

Survey on Web Ontology Editing Tools

Sabin Corneliu Buraga, Liliana Cojocaru, Ovidiu Cătălin Nichifor

Faculty of Computer Science, "A.I.Cuza" University of Iasi, Berthelot Street, 16 Iași, Romania

Phone: (+40) 232 201090, E-Mail: {busaco, diaboliiq, cilioo}@infoiasi.ro, WWW: <http://www.infoiasi.ro/~busaco/>

Abstract – *The paper presents an original survey on Web ontology editing tools, in respect with Semantic Web actual technologies. The material is focused on features that these tools might expose: granularity of expressivity, Web standards compliance, reasoning support, provided APIs and, not least, the interoperability issues. The comparative study is used a specific test ontology regarding the main characteristics and features of Semantic Web applications.*

Keywords: *Semantic Web, ontology, OWL, Knowledge Representation, Comparison, Software Tool*

I. INTRODUCTION

The Semantic Web can be considered as a mesh of linked information (resources) or a globally linked distributed database [4, 5, 7]. The main goal is that data should easily be processed by machines. Since the data has a lot of representations, meaningful in some contexts, but not in others, there are difficulties to use it on a large scale, because there is no global system for publishing data in such a way as it can be easily processed by anyone.

This is where the Semantic Web interferes. To give an abstract view of Semantic Web, let us imagine it as a huge engineering solution which addresses data publishing in a purpose-oriented form, thus enabling semantic associations between information.

To improve, extend and standardize the system, many languages, publications, tools and so on have already been developed or proposed by the World-Wide Web Consortium [21], the international organism that coordinate the Web.

One important aspect of the actual Semantic Web directions of research is the use of ontologies as a suitable representation of knowledge available on the Web. A noticed difficulty is the use of a proper tool for modeling, editing and exploiting various ontologies in the context of Web space.

The goal's paper is to offer a comprehensive and comparative view regarding the actual tools focused on Web ontology editing. For this, in section II, a short

presentation of Semantic Web technologies is provided, including several important aspects concerning ontologies. Section III is dedicated to our survey on Web ontology editing software tools. We will overview and describe several of the existing ontology-oriented applications, such as *OilEd*, *OntoEdit*, *pOWL*, *Protégé*, and *SWOOP*. The results of the comparative case study are also provided by this section. The paper is ended with conclusions.

II. SHORT PRESENTATION OF SEMANTIC WEB

In order to represent data, Semantic Web is using *Uniform Resource Identifiers* (URI) that address Web resources.

In addition, for processing metadata (data about data), *Resource Description Framework* (RDF) [11] model can be used, in a standardized way. The used metadata format should permit to reason about data. The RDF is intended to be used to capture and state the conceptual structure of information offered in the Web. The RDF assertions – triples of URIs – can be viewed as a data model for describing machine processable semantics of data to build the infrastructure for that Sir Tim Berners-Lee, the creator of WWW space, called *Semantic Web* [2].

RDF is a standard based on *Extensible Markup Language* (XML) [3] meta-language, with the meaning to map the information directly and unambiguously to a model, a model which is decentralized, and for which there are many generic parsers already available. This means that when we develop an RDF application, we know which bits of data denotes semantics of the application (e.g., business logic), and which bits represent just syntactic conventions.

To gain benefit of the full potential of Semantic Web, the main idea is to start publishing data as RDF, a common data annotation and representation, with many existing applications in different areas [7-9].

The first "layer" of the Semantic Web above the syntax discussed above is the simple data-typing model. A *schema* is simply a document or piece of code that controls a set of terms in another document or piece of code. Focusing on the aspects mentioned above, in order to restrict the syntax of XML applications there has been proposed a language called XML Schema [21]. XML Schema in conjunction

with RDF may be useful for creating data-types in order to stored different kinds of information.

A simple data-typing model for RDF is *RDF Schema (RDFS)* [6] which can be able to model simple ontologies. In this way, Web resources are possessing class hierarchies and properties with ranges and domains. This allows us to quickly build up knowledge databases in RDF. Also, RDFS also contains a set of properties for annotating schemata, providing comments, labels, making them easily to be understood.

By providing documents with respect to those suggestions *a priori* mentioned, we can create what we call *ontologies* – documents describing abstract, inference and logic information representation [7-8]. The inference means the capability to derive new data from data that you already know. In a mathematical sense, querying is a form of inference (being able to infer some search results from a collection of data, for example). Inference is one of the driving principles of the Semantic Web. Also, in order to Semantic Web to become expressive enough to help us in a wide range of situations, it will become necessary to construct a powerful logical language for making inferences (e.g., predicate logic) [1].

Ontologies provides in depth properties and classes such as inverses, unambiguous properties, unique properties, lists, restrictions, cardinalities, pair wise disjoint lists, data types, and so on. Ontologies are often able to provide an objective specification of domain information by representing a consensual agreement on the concepts and relations that characterize the manner knowledge in that domain is expressed. This specification can be the first step in building semantically-aware information systems to support diverse enterprise, government, and personal activities.

Ontologies may vary not only in their content, but also in their structure and implementation. Among the features and requirements that certain ontology might compel, we can enumerate [4, 7]:

- Level of description – building an ontology, we might reveal different aspects to different practitioners. Describing knowledge starts from simple lexicons or controlled vocabularies, to categorically organized thesauri or taxonomies where terms are related hierarchically and can be given distinguishing properties that can define new concepts and where concepts have named relationships with other concepts;
- Conceptual scope – ontologies can be used in different domains describing specific information (e.g., medicine, aeronautics, etc.); also, there are upper level ontologies, such as *Suggested Upper Merged Ontology (SUMO)* [13], describing the basic concepts and relationships invoked when information about any domain is expressed in natural language;

- Instantiation – this aspect concerns populating the ontology with instances or individuals that manifest that terminological definition. This extension can be separated in implementation from the ontology and maintained as a knowledge base (*ABox* component in terms of description logic [1]);
- Specification language – a number of possible languages can be used, including general logic programming languages like Prolog. More common, however, are languages that have evolved specifically to support ontology construction, for example OKBC (*Open Knowledge Base Connectivity*) model and KIF (*Knowledge Inference Format*). These proposals have become the bases of other ontology languages. There are also several languages based on a form of logic thought to be especially computable known as description logics [1]. A standardized language is *Web Ontology Language (OWL)* – details in [4], [7] and [12]. When comparing ontology languages, what is given up for computability and simplicity is usually language expressiveness. A language needs only be as rich and expressive as is necessary to represent the fine distinction of knowledge that the ontology's purpose and its developers demand.

III. SURVEY ON ONTOLOGY EDITORS

Requirements

When starting out on an ontology project, the first and reasonable reaction is to find a suitable ontology software editor. Our main concern must include the provided capabilities like ontology versioning (the project development often involves various ontologies – external as well as newly in-house developed), mapping and linking, comparing, merging, reconciling and validating, converting them into other forms (such as XML Schemas, database schemas, and others).

For this paper, we considered the following categories covering important functions and features of the software, such as:

- *Application* – the two main developing directions include the commercial products designed exclusively for building ontologies in any domain, and academic or government funded projects in order to investigate the technical application of ontologies. Some editors are intended for building ontologies in a specific domain, but are still capable for general-purpose ontology building regardless of content focus;
- *Methodology* – the process of building ontologies is a high-cost process. Methodologies (supported by certain tools) are essential to:
 - Help the developer to mark and annotate a concept;
 - Modularize (and align) the ontologies;

- Ensure relevance and avoid over-elaboration (this degrades the level of abstraction);
- Verify the ontology fits its purpose and if it is reusable in other contexts.

Details about the actual proposed methodologies can be consulted in [4], [8] and [9].

- *Interoperability* – ontologies are for knowledge sharing. They are intended to serve as consensual rallying points to exchange and interpret information. The wider the range of applications and other ontologies that can (re)use a certain ontology, the greater is the utility. Also, a tool must provide support for diverse ontology representations (importing and exporting ontologies in various language serializations);
- *Usability* – in addition to the features already mentioned, ontology editors vary considerably in their overall feel to the user. We must not focus on attempt to compare editors under use, but for the features, plug-ins that a tool may provide – management, manipulation of ontology’s interlinking concepts and relations are essential. Because many ontology models support multiple inheritances in the concept hierarchies and relation hierarchies, keeping the associations straight is a challenge. The standard approach is the use of multiple tree views with expanding and contracting levels. A graph presentation is less common, although it can be quite useful for actual ontology editing functions that change concepts and relations.
- *Inference* – while ontologies themselves can be treated as standalone specifications, they are ultimately used to help answer queries about some information. Certain ontology editing applications offer reasoning services to be used in order to obtain a (new) inferred knowledge model.

Tools under Evaluation

This section provides details regarding the ontology editing tools considered for our survey.

We focused on the following applications:

- *pOWL – Semantic Web Development Platform* [17] has the aim to deliver a PHP and Web-based ontology editing and management solution to the open source community. The application is a Web-based one and supports viewing, editing of RDFS/OWL ontologies of arbitrary size. pOWL currently offers an RDQL query builder as well as a full-text search for literals and resources. pOWL offers support for extensions, but unfortunately still lacking exhaustive documentation on this. All functionality is accessible by a solid application programming interface (API) and the access could be authenticated. Models are stored in database tables (e.g., MySQL, PostgreSQL, Oracle, etc.) and only those parts of the model are loaded into main memory which are actually needed – this assures a fast response. The ontology can be imported from

and exported to different RDF syntaxes (XML, N3, N-triples). Also, the results can be available as HTML pages. pOWL is freely available under the terms of GNU General Public License (GPL).

- *SWOOP (Semantic Web Ontology Overview and Perusal)* [20] is a simple, scalable, hypermedia-inspired OWL ontology browser and editor written in Java. Other familiar web-browser look and feel features include an address bar to load ontological entities, history buttons and bookmarks. SWOOP has been designed in-keeping with the W3C OWL recommendations and has reasoning support (Pellet, an OWL inference engine). Another facility is the multiple ontology environment, whereby entities and relationships across various ontologies can be seamlessly compared, edited and merged. All ontology editing in SWOOP is done inline with the HTML renderer, using different color codes and font styles to emphasize ontology changes, e.g. diverse representations for added, deleted or inferred axioms. Undo/redo options are provided with an ontology change log and a rollback option. SWOOP could import ontologies from OWL, XML, RDF and text formats. These formats could be used to save the edited ontologies. The overall tool architecture is based on MVC (Model-View-Controller) design pattern.
- *OntoEdit* [16] is part of OntoStudio, based on IBM Eclipse framework. OntoEdit is a development environment for ontology design and maintenance. It supports multilingual development, and the knowledge model is related to frame-based languages. OntoEdit is based on an open plug-in structure. Every plug-in provides other features to deal with the requirements an ontology engineer has. Data about classes, properties, and individuals may be imported or exported via different formats, such as OXML, F-Logic, RDF/RDFS, OWL and other formats supported by WebDav. The professional version of the tool is available as a commercial product.
- *Protégé* [18] is a free, open-source Java-based platform that provides a growing user community with a suite of tools to build domain models and knowledge-based applications with ontologies. Protégé implements a rich set of knowledge-modeling structures and actions that support the creation, visualization, and manipulation of ontologies in various representation formats. Protégé can be customized to provide domain-friendly support for creating knowledge models and entering data. Further, Protégé can be extended via a plug-in architecture. Various facilities could be programmatically available via the provided Java API. Protégé gives support for building the ontologies that are frame-based, in accordance with the Open Knowledge Base Connectivity protocol (OKBC). Also, the tool provides support for OWL ontologies. A special plug-in can be used to generate graph representations of the editing ontologies.

- *OilEd* [15] is an ontology editor allowing the user to build ontologies using DAML+OIL [8], the language that inspire the actual OWL standard. The current versions of OilEd do not offer a full ontology development environment, but provides enough functionality to allow users to build ontologies and to demonstrate how we can use the FaCT reasoner to check those ontologies for consistency. Data can be imported from DAML+OIL, OWL RDF/XML, and OIL text formats. OilEd can save ontologies as DAML+OIL documents only. OilEd is available as an open-source Java project under the GPL license.

Comparative Case Study

The fundamental idea of this study is to observe the major variations, between the presented tools – lack or support, and at the end, we shall conclude which one we found more interesting, more easy to use, more precise, and appropriate to help developing high-quality Semantic Web ontologies.

In order to create a complete comparative study of the tools under survey, we proposed a simple test of modeling, visualization, editing, import/export on every environment with a given test ontology.

The selected test ontology embodies an ontology instrument, exposing its characteristics. We started working on this ontology with Protégé, one of the instruments under survey, which, by the way, we found it to be a very expressive and easy to use tool, helping us to develop the ontology as quickly as efficient.

The developed ontology concerns the characteristics of each tool (module list, feature list, support for actual W3C standards, code availability, data representation, API support, platform, etc.). A fragment regarding several properties associated to *Tool* class is depicted in figure 1.

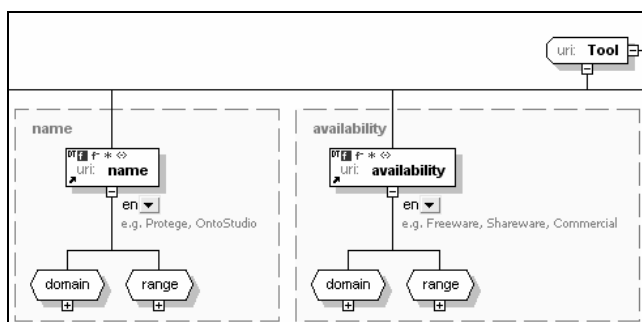


Figure 1. Ontology regarding Semantic Web ontology editing tools (fragment)

In the next paragraphs, we will provide an ontology management walk-through on the tools presented in the previous section.

The test with pOWL failed from the beginning, because the instrument couldn't import the newly created ontology though the export was made in a RDF/XML format which

is a standard. Since the pOWL tool provided us ontology construction based on text RDF/XML source, we tried to force loading the ontology directly from the text source. But, unfortunately, this test also failed. These incompatibility issues made us think that there are some problems regarding the implementation of W3C standards and we ended this tool test at this point. Since pOWL project is only at it's beginning, we hope that these issues will be solved as soon as possible, because this instrument seems quite fine and also has some great features – Web application, on-line help and support (e.g., the user can always see the help or samples associated with the RDF, OWL