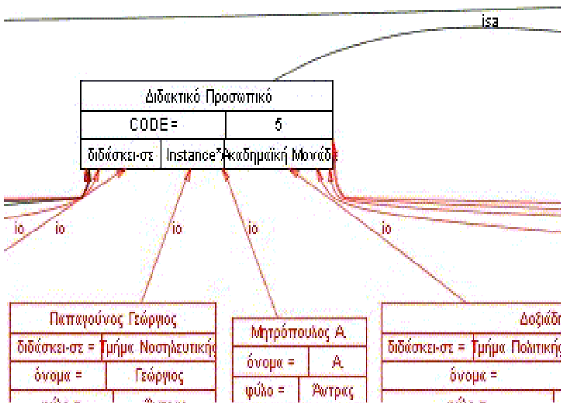


Figure 2.4. Protégé OntoViz Visualization.

2.5. OWL Viz

It [11] is designed to be used with the protégé OWL plug-in. The taxonomy used here is graph. It uses the same colour scheme so that primitive and defined classes can be distinguished and inconsistent concepts can be highlighted in red. Primitive classes are coloured yellow and defined classes are in orange. Selected classes are coloured blue and the edges between selected classes are indicated by light grey colour. In this context, super class edges are represented by green colour and sub classes edges by purple colour. Expansion arrows are used here to indicate that a particular class has some classes that are not shown in the display. [Figure 2.5]

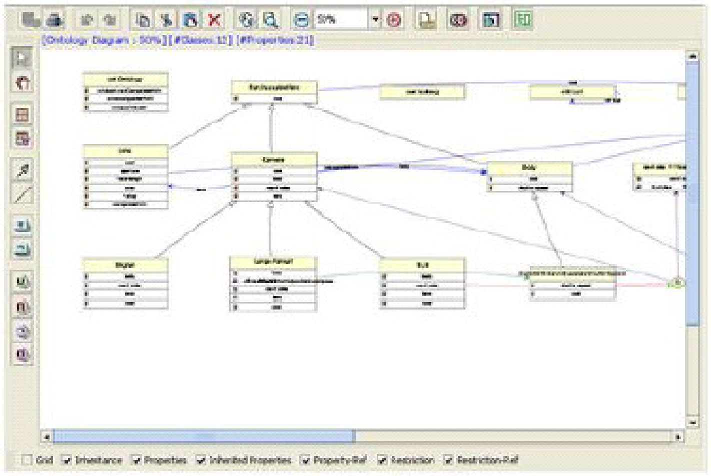


Figure 2.5. Protégé OWLViz plug-in.

2.6. GrOWL

It [12] is based on the prefuse library [17] and implemented as a java applet plug-in and a stand-alone java application. The taxonomy used here is graph. The diagrams from figure 2.6 to 2.13 illustrate the various notations used in ontology representation while the figure 2.14 shows the GrOWL in editing mode. Figure 2.7 and Figure 2.8 provide simple examples that illustrate idioms for axioms SameIndividual and DifferentIndividuals.

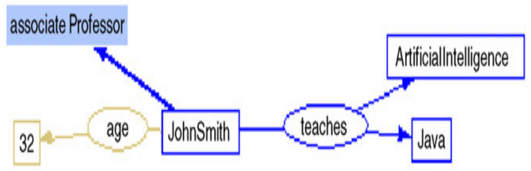


Figure 2.6. Graphic idioms for association about individuals.

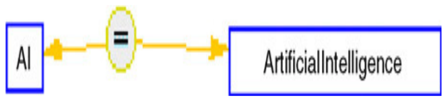


Figure 2.7. Representation of axiom *Same Individual* (AI Artificial Intelligence).

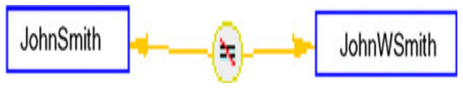


Figure 2.8. Representation of axiom *Different Individual* (JohnWSmithJhonSmith).

In figures 2.9, 2.10 and 2.11, the node labelled BN(C1) is a base node of class C2. They need not have to be of any shape permissible for base nodes.

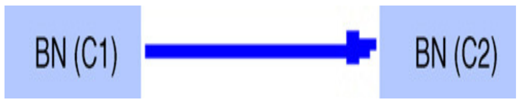


Figure 2.9. Graph G (C1 $\subseteq$ C2).

