UNIVERSITY
OF TRENTO

DEPARTMENT OF INFORMATION AND COMMUNICATION TECHNOLOGY

38050 Povo – Trento (Italy), Via Sommarive 14
http://www.dit.unitn.it

EVALUATION AND RANKING OF ONTOLOGY CONSTRUCTION TOOLS

Md. Ahsan-ul Murshed and Ramanjit Singh

March 2005

Technical Report # DIT-05-013

Evaluation and Ranking of Ontology Construction Tools

Ahsan-ul Morshed[1], Ramanjit Singh[2]

[1]DIT-University of Trento
38050 Povo, Trento, Italy
[2]School of Informatics
University of Manchester
United Kingdom
Emails: morshed@dit.unitn.it[1]
ramanjit.singh@postgrad.manchester.ac.uk[2]

Abstract. The emphasis of reusing domain knowledge in system development makes ontology the most important disciplines in the information system industry at present. Many domain specific ontology editors are currently available for developing ontologies. However, selecting the most appropriate editor is a challenge because each ontology construction initiative requires its own budget, time, and resources. Therefore, to overcome this challenge, we propose a criterion to evaluate ontology construction tools. We define evaluation criteria such as functionality, reusability, data storage, complexity, association, scalability, resilience, reliability, robustness, learnability, availability, efficiency, and visibility. Then, on the basis of criteria, we propose a technique to rank ontology tools and finally, we evaluate three ontology construction editors based on the ranking technique.

Keyword: Ontology, Ontology Evaluation, Ontology Editors, Ranking, Knowledge Model.

1 Introduction

The ontology can be defined as the textual or graphical description of a conceptualization. It can be used to share knowledge, by using similar vocabulary, semantics, and relationships among concepts of a particular domain. In addition, ontologies are very practical for explaining metadata terms and structuring domain knowledge in a structured and standard way. This type of standardization facilitates reuse and enables the applications to cooperate with one another more efficiently. Using ontologies, developers can build more intelligent systems that can understand each other more thoroughly [1]. Moreover, ontology in general is costly to develop and maintain. In order to save time and cost of developing and maintaining ontologies, one need to select the right ontology construction tool. This however is not an easy task since ontology construction tools are still premature and lack the ability to fully manage development and evolution of ontologies. Therefore, it is important to first evaluate the ontology construction tool before starting the ontology development.

The rest of the paper is organized as follows: In following section 2, we introduce the ontology and construction tools, the research method and define some selection criteria for ontology editors explicating the reasons for selecting those criteria. In section 3, we formulate the scenario and cases, and in section 4, we propose a ranking technique to evaluate ontology editors. Section 5 presents the results and discussion. Lastly, the paper ends with a short conclusion.

2 Ontology and Construction Tools

Ontology is constructed by defining classes, properties, relationships, and instances for a particular domain. It can be used to share knowledge, by using similar words, meanings, and relationships among abstractions of a particular domain. Furthermore, the development of ontology has been shifted from AI experts to domain experts. So ontologies on the Internet now vary from large classifications of web sites such as Yahoo [16] to grouping of products for sale on Amazon.com [3, 17]. The WWW consortium [11] and the US defense have been creating ontology languages such as RDF and DAML [11] to facilitate the communication among various types of web sites on the Internet. These languages aim to standardize the ontologies so that domain experts can create the knowledge on the similar grounds. For example, the field of medicine has created huge number of common vocabularies, and semantics to create standardized application from hospital to hospital [3]. Also, the Stanford institution has developed a standard classification for wines [3].

Although the research in the area of ontology is progressing rapidly, there is a paucity of research in selecting an appropriate ontology tool. Some progress work has been done by Stojanovic et al., [6], who propose a criterion to evaluate and assess ontology editors but mainly to discuss ontology evolution. Also, Denny [7] proposes criteria such as version, release date, source, graph view etc., which gives an overview of the editor but does not help the developer to rank quantitatively. Furthermore, Jürgen et al, [8] evaluate the ontology editors based on both technological and ontology-related properties and Lambrix et al., [9] evaluate ontology-merging tools for bioinformatics. However, none of these works rank ontology editors quantitatively. In this paper, we propose an

appropriate evaluation criteria and a ranking technique to help the ontology developers to choose the right construction tool. Then, some popular editors such as Protégé-2.1.2, Metis-3.4 and OntoEdit Free are evaluated based on quantitative and in qualitative measures.

2.1 Research Method

This research will be qualitative as well as quantitative. The qualitative criteria will be applied for each tool. Then ranking technique will be used to provide a rank for the criteria. More specifically, the research method can be described into seven consecutive steps.

1. We initiated the research work by conducting a literature study on ontologies. We investigated various tools such as Protégé-2.1.2, Metis-3.4, and OntoEdit Free to gain some background knowledge tools. More emphasis was given to Protégé since the prestige's Stanford institution developed it. Also, Stanford's is first mover and it has gained valuable experience in the ontology tool industry. Thus, we examined various ontology examples on the Protégé's website to gain a thorough understanding regarding ontology development. Then series of ontology examples in the user manuals of each editor were studied. Upon conducting a thorough literature analysis, questions such as: what is ontology, what purpose does it serve, and how can one create ontology for a particular domain were answered.
2. We developed a set of criteria to evaluate ontology construction tools. Each criteria aspect has relevant sub aspects.
3. We created a set of scenarios with appropriate test cases to develop five ontologies with each editor.
4. We developed an appropriate ranking technique.
5. We evaluated and ranked each construction tool with respect to the evaluation criteria.
6. We compared the evaluation results with a discussion.

2.2 Evaluation Criteria for Ontology Construction Tools

Upon conducting the literature study, we identified important key elements that are most significant for the selection of an ontology editor. The proposed criteria are functionality, reusability, data storage, complexity, association, scalability, resilience, reliability robustness, learnability, availability, efficiency and visibility.

Furthermore, an ontology tool consists of many atomic functions, with a set of function comprising a criterion and an atomic function representing a sub-criterion. To find out the criteria, we follow the bottom-up approach where, we find out all the individual atomic functions and then group the set of related functions into a criterion. The criteria and the reasons behind their selection are given below.

I. Functionality

Add: Degree to which the tool enables the addition of ontology classes, properties, relations, and class instances.

Modify: Degree to which the tool enables the modification of ontology classes, properties, relations, and class instances.

Delete: Degree to which the tool enables the removal of ontology classes, properties, relations, and class instances.

Functionality was used as criteria aspect since adding, modifying, and removing elements in a model are one of the important features of any given ontology construction tool. Adding enables one to add classes, properties, relationships, and instances to the ontology model. Modifying allows one to change the added classes, properties, relationships, and instances from the ontology model. Removing enables the deletion of added or changed classes, properties, relationships, and instances from the ontology model. So the above-mentioned functionality is necessary for the editor to facilitate ontology development in a professional manner.

II. Reusability

Reusing ontology: Degree to which, existing ontology files can be used from a new editor version.

Reusing class: Degree to which, existing class can be reused for creating a new ontology.

Reusing slot: Degree to which, existing slot can be reused for creating a new ontology.

Reusing instance: Degree to which, existing instance can be reused to create a new instance of a class.

Reusing data: Degree to which, existing data can be reused to populate instances a new ontology.

Reusability allows the reuse of existing ontologies, classes, slots, and instances. Reuse of ontologies as a whole facilitates the creation of homogenous systems on the Internet. Reuse of classes allows the faster development of new ontologies, which facilitate reuse and knowledge sharing in general. Then, slots and instances contribute to the low cost of ontology development since one does not have to reformulate the data entry. To conclude, this type of sharing within the organization or industry decreases the developing cost of ontologies and promotes the use of existing ontologies.

III. Data Storage

XML: Degree to which, the ontology can be stored in the XML format.
Database: Degree to which, the ontology can be stored in the database format.
Schema: Degree to which, the ontology can be stored in the Schema format.

We use data storage because one should be able to store ontology in the formats such as XML [11], database, and schema. Hence, it is important for ontology editors to allow importing and exporting of ontologies in other languages such as XML [11], Oracle [12], and RDF [11]. Also, ontologies available in several languages increase their use and result in the wider acceptance among the developers.

IV. Complexity

Installation Process: Number of steps and time involved in the installation process
Platform Requirement: Degree to which the editor is capable of running on different platforms.
Interface: Degree to which the editor offers user friendly interface.

Complexity is considered as an aspect because one should be able to install the given tool without much difficulty. The program should have an executable file that one can double click for an easy installation. Also, It is important to know various platforms that the editor supports and if the editor is user friendly in a sense that it supports easy browsing of the ontology in the development as well as in the presentation mode.

V. Association

Database connectivity: Degree to which the editor is interoperable with databases.
Macro capability: Degree to which the editor allows one to add Macros to the ontology.

It is important for the ontology editors to interoperate with databases. This allows one to store ontologies in a centralized location, which facilitate reuse and knowledge sharing with different departments of the organization. Also, it is possible to create homogeneous systems within the enterprise with the wide use of standardized ontologies. Macros are important because they ensure proper data entry and formatting. The developers can create macros to facilitate data entry from the users for consistency purposes. In addition, macros can be used to enhance data presentation when viewed by the users.

VI. Scalability

Add new super class: Whether the editor allows the addition of a new super class to an existing ontology.
Add new sub class: Whether the editor allows the addition of a new sub class to an existing ontology.
Modify super class to sub class: Whether the editor allows the modification of a super class to sub class.
Modify sub class to sub class: Degree to which the editor allows one to add Macros to the ontology.
Delete existing super class: Whether the editor allows the deletion of an existing super class.
Delete existing sub class: Whether the editor allows the deletion of an existing sub class

Scalability is a criterion that helps to improve the ontologies by adding, changing, deleting classes, relations, slots and instances from time to time. Therefore, a built-in method is required for the editor to allow changes to the ontology as the domain knowledge is constantly changing. This method could allow the addition of more classes, properties, relations, and instances to an existing ontology. On the other hand, the method may or may not allow the change/deletion of classes, properties, and relations if they have been populated. Thus, it is crucial that one is familiar with method that each editor has before developing an evolving ontology.

VII. Resilience

Undo and redo added new super class: Whether the editor allows undoing and redoing of the addition of a new super class.

Undo and redo added new sub class: Whether the editor allows undoing and redoing the addition of a new sub class.

Undo and redo modification of the existing super class: Whether the editor allows undoing and redoing the modification of an existing super class.

Undo and redo modification of the existing sub class: Whether the editor allows undoing and redoing the modification of an existing sub class.

Undo and redo deletion of the existing super class: Whether the editor allows undoing and redoing the deletion of an existing super class.

Undo and redo deletion of the existing sub class: Whether the editor allows undoing and redoing the deletion of an existing sub class.

Due to the ever-changing domain knowledge, ontology is expected to evolve over time. Therefore, it is essential that an editor have the capability to handle effects of changes that take place over time. This mechanism can reduce many unintentional errors and can facilitate the verification task at the later stages of the ontology development. Also, this allows the developer to control modifications and make accurate decisions.

VIII. Reliability

Validation: Degree to which the editor prevents the developer to add similar class names and slot names in the ontology.

Verification: Degree to which the editor verifies and corrects the minor errors made during the ontology development.

Reliability is used since developer is bound to make mistakes at the first ontology development effort. So, ontology construction tool has to assist the developer in the ontology construction process by giving more information regarding the developer's actions such as why an error occurred, and what can be done in the future to accomplish the given activity without the errors. In addition, the editor should provide the verification capability, which checks the ontology for errors and makes automated corrections to ensure consistency. Thus, it is important that the editor provides sufficient assistance in explaining the cause of errors and provides the implications of the suggested corrections.

IX. Robustness

Transparency: Degree to which the super class slot values are transparent in all of its sub classes.

Relationship: Degree to which the editor allows representation of various relations between classes to facilitate efficient query searches.

User defined type: Degree to which the editor allows declaration of user defined types.

Form formatting capability: Degree to which the editor allows formatting of the form for data entry and data presentation purposes.

Robustness is included since it allows the creation of high quality ontology. Due to the inheritance mechanism, a change in one part of ontology will be transparent in the other parts. For example, changes made to a super class are transparent in all of the sub class instances. Thus, to reduce the errors of the change, the editor should provide the transparency from a super class to a sub class to ensure consistency. Additionally, robustness is achieved through strong relations between classes since it is important to do complex query searches. User defined types contribute to the robustness of the ontology since one can add slots based on original ontology requirements. Lastly, to ensure robustness, one should be able to format forms to allow effective data entry and presentation.

X. Learnability

Pre-requisition knowledge: Amount of knowledge required to construct a basic ontology.

Helping manual: Utility of the helping manual in the learning process.

Time length: Amount of learning time required to create a basic ontology.

Learnability is included since it is important to know the type of prerequisite knowledge needed to effectively create a full-fledged ontology. It is also important to know the time required to learn a particular ontology editor since every project has a deadline. Moreover, it is important to know if the user manual associated with the particular editor played a crucial role in the learning process since its function is to assist the user learning process.

XI. Availability

Access requirement: Degree to which, the editor is accessible online.

Price: Amount of money involved in the purchase of the editor.

Since one should be able to access a given editor without much difficulty, we used availability. It should be easy to access the tool either from the Internet or from a retail store without much effort. Also, it is important to compare price before acquiring the tool since every project has limited resources.

XII. Efficiency

Model Browsing: Degree to which the editor allows easy browsing of the ontology model.

Query capability: Degree to which the editor allows efficient query searches.

Design tools: Degree to which the editor provides the developer with easy to use design tools.

Efficiency was used because it is important to consider model-browsing capability for ontology creation purposes. It is important to move from one action to the other since ontology creation is an iterative process. For instance, it is necessary to add relations and properties successively between classes.. In addition, it is essential that users be able perform efficient query searches otherwise the ontology construction tool may not be so useful among the user community. It is also vital that the design tools are easy to use, so that, developers can easily learn and use the tools to represent the domain knowledge in an effective way.

XIII. Visibility

Model creation using external program: Degree to which the editor makes available the capability to draw a visual model of the ontology by using external program as plug-in.

Model creation using internal program: Degree to which the editor has a built-in capability to draw a visual model of the ontology.

Conversation to HTML pages: Degree to which the editor permits the conversion of ontology to HTML Pages.

Visibility is included since it is important to represent the ontology in a visual mode. Visual presentation of ontology facilitates ontology presentation and allows the users to easily understand the ontology content.

3 Scenarios and Test Cases

The development started with the creation of Treatment ontology with the appropriate test case. We developed Used-car dealership ontology with the appropriate test data. Research ontology was created next by applying a suitable test case. Then, University enrollment ontology was constructed with the predefined test data. Lastly, Order taking ontology was developed with the appropriate test case. The following scenarios were used because they were appropriate for ontology construction. For instance, a patient receiving treatment for sickness from a doctor is universal. A student enrolling in the university is also similar from university to university. So we developed ontologies that facilitate reuse of the domain knowledge and benefit the research community.

The list of following scenarios and test cases are:

1. A patient receives treatment for his/her sickness from a doctor.

Test case: Mr. Robin went to Doctor Richard Hasel for sudden pain in his chest. Dr. Richard observed him carefully and he found blood pressure high and then sent him for testing to clarify his doubt. Upon getting the result of the test, Mr. Richard confirmed that Mr. Robin had heart congestion. In this instance, he prescribed aspirin as a medication.

2. A used-car dealership keeps track of their cars sale status.

Test case: A car of category sedan, model A6, make Audi, year 2001, mileage 1000, and price 20,000 SEK is not sold.

3. A student enrolls in a program of a university.

Test case: Student Ahsan Morshed was admitted to Graduate program of Master's of Science in Computer Science on 30-06-2004 by admission officer Anderz Delge.

4. A commercial website takes order for its merchandise.

Test case: Customer Ramanjit Singh placed an order for a book called 'A fable of leadership through the storytelling' and 2 copies of movie called 'Training day' from a sales representative called John Doe on 2004-09-26.

5. A scientist looks for past research papers for his/her research.

Test case: Mr. Fox Hans works in IBM Inc. He wrote a research paper in the field of Knowledge management. The name of the research paper is "The evaluation of knowledge management in the Enterprise system". It was published in IT-today journal of Royal Institute of Technology.

4 Ranking Technique

In the preceding section, we defined the criterion where each of the criteria consisted of sub-criteria. Now, we propose a straightforward ranking technique [10] for ontology tools, which is as follows: Let there be n criteria $x_1, x_2, x_3, ...x_n$ having weights $w_1, w_2, w_3, ...w_n$ respectively. These weights represent the degree of importance of the criteria in building the ontology and the user can assign them according to the level of importance he gives to each criterion. The overall rank of an ontology tool can be represented as the weighted average of all aspects of criteria, that is

$$R = \frac{\sum_{i=1}^{n} w_i x_i}{\sum_{i=1}^{n} w_i}$$

Where, the ranking of each criterion can be represented as

$$x_i = \frac{\sum_{j=1}^{k} w_{ij} x_{ij}}{\sum_{j=1}^{k} w_{ij}}$$

Where,

$0 \le x_i \le 1$, and $0 \le x_{ij} \le 1$

k = number of sub-criteria in a criteria i

x_{ij} = ranking of jth sub-criteria of a criterion i.

w_{ij} = weighting factor of jth sub-criteria of criteria i.

Using the above-mentioned formulas, the user can evaluate ontology construction tools and select the one that suits best for his/her requirements. Finally, we evaluated Protégé-2.1.2, Metis-3.4 and OntoEdit Free tools based on the proposed set of criteria and ranking technique.

5. The result and discussion

Functionality: All three editors provide similar functionality. One can add, change and delete concepts, slots, and relations. However, Metis-3.4 does not allow the addition of instances to concept and therefore, it is not possible to inference ontology.

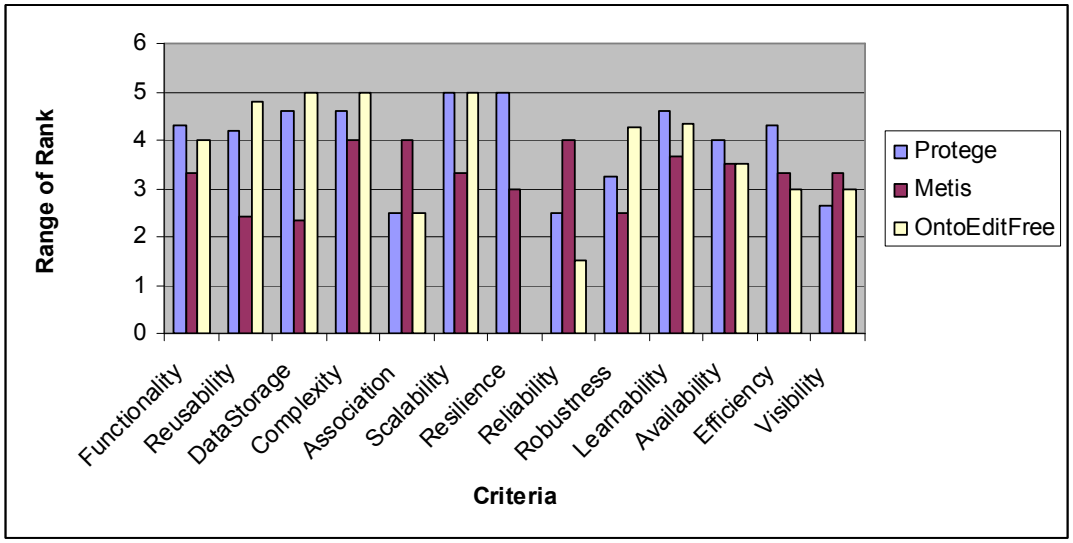

Fig. 1. The result of different criteria.

Reusability: Protégé-2.1.2 and OntoEdit Free has similar reusability features. One can reuse ontology, class, slot, and instance. However, there is minor difference between the two, Protégé-2.1.2 does not support the import of data to populate ontology and OntoEdit Free does. Metis-3.4 alternatively allows only the reuse of ontology and class.

Data Storage: The Protégé-2.1.2 and OntoEdit Free allow one to store the ontology in the XML, Database, and RDF schema format. In contrast, Metis-3.4 supports the XML and Database storage, but not in form of RDF schema.

Complexity: The installation process of Protégé-2.1.2 and OntoEdit Free is very simple. One can download the software from the Internet and install it in the matter of few minutes. Protégé-2.1.2 runs only on Windows but OntoEdit Free is compatible on both Windows and Linux with Java Runtime Environment. Metis-3.4 on the other hand runs on most of the platforms that have JDK 1.3 version installed.

Association: Protégé-2.1.2 and OntoEdit Free supports variety of different databases for ontology storage. However, Metis-3.4 supports only Microsoft SQL Server 2000[5], Oracle 8i, and Oracle 9i [5, 12]. Protégé-2.1.2 and OntoEdit Free do not support macro-adding feature but Metis-3.4 does.

Scalability: The Protégé-2.1.2 and OntoEdit Free are very friendly when it comes to improving or scaling the ontology. One can add, change and delete classes, slots, and relationship with few mouse clicks. Metis-3.4 on the contrary is less-user friendly when one wants to make changes to the ontology. It allows the addition of super class and sub class but it does not allow one to change the super class or sub class without first deleting the given super or sub class.

Resilience: Protégé-2.1.2 fully supports the undo and redo actions taken by the user. Metis-3.4 allows one to merely undo the actions taken by the user and OntoEdit Free does not allow one to either undo or redo actions.

Reliability: The Protégé-2.1.2 and OntoEdit Free are reliable editors since they do not permit the use of similar class and slot names. Nevertheless, they do lack one important feature; they do not have the ability to automatically verify if the ontology is free from errors and inconsistencies. Metis-3.4 alternatively allows one to add similar class and slot names, but it does have the verification mechanism, which enables it to identify and correct errors.

Robustness: Protégé-2.1.2 and OntoEdit Free are very similar when it comes to robustness. They both support the transparency principle, variety of relationship building, and form formatting capability. However, they do not enable the user to define their own attribute types. In Contrast, Metis-3.4 is very different from it counter parts when it

comes to robustness. Transparency principle is questionable since one cannot add instances to objects and one cannot format forms. Nevertheless, there is a possibility to add relationships as well as own user defined types.

Learnability: The Protégé-2.1.2 and OntoEdit Free are two easy to use tools. They come with detailed user manual, which makes it an easy task to learn and construct a basic ontology in one or two days. Metis-3.4 in contrast is an extensive tool with many components. So learning Metis-3.4 is a long process and it is estimated that ontology construction time is 3 to 4 days with Metis-3.4 [5].

Availability: Protégé-2.1.2 can be obtained free of cost. OntoEdit Free can be used for two days without any cost, but after the expiry of the trail version, there is a need to pay 890 Euros [4] to purchase the full fledges OntoEdit editor. Metis-3.4 on the other hand is more expensive. It comes with a high price tag of 25,000Kr [5].

Efficiency: Protégé-2.1.2 does not have model browsing capability but it offers one to do simple query searches, and it has user-friendly design tools. OntoEdit Free and Metis-3.4 in contrast have impressive model browsing capability. However, there is one important difference between them, OntoEditFree can inference with a plug-in but Metis does not have such capability.

Visibility: The Protégé-2.1.2 does not offer the visual model building facility. There is a need to draw the visual model by using an external program as a plug-in. Metis-3.4 and OntoEdit Free on the other hand have built-in visual model generation capability. Lastly, Protégé-2.1.2 and Metis-3.4 supports the conversion of ontology to HTML pages but OntoEdit Free does not.

6 Conclusion

In this paper, we proposed ontology evaluation criteria and a ranking technique to evaluate ontology editors. After our evaluation, we reached the following conclusion: Protégé-2.1.2 can be used to construct small-sized ontologies and OntoEdit, with its full functionality, can be used to develop medium to large-sized ontologies. Metis-3.4 is good for enterprise architecture modeling. The proposed framework is very flexible to cope with the specific needs of ontology developers and further research can be conducted to enhance the evaluation criteria and ranking technique.

So, we suggest that more technical aspects, such as the import and the export of ontologies in other languages such as RDF [11], DAML+OIL [11], XML [11], and other database languages should be included in the evaluation criteria.

Acknowledgement

Thanks to Prof. Fausto Giunchiglia of DIT, University of Trento, Italy, Prof. Trevor Wood-Harper of School of Informatics, University of Manchester, and Prof. Paul Johannesson of Royal Institute of Technology, Sweden for their valuable suggestions.

References

[1] T. R. Gruber, *A translation approach to portable ontologies*. Knowledge Acquisition, 5(1993), 199-220.

[2] Y.Sure, Onto-To-Knowledge-Ontology based Knowledge Management Tools and their Application, German Journal Kuentliche Intelligenz, Special Issue on Knowledge Management(01/02), 2002.

[3] Stanford medical informatics Home page, at URL: http://protege.stanford.edu

[4] Ontoprise®GmbH (1999), Onto Edit tutorial Homepage [Online], at URL: http://www.ontoprise.de/documents/tutorial_ontoedit.pdf

[5] Computus (1985) Home page [Online], at URL: www.computus.com

[6] L.Stojanovic and B.Motik, Ontology evolution within ontology editor, 13th International Conference on Knowledge Engineering and Knowledge Management EKAW02, Sigüenza (Spain)

[7] M.Denny, Ontology Building: A survey of Editing Tools, Nov 6, 2002.

[8] Jürgen, York Sure, White paper: Evaluation of ontology-based tool, 13th International Conference on Knowledge Engineering and Knowledge Management EKAW02, Sigüenza(Spain).

[9] P.Lambrix and A.Edberg, Evaluation of ontology merging tools in bioinformatics, Proceedings of the Pacific Symposium on Biocomputing PSB03, 8:589-600, Kauai, Hawaii, USA, 2003.

[10] D.Lane's Home page [online] Accessed on Jan 2, 2005 at URL:http://davidmlane.com/hyperstat/A15885.html

[11] World Wide Web.http://www.w3.org/

[12] Oracle. http://www.oracle.com/index.html

[14] M.A.Musen (2000), *the Evolution of Protégé: An Environment for Knowledge-Based Systems Developmen*t, Retrieved October 10, 2004 at URL: http://www.smi.stanford.edu/pubs/SMI_Reports/SMI-2002-0943.pdf

[15] N.Noy, R.W. Fergerson, M. A.Musen (2000), The knowledge model of Protégé-2000: combining interoperability and flexibility, Retrieved October 11, 2004 at URL: http://www-smi.stanford.edu/pubs/SMI_Reports/SMI-2000-0830.pdf

[16] Search Engine Yahoo: www.yahoo.com

[17] Amazon. www.amazon.com

[18] A.Morshed and R.Singh, Master thesis (No: 05-x-223) Evaluation and Ranking of Ontology Construction Tools, Royal Institute of Technology, 13th Jan, 2005