OntoSri: A Domain Independent Ontology Visualization Tool

V. Swaminathan, R. Sivakumar

Abstract— **Ontologies provide a shared and common understanding of a domain that can be communicated between people and application systems. Therefore, they may play a major role in supporting information exchange processes in various areas. We describe the role of ontologies in supporting knowledge sharing activities, and then present a set of criteria to guide the development of ontologies for these purposes. We show how these criteria are applied in case studies from the design of ontologies for Clinical Information system. The purpose of Ontosri is to help building ontology, both from scratch and from texts, without control by any task. Requirements have been defined for a methodology on the basis of real experiments. Ontosri fulfills these requirements, involving theoretical bases from linguistics and knowledge representation. Its strong points are integration of a terminological approach and an ontology management, precise definition of concept types reflecting modeling choices, and traceability facilities. The use of these tools is discussed with respect to a case study in health care.**

Index Terms— **Description logics, ontology, tautology, satisfiability, visualization.**

I. INTRODUCTION

Digitization and dissemination of huge amounts of documents have been made possible by the recent developments and continuing progress in network and data storage technologies, which have made searching and retrieval of information either in the web or the digital document collection more and more difficult. The need for easier and more effective information retrieval has necessitated the creation of semantic web and personalized information management concepts, which are areas of study that take advantage of the semantic context of documents, to facilitate their management. Taking advantage of ontology is one of the various solutions proposed, in this context. Ontology, though it is a term originally borrowed from philosophy, is now used to denote a set of concepts and their interrelations in a specific domain. As a consequence, the need for effective ontology visualization for design, management and browsing has emerged. Ontology is something more than a hierarchy of concepts. It is enriched with role relations among concepts and each concept has various attributes related to it. Furthermore, each concept most probably, has instances attached to it, which could range from one or two to thousands. Therefore, it is not simple to create a visualization that will effectively display all this information and at the same time allow the user to easily perform various operations on the ontology. In the field of ontology visualization, there are several works, mostly in 2D. Apart from the systems that propose visualizations especially tailored for ontologies, there are a number of other techniques used in other contexts such as graph or file system visualization that could be adapted to display ontologies.

Manuscript received Jan 23, 2015

V. Swaminathan, Department of Computer Science, A.V.V.M. Sri Pushpam College, Bharathidasan University, Trichirappalli, India

R. Sivakumar, Department of Computer Science, A.V.V.M. Sri Pushpam College, Bharathidasan University, Trichirappalli, India

II. METHODOLOGY

Highlights of OntoSri tool:
Full ontology visualization
Object properties graphical presentation
Distinguished color schemes
Built-in Reasoner

A. Selected Visualization Techniques for OntoSri

The difficulty when designing a visualization tool OntoSri is how to achieve a good visualization where all the information is showed in a coherent manner. Features such as New, Save, Open, Refresh and Exit are mainly too much information to show in one representation. (Figure 4.1).This is not only important to show all the components, the visualization must also allow a human being to build a correct mental model of the model shown. Even when the model is very large, the visualization must support this.

One can say that the key to measuring the solvency of good feature model visualization is obtained by the impressions from a user when looking to the model. In the effort to find a single visualization that meets all the characteristics discussed above, it is that the best way to visualize all the content of a OntoSri was using the graph view.

Figure 2.1a. OntoSri Open Page

The user has to be able to visualize the two types of representations. This means that when the user is interested in analyzing the hierarchy he/she can use the Reasoner. Otherwise, from the graph model the user can visualize all the possible information and interact with it. The more different visualizations are provided for a certain feature model, the easier it will be for the user to understand and manipulate the model.

Following this theory, it is also decided to add another representation: the indented list. The idea is to provide the graph view with a complementary view and interaction mode representing the hierarchy of the feature model. In this way, the user can see the structure of the hierarchy (using the indented list) while is working with the graph model. Thus, the final decision was the utilization of two types of visualizations for the OntoSri Tool: the Graph model and the Reasoner model. The following subsections show in detail the design of the different visualizations, and also how the user can interact with each one.

The Graph Model

A graph is a tree that shows a set of relationships, often functional, between a group of points or numbers. Each of these points or numbers has coordinates determined by their relationships. This tree represents a mathematical structure or a symbolic representation of a network.

The graph representation of OntoSri is based on this definition. This is the only representation that tries to show almost all the content of the model at once. The graph representation will be given in the main window of the tool. From there, the user will work most of the time when developing the OntoSri model.

The Reasoner Model

OntoSri allows for special-purpose reasoner that provides standard reasoner services such as satisfiability (of a single class as well as consistency of the ontology) and subsumption (between classes and between properties).

For the ease of user interaction, the open menu that contains all the possible modeling actions is provided. It is composed of(Figure 2.1a) five buttons, and the function of each one is described in the next paragraphs.

The first button gives all the possible modeling actions provided. It is composed of eight buttons, and the function of each one is described in the next paragraphs. The first button gives the modeler the possibility to create a new project to the graph-view. When clicking this button, the representation of the new project appears as a box situated at the position of the cursor. The user only has to click (yes) this box to create a new project as menu with eight buttons (Figure 2.1b).

The first button is used to open Create Class. When the user clicks on it, the cursor becomes a point and the tool changes the status of operation, which in this case open a window with Class Name and Available Classes. The second button is used to open Adding Objects. When the user clicks on it, the cursor becomes a point and the tool changes the status of operation, which in this case open a window with Select the Class , Object Name and Data Type and also with three buttons namely Add, Remove Member and Remove all Members.

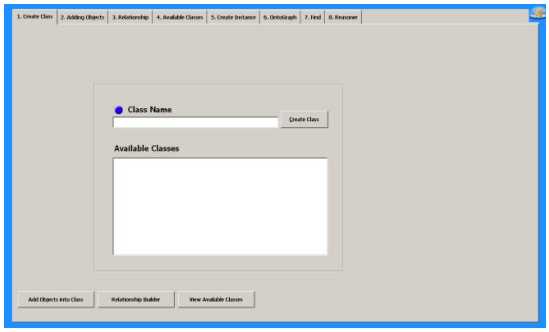

Figure 2.1b Use Case of Create Class

The third button is used to open Relationships. When the user clicks on it, the cursor becomes a point and the tool changes the status of operation, which in this case opens a window with three options, namely Class 1, Class 2, and Member – Relates- Member, also with New Relation Creation button, View Existing Relation button and Create Relation button. The fourth button is used to open Available Classes. When the user clicks on it, the cursor becomes a point and the tool changes the status of operation, which in this case open a window with Available Class.

The fifth button is used to open Create Instance. When the user clicks on it, the cursor becomes a point and the tool changes the status of operation, which in this case open a window with Select Class and Available Instances also with Create New Instance button.

The sixth button is used to open Ontograph. When the user clicks on it, the cursor becomes a point and the tool changes the status of operation, which in this case open a window with Draw Class and Current Class. The seventh button is used to open Find. When the user clicks on it, the cursor becomes a point and the tool changes the status of operation, which in this case open a window with Select Class, Select Object and Enter Keyword also with two buttons namely, Search and Show All. The eighth button is used to open Reasoner. When the user clicks on it, the cursor becomes a point and the tool changes the status of operation, which in this case open a window with Enter the Keyword and Search button.

The addition of save the current project is implemented by means of the second button in the open menu. When the user clicks on it, the cursor becomes a point and the tool changes the status of operation, which in this case save in the desired location. The third button is used to open the existing project. When the user clicks on it, the cursor becomes a point and the tool changes the status of operation, which in this case open in the desired location.

The fourth button is used to refresh the existing project. When the user clicks on it, the cursor becomes a point and the tool changes the status of operation, which in this case refresh the existing project. The fifth button is used to exit the project. When the user clicks on it, the cursor becomes a point and the tool changes the status of operation, which in this case exit from the application.

The tableau method is a way of verifying tautologies. It can be faster than truth tables, and for this method, not even need a definition of truth and false. The analytic tableau method consists of three rules with which it is possible to convert a formula into a graph where it can easily be seen whether the formula is a contradiction. Since a tautology is

always a negation of a contradiction, one can negate a suspected tautology and test if it is a contradiction.

Description Logics

The basic notions in Description logics are concepts (unary predicates) and roles (binary predicates). Description logics are logic formalisms used as a basis for the Semantic Web ontology languages and they offer reasoning services, which can be applied to reasoning with ontologies. Reasoning is important to ensure the quality of ontology. This section defines syntax and semantics of ALC (Attributive Language Complements) and defines the tableau algorithm.

The ALC Description Logic

The smallest propositionally closed Description logic is the ALC Description logics. The syntax and semantics of the language is given by the following two definitions:

Definition 1 (Syntax of ALC Language)

Let NC and NR be disjoint and countable infinite set of concepts and role names. The set of ALC-concepts is the smallest set, such that:

1. Every concept name A∈ NC is an ALC concept.
2. If C and D are ALC concepts and R≡NR then ¬C, C ∩ D, C ∪ D, ∃R.C and ∀R.C are ALC concepts.

Every ALC formula can be constructed applying 1 and 2 rules.

Definition 2 (Semantics of ALC Language)

An ALC interpretation is a pair $(\Delta I, \cdot I)$ where ΔI is a non-empty set called domain, and $\cdot I$ is an interpretation function that maps every concept name A to a subset AI of ΔI and every role name to a binary relation RI over ΔI. The Interpretation function is extended to complex concepts as follows:

1. $(\neg C)\ I = \Delta I \setminus CI$,
2. $(C \sqcap D)\ I = CI \cap DI$,
3. $(C \sqcup D)\ I = CI \cup DI$,
4. $(\exists R.C)\ I = \{d \in \Delta I \mid (\exists e)\ ((d, e) \in RI \wedge e \in CI)\}$,
5. $(\forall R.C)\ I = \{d \in \Delta I \mid (\forall e)\ ((d, e) \in RI \Rightarrow e \in CI)\}$.

The Tableau Algorithm

The basic reasoning services in Description logics are: subsumption, consistency, satisfiability, and instance checking. Satisfiability of a concept expression C is a problem of checking whether there exists a model. It means whether exists an interpretation I in which CI≠ In that context the interpretation I is a model for a concept C. Other reasoning services can be calculated with satisfiability.

The tableau algorithm tries to prove satisfiability of a concept term C, by demonstrating a model in which C can be satisfied. A tableau is a graph that represents such a model with nodes corresponding to individuals and edges corresponding to relationships between individuals. Every Description logics term can be represented by a tree (a special case of graph) structure, but a tableau has different structure then a Description logic formula.

Reasoning Services

Given a Knowledge base KB = (T, A) and two concepts C and D: Concept Satisfiability , written KB⊭ C ≡ ⊥, is the problem of checking whether the Concept C is satisfiable w.r.t. KB, i.e. there exists a model I of KB such that $CI \neq \emptyset$

Subsumption , written KB⊨ C ⊑ D , is the problem of checking whether the Concept C is subsumed by the concept D w.r.t KB i.e. CI⊆DI in every model of KB Consistency , written KB⊭ ⊥ , is the problem of checking whether KB is satisfiable or has a model Instance Checking, written KB⊨ C (a), is the problem of checking whether the assertion C(a) is satisfied in every model of KB

Introducing Individuals

C is satisfiable ⇔ C⋢ ⊥

C is satisfiable ⇔ { C(a)} is satisfiable

C ≡ D ⇔ C ⊓ ¬ D is not satisfiable

C ≡ D ⇔ { C(a)} ⊨ D (a)

KB is satisfiable ⇔ KB ⊭ ⊥

KB ⊨ C (a) ⇔ KB ∪ { ¬C(a)} is not satisfiable

The Subsumption Testing Algorithm Used in this paper

Step 1 Transform C ⊏ D into the test of unsatisfiability of the expression C∏ ¬D.

Step 2 Make full negations available for the last test.

The Tableau-Based Algorithm Used in this paper

Step 1 Transform KB within a Constraint system SK.

Step 2 Apply Systematic available tableau rules for reducing the constraint system until saturation.

Step3 if the complete system is free of contradiction or clash-free, it is satisfiable and a model of KB can be produced.

The Constraints

The construction of a constraint system relies on the use of variables allowing to define constraints.

x: C or C(x) means that x is an instance of C (concept instantiation).xry or r(x, y) means that x is in relation with y through the role r (role instantiation where y is a r-successor of x). x≠y means that the individuals x and y have a different interpretation; x and y are then separated.

The variables are supposed to be ordered by their apparition order, and two variables x and y are equivalent in S∑ if they verify the same instantiations

The Tableau-Based Algorithm Used for Testing Satisfiability in Dls

Aim- To Test the satisfiability of a knowledge base using the tableau-based Calculus

Step 1 Transform ∑ into a constraint system S∑

Step 2 Apply a set of decomposition rules until the no more decomposition rule is applicable

Step 3 If the complete system obtained does not contain any clash or contradiction, then the system is satisfiable

The Subsumption Testing Algorithm Used in this paper

Step 1 Transform C ⊏ D into the test of unsatisfiability of the expression C∏ ¬D.

Step 2 Make full negations available for the last test.

The Tableau-Based Algorithm Used in this paper

Step 1 Transform KB within a Constraint system SK.

Step 2 Apply Systematic available tableau rules for reducing the constraint system until saturation.

Step 3 If the complete system is free of contradiction or clash-free, it is satisfiable and a model of KB can be produced.

Indented List

The indented list is the second type of visualization in this tool (Figures 3.2(a), 3.2(b), 3.2(c) and 3.2(d). The reason for the inclusion of this representation is due to the difficulty for the users to recognize the composition hierarchy from the graph-view. So, the indented list can be considered a visualization support for the graph-view

Figure 2.2 (a) OntoSri Indented List

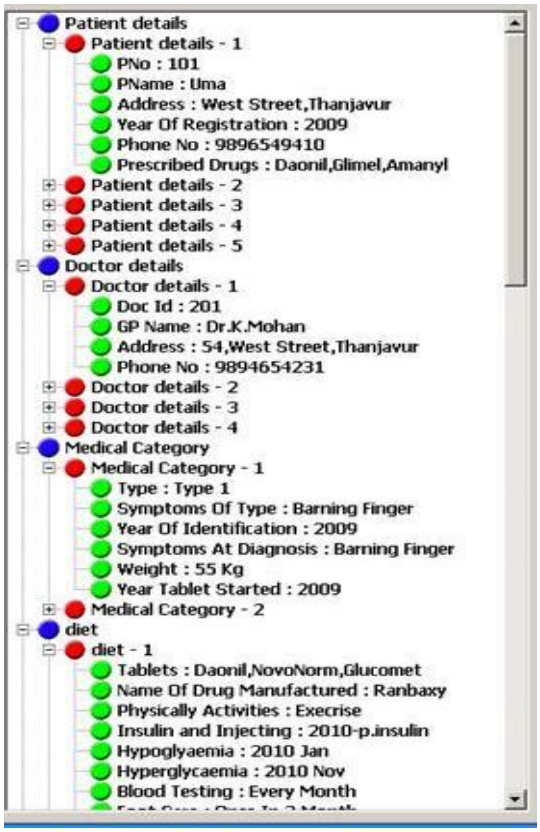

Figure 2.2 (b) OntoSri Indented List

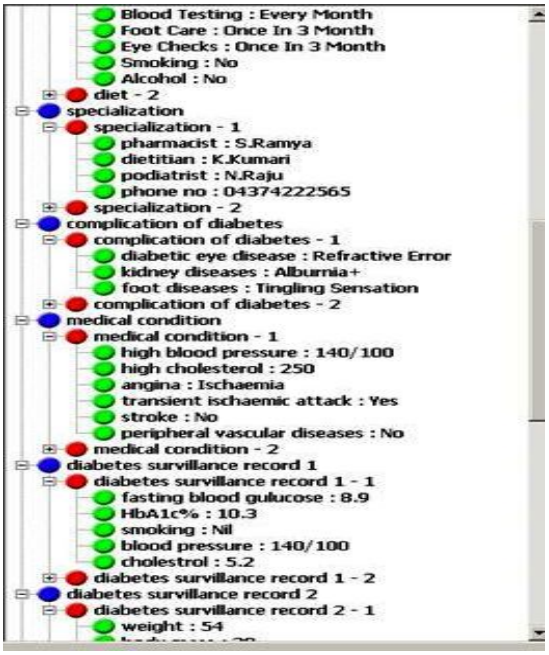

Figure 2.2 (c) OntoSri Indented List

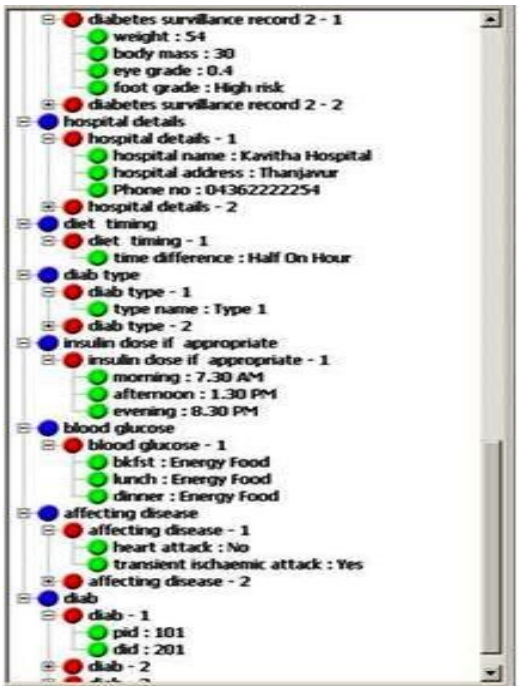

Figure 2.2 (d) OntoSri Indented List

A list is a series of objects organized in a logical order. In this case the objects are the features of the model, and the organization is the composition hierarchy of this elements. Sometimes a list representation is difficult to read, especially when the organization of the list elements is more than a simple succession of objects. The indented list is a solution to represent the hierarchical organization of the feature model as a serial. The child elements in the list are placed under their parents and indented to the right. Development of advanced features such as filtering, querying and reasoning tasks in an ontology visualization tool named Ontosri that increases the cognitive support of domain experts who are not ontologists.

Contributed Features:

Using color and decorations for different resource types and sub-types Optimizing the layout of a diagram to illustrate hierarchy and other types of relationships Expanding and contracting classes Representing property restrictions, associations and compound resources. Development of Reasoning capabilities (Tabulax method)

Description of Use Cases of OntoSri

The Use Cases describe what the system has to do from the point of view of the user (Figure 2.3). That is, the description of the tool's behavior regarding the use of it and its interaction with the user. The Use Case depends on the origin of the actions and the users preferred visualization view.

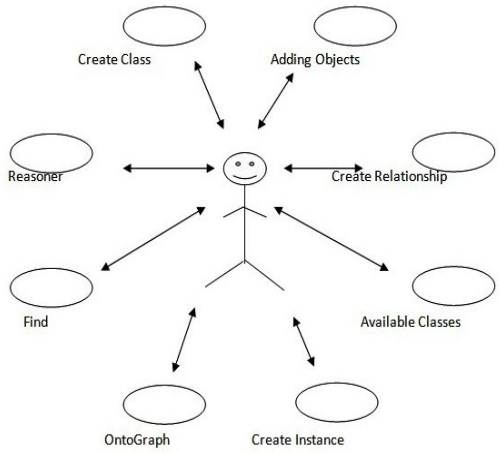

Figure 2.3 Use Cases of OntoSri

The set of Use Cases are related to the relevant actions that the ontologiest is allowed to realize interacting with the OntoSri tool. The lists of these actions are: Create Class, Adding Objects, Relationship, Available Classes, Create Instance, OntoGraph, Find and Reasoner. In addition, each description of Use Case contains a screenshot as auxiliary information for the understanding of the flow.

The functioning of OntoSri tool is tested with a case of Diabetes ontology. Indented List presents the basics of diabetes, ontology development and use case description of OntoSri with a case of diabetes ontology

CONCLUSIONS

The main objective of this paper was to design a user-friendly application visual interactive application for feature modeling. Current feature models do not scale very well in real industrial cases where the number of features becomes very large. As the number of features grows, along with the increasing number of relations between features, the need arises to have good visualization tools that allow modelers to quickly and efficiently create scalable feature models that clearly show features and their relations. The main challenge in this thesis was to identify from an HCI perspective the best method(s) to visualize feature models such that not only inspect and understand the model but also efficiently create and interact with the model. In addition, the existence of feature modeling tools is not very large and the majority of these tools are designed with regard to support the configure product lines from the feature models. Therefore, the design was more difficult because of the lack of information and examples. This paper tried to provide the state of the art, complemented with some own reflections, about the visualization and interaction of feature models. From this research and the design of the tool, the following knowledge is acquired. Need of feature model visualization The different possible presentations of feature models Benefit of the different type of relations available in the feature model to have multi types of representation Possibility to have more than one presentation of a feature model available at the same time in a tool and being efficient for feature modeling Application of HCI perspective in visualizing feature models, and to make the user experience better The performance results show the effectiveness of the designed tool OntoSri in all aspects. When a reasoner is needed for a more expressive language, existing reasoners are not applicable. Hence such an expressive language needs to be designed for reasoning more expressive ontology. Another important issue in ontology feature modeling is development of additional features such as ontology mapping, alignment and merging activities in the existing tools and is due to the fact that existing such tools are semi automatic in nature.

REFERENCES

[1] Katifori. A, Torou. E, Halatsis. C, Lepouras. G, Vassilakis. C, (2006). A Comparative Study of Four Ontology Visualization Techniques in Protégé: Experiment Setup and Preliminary Results,URL:http://ieeexplore.ieee.org /stamp/stamp.jsp?arnumber =01648294.

[2] ACM Special Interest Group on Computer-Human Interaction Curriculum Development Group. New York, ACM Akrivi Katifori, Constantin Halatsis, George Lepouras, Costas Vassilakis, and Eugenia Giannopoulou, "Ontology visualization methods - a survey.," , 2007, pp. Bd. 39. Nr. 4. S. 10:1 - 10:25.

[3] Alani, H. (2003). TGVizTab: An Ontology Visualisation Extension for Protégé. In: Proceedings of the Workshop on Visualizing Information in Knowledge Engineering at (K-CAP'03), Sanibel Island, Florida, USA

[4] AMANN, B., AND FUNDULAKI, I. 1999. Integrating Ontologies and Thesauri to Build RDF Schemas. In Proceedings of the Third European Conference on Research and Advanced Technology for Digital Libraries, 234 – 253.

[5] Andreas Rosendahl, "Visualization of knowledge in the eParticipation ontology,".Suntisrivaraporn. B. Polynomial-Time Reasoning Support for Design and Maintenance of Large-Scale Biomedical Ontologies. PhD thesis, TU Dresden, 2009.

[6] Benjamin B, Emmanuel P, Ilaria L, Gennady L (2011) OntoTrix: A Hybrid Visualization for Populated Ontologies. Proceedings of WWW2011, Hyderabad, India

[7] Berners-Lee, T., Handler, J., Lassila, O, (2001), The Semantic Web. Scientific American.

[8] Boinski T, Bundnik L, Jaworska A, Mrozinski J, Mazurkiewicz, K (2009) OCS-Domain Oriented Ontology Creation System. Polish Journal of Environmental Studies. 18:3B.pp 35-38.

[9] Daren Nestor, Steffen Thiel, Goetz Botterweck, Ciar´an Cawley, Patrick Healy. Applying Visualisation Techniques in Software Product Lines. Lero, the Irish Software Engineering Research Centre - University of Limerick, Ireland.

[10] Dix A., Finlay J. E., Abowd G. D. and Beale R. (1998) Human-Computer Interaction. 2nd Edition. London, Prentice Hall EuropeEades P (1984) A Heuristic for Graph Drawing. Congressus Numerantium,42:149-160.

[11] Baader. F, Ganter. B, Sertkaya. B, and Sattler. U. Completing description logic knowledge bases using formal concept analysis. In IJCAI, pages 230–235, 2007.
[12] Baader. F, Calvanese. McGuinness. D.L, Nardi. D, and Patel-Schneider. P. F, editors. The Description Logic Handbook. Cambridge University Press, 2nd edition, 2007.
[13] Donini. F, Lenzerini. M, Nardi. D, and Schaerf. A. Reasoning in Description Logics. Principles of knowledge representation, pages 191–236, 1996.
[14] Fluit, C., Sabou, M., & van Harmelen, F. (2005). Visualizing the Semantic Web: XML-based Internet and Information Visualization. In V. Geroimenko & C. Chen (eds.), Visualising the Semantic Web (pp. 45-58): Springer Verlag.
[15] Furnas GW (1986) The FISHEYE view: A new look at structured files. Proc. of the Conf. on Human Factors in Computing Systems ACM. pp 16-23.
[16] Gene Ontology Consortium: http://www.geneontology.org/.
[17] Gomez-perez A Survey on ontology Tools, OntoWeb deliverable D1.3. http://ontoweb.aifb.uni-karlsruhe.de/About/Delieverables /D13_1-0.zip.
[18] Graham M, Kennedy J, Benyon D (2000) Towards a Methodology for Developing Visualizations. Int. J. of Human-Computer Studies 53(5): 789-807.
[19] Graham, M., Kennedy, J., Benyon, D., 2000. Towards a Methodology for Developing Visualizations. Int. J. of Human-Computer Studies 53(5): 789-807.
[20] GRUBER, T. R. 1993. A Translation Approach to Portable Ontology Specifications, Knowledge Acquisition. Special issue: Current issues in knowledge modeling, Vol 5, Issue 2, 199-220
[21] Guarino N, Giaretta P (1995) Ontologies and Knowledge bases: towards a terminological clarification. Towards Very Large Knowledge Bases: Knowledge Building and Knowledge Sharing, IOS, 25- 32.
[22] 22.Guarino, N. (1998). Formal Ontology and Information Systems. In: Proceedings of the 1st International Conference on Formal Ontologies in Information Systems (FOIS'98) (pp. 3- 15), Trento, Italy.
[23] Herman I, Melancon G, Marshall MS (2000) Graph Visualization and Navigation in Information Visualisa-tion: a Survey. IEEE Transactions on Visualisation and Computer Graphics 6(1):
[24] http://www.infovis.net/E-zine/2002/num_85.htm
[25] Ioannis Papadakis and Machalis Stefanidakis, "Visualizing Ontologies on the Web," in Studies in Computational Intelligence, Volume 142.: Springer-Verlag, 2008, pp. 303-312.
[26] Jambalaya,URL:http://www.thechiselgroup.org/jambalaya.
[27] Katifori A, Halatsis C, Lepouras G, Vassilakis C, Giannopoulou E (2007) Ontology Visualization Methods - A Survey, ACM Computing Surveys, Vol. 39, Issue 4.
[28] Katifori A, Torou E, Halatsis C, Lepouras G, Vassilakis C (2006) A Comparative Study of Four Ontology Visualization Techniques in Protégé: Experiment Setup and Preliminary Results, URL:http://ieeexplore.ieee.org /stamp/stamp.jsp?arnumber =01648294
[29] 29.Katifori, A., Halatsis, C., Lepouras, G., Vassilakis, C., Giannopoulou, E., (2007). Ontology Visualization Methods - A Survey, ACM Computing Surveys, Vol. 39, Issue 4.
[30] Nadia C, Lorenzo, Riccardo (2009)User-friendly ontology editing and visualization tools: the OWLeasyViz approach. 13th International Conference Information Vi sualisatio.
[31] TGVizTab Protégé plug-in URL:http://users.ecs.soton.ac.uk/ha/TGVizTab/