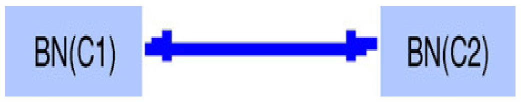


Figure 2.10. Graph G (C1 = C2).

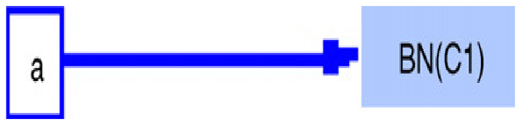


Figure 2.11. Graph G (a:C1).

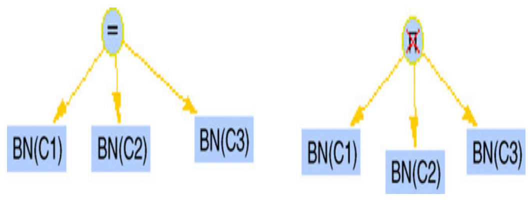


Figure 2.12. Mapping of OWL class axioms EquivalentClasses (C1 C2 C3) and DisjointClasses (C1 C2 C3).

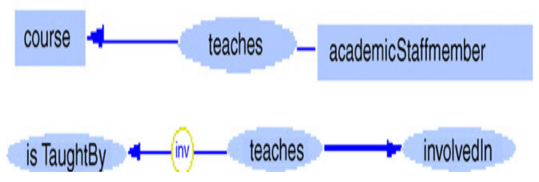


Figure 2.13. The separate diagrams generated by the object property specification.

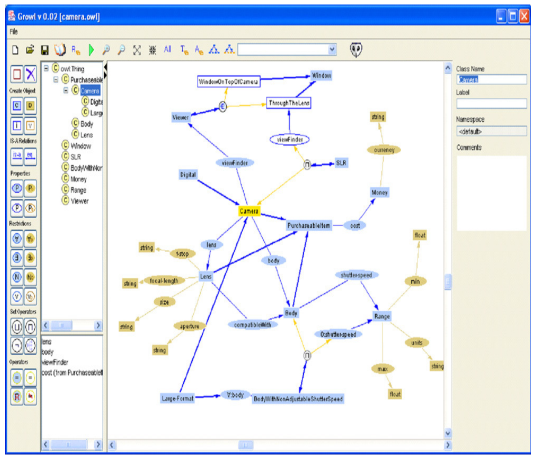


Figure 2.14.GrOWL in editing mode.

Figure 2.10 describes the mapping on the “subclass of” axioms next, figure 2.12 represents the mapping of OWL class axioms.Figure 2.13 shows separate diagrams generated by the object property specification.

Thus it is understood that Protégé tools like Jambalaya and OWLViz were implemented as plug-in using graph based visualizations. However the cognitive ability of visualization of various properties via edges with labels is poor due to overlapping of those labels. The main reason here is the representation of classes with big square symbols that requires a lot of display space. In Jambalaya, the visualization is based on a hierarchy and when it gets degenerated, the human cognitive ability becomes complex due to the fact of its inability to represent or display all relations at once. GrOWL is an another solution for problem with jambalaya but the problem here is its heavy library dependencies. Hence in this paper, a set of notations for ontology visualization is proposed to overcome the reported difficulties and the ontologies stored using OWL DL dialect of OWL language and accessed by including java based OWL Application Process Interface. The purpose of this interface is to reduce the heavy library dependencies.

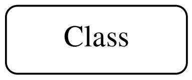
3. PROPOSED NOTATIONS TO INCREASE HUMAN’S COGNITIVE ABILITY

To define ontology elements and its restrictions, OWL language specification permits wide usage of its concepts. It is the responsibility of visualization software to synchronize between ontology definition and its visualization. The human’s cognitive ability of the visualized image needs to be improved still better. This is possible by the way of introducing much simple notations to represent classes, properties and data types. Hence this paper proposes a set of simplified notations to represent classes, properties and data types in ontology visualization.

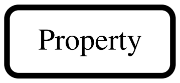
3.1. Proposed notations

The following are the proposed simplified notations used to synchronize ontology representations with visualization for human’s effective knowledge understanding.

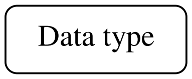
- Class – a rectangle shape with round edges



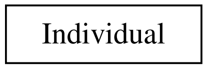
- Properties – a rectangle shape with highlighted round edges



- Data type - a rectangle shape with round edges

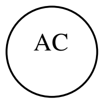


- Individual - a rectangle shape



In complex class description, another problem with cognitive ability is representation of anonymous classes. The solution to this problem is the proposed circle notation containing meaningful character inside.

- Anonymous class -

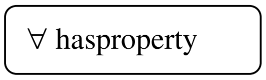


The mathematical symbols $\cap$,$\neg$,$\cup$ and N can be used to represent intersection, complementary, union and cardinality. Further the following symbols can be used as follows.

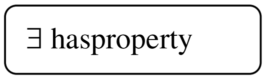
- "SameAs" relation - "="
- allDifferent
- differentfrom } - "≠"

The following notations are introduced for the representation of relations "allValuesFrom" and "someValuesFrom".

- allValuesFrom



- someValuesFrom



Lines with different shafts can be used to represent simple relations. In UML, "subclass" and "subproperty" relations are normally represented by an arrow with an empty shaft and the same is followed here also. But by the meantime, the following changes are suggested for edges that represent "equivalent" and "disjoint" relations as follows.

"equivalent"
"disjoint" } – shafts pointed in opposite directions stressing the reversibility of that relations

The property definition “inversely” is marked by light red colour of the arrow and marked in two different ways depending on the fact whether given property is symmetric or asymmetric. Then how to distinguish equality of classes and properties?

Equality of classes – dark blue colour
Equality of properties –light blue colour

Similarly to distinguish “range” and “domain” relations attached to a property, two additional shafts were introduced [Figure 3.1 to 3.12].

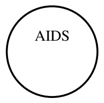


Figure 3.1. Anonymous Class

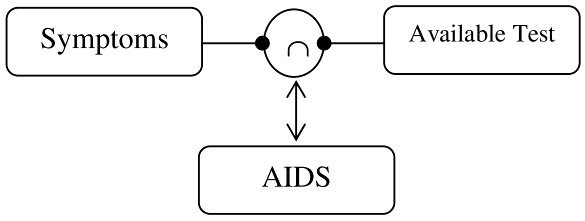


Figure 3.2. Intersection

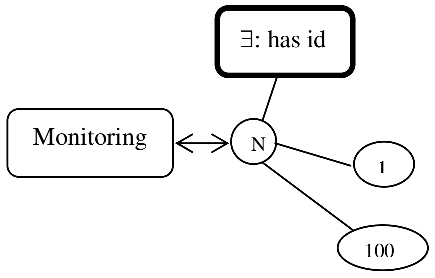


Figure 3.3. Min and Max cardinality

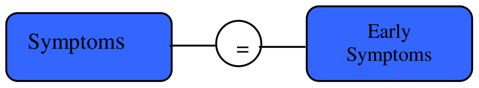


Figure 3.4. SameAs relation

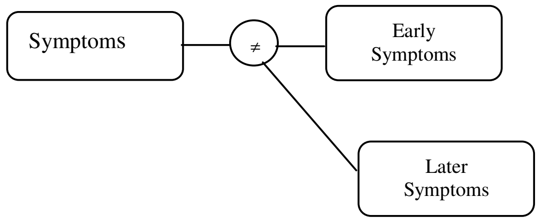


Figure 3.5. all different relation

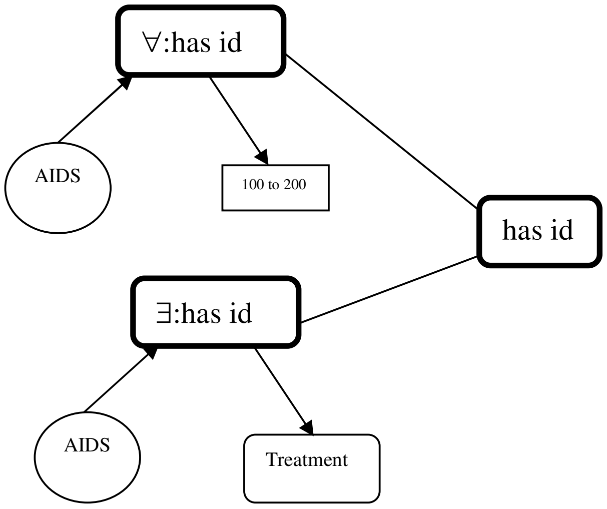


Figure 3.6. someValuesFrom and allValuesFrom relations representation

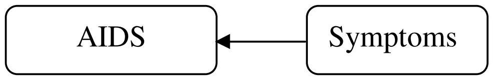


Figure 3.7. rdfs:subclassOf

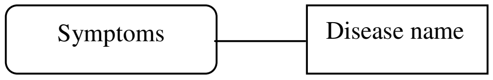


Figure 3.8. instanceOf

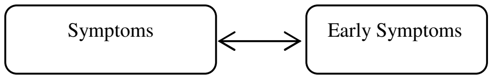


Figure 3.9. owl:equivalentClass

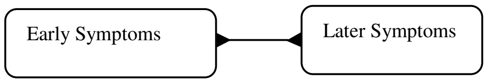


Figure 3.10. owl:disjointWith

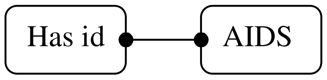


Figure 3.11. rdfs:domain

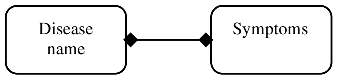


Figure 3.12. rdfs

The following section presents the formal description of algorithm for graph visualizing ontology.

3.2. Algorithm – Formal Description

```
Step 1  Start
Step 2  Input classes, properties, individuals.
Step 3      For each  fact begin
Step 4          If property fact begin
                    *// functional, inverse functional, symmetric, transitive//*
Step 5              Add anonymous node
Step 6              Add links between nodes
Step 7          Endif
Step 8          If individual fact begin *//different, all different, same individuals//*
Step 9          Add anonymous node
Step 10         Add links between nodes
Step 11         Endif
Step 12         If class fact begin
Step 13             If description fact begin
                        *//equivalent classes, subclass, disjoint class//*
Step 14                 Call procedure include details
Step 15             Endif
Step 16         Add links between nodes
Step 17         Endif
Step 18     Endfor
Step 19 Add OWL: Thing element
Step 20 Link not super classes with OWL:Thing
Step 21 Procedure Includedetails
Step 22     begin
Step 23         If SomeValuesFrom or allValuesFrom or haveValues or
                                        Cardinality fact begin
Step 24             Add property usage node
Step 25             Add edges
Step 26             Call procedure Includedetails
Step 27         Endif
Step 28         If set type fact begin *// Union Of , intersection Of, Complement Of//*
Step 29             For each description node begin
Step 30                 Call procedure Includedetails
Step 31             Endfor
Step 32                 If class or individual begin
Step 33                     Add links
Step 34                 Endif
Step 35         Endif
Step 36 Stop
```

3.3. Algorithm Informal description

The aim of this algorithm is to visualize all classes, properties and individuals as nodes. Then it will create proper anonymous classes and insert respective edges. This is possible by scanning all facts and relations related to classes, properties and individuals. At the end this algorithm will add properties with quantifiers such as $\forall$ and $\exists$. The algorithm will first accept all inputs like classes, properties and individuals and scan through facts like classes, properties, individuals and descriptions. The property fact will contain the following cases: functional, inverse, symmetric and transitive. Similarly the relations will take the different forms including inverse property, equivalent and sub property. On the other hand the relations between individuals may take different forms like different, allDifferent and sameIndividuals.

Complex class description can be defined using the following relations between classes: equivalentclasses, subclass and disjointclasses. For fact represents has value, someValuesFrom or allValuesFrom relation with the case as individual, first an anonymous class will be inserted into a graph. Then if require, the cardinality restriction will be added. Similarly a fact defines a set of classes then the following are to be added wherever required: union, intersection and complement.

4. CONCLUSION

In spite of availability of number of plug-ins that supports visualizations in ontology tools, there exist still challenges for easier representation of visualization. In this work a study of various protégé plug-ins for ontology visualization is presented by analyzing various characteristics and notations. Also this work proposed a simplified version of various notations to represent classes, properties and individuals for visualization that synchronizes ontology representations. The future work may incorporate audios with different notations that will definitely improve the cognitive ability of the users.

ACKNOWLEDGEMENTS

This work has been financed by University Grants Commission Major Research Project – Ref. No. 38-4/2009(SR) (India).

REFERENCES

[1]. Allen, M. M., (2003) “Empirical evaluation of a visualization tool for knowledge engineering,” M. Sc., University of Victoria,.

[2]. Guarino, N., Giaretta, P., (1995) “Ontologies and Knowledge bases: towards a terminological clarification”, Towards Very Large Knowledge Bases: Knowledge Building and Knowledge Sharing, IOS, pp25-32.

[3]. Johnson, B., Shneiderman, B., (1991), “Treemaps: a space-filling approach to the visualisation of hierarchical in-formation structures”, Proc. 2nd Int. IEEE Visualization Conference, San Diego.

[4]. Furnas, G. W., (1986), “The FISHEYE view: A new look at structured files”, Proc. of the Conf. on Human Factors in Computing Systems ACM, pp 16-23.

[5]. Lamping, J., Rao, R., Pirolli, P., (1995) “A focus + context Technique based on Hyperbolic Geometry for Visual-izing Large Hierarchies”, ACM Conference on Human Factors in Computing Systems (CHI'95), New York, ACM Press, pp 404-408.

[6]. Munzner, T., (1997), “H3 Laying Out Large Directed Graphs in 3D Hyperbolic Space”, Proc. of the IEEE Symp. on Information Visualisation., Phoenix, USA.

[7]. Graham, M., Kennedy, J., Benyon, D., (2000), “Towards a Methodology for Developing Visualizations”, Int. J. of Human-Computer Studies Vol. 53, No. 5, pp 789-807.

[8]. Risden, K., Czerwinski, M.P., Munzner, T., Cook, D., (2000 “An initial examination of ease of use for 2D and 3D information visualizations of web content”, Int. J. of Human-Computer Studies 53, pp 695-714.

[9]. Katifori, A. Halatsis, C. Lepouras, G. Vassilakis. C, Giannopoulou, E., (2007) “Ontology Visualization Methods -A Survey”, ACM Computing Surveys, Vol. 39, No.4, Article 10.

[10]. Storey, M. Lintern, R. Ernst, N. Perrin, D., (2004), “Visualization and Protégé”, http://protege . stanford.edul conference/20041 abstracts/S torey. pdf.

[11]. Horridge, M., “OWL Viz”, (2010), http://code.google.com/p/co-ode-owlplugins/ wiki/OWL Viz.

[12]. Krivov, S. Williams, R. Villa, F., (2007), "GrOWL: A tool for visualization and editing of OWL ontologies", Web Semantics: Science, Services and Agents on the World Wide Web , Vol 5, Issue 2, 2007, pp. 54—57.
[13]. Sivakumar, R. Arivoli, P.V., (2011), "Ontology Visualization Protégé Tools – A Review", International Journal of Advanced Information Technology (IJAIT),Vol. 1, No. 4.
[14]. Storey, M.A., Mussen, M., Silva, J., Best, C., Ernst, N., Fergerson, R., Noy, N., (2001), "Jambalaya: Interactive visualization to enhance ontology authoring and knowledge acquisition in Protégé", Workshop on Interactive Tools for Knowledge Capture, K-CAP-2001, Victoria, BC, Canada, http://www.thechiselgroup.org/jambalaya.
[15]. Alani, H.,(2003), "TGVizTab: An Ontology Visualisation Extension for Protégé", Proceedings of Knowledge Capture (K-Cap'03), Workshop on Visualization Information in Knowledge Engineering, Sanibel Island, Florida, USA.
[16]. Sintek, M., (2003) "Ontoviz tab: Visualizing Protégé ontologies", http://protege.stanford.edu/plugins/ontoviz/ontoviz.html.
[17]. Heer, J.. Card, S.K. Landay, J.A, (2005) "Prefuse: a toolkit for interactive information visualization", In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems CHI '05, ACMPress, Portland, OR, USA/New York, NY, USA, pp. 421–430.

Authors

R. Sivakumar received his M.Phil degree (1996) in Computer Science from Regional Engineering College, India and Ph.D degree (2005) in Computer Science from Bharathidasan University, India. He has published a number of articles both in national level and international level journals. He has also completed a minor research project on Bioinformatics and currently working on developing visualization tools for health informatics. His areas of interest are Data mining, HCI, Ontology and Information Systems.

P.V. Arivoli received his M.Phil degree (2008) in Computer Science from Bharathidasan University, India. He is currently working as a research project fellow in "Development of Visualization Methods to Integrate Patient Records for the domain of Acquired Immune Deficiency Syndrome (AIDS)" at A.V.V.M. Sri Pushpam College, Bharathidasan University, India. His areas of interests are HCI and Ontology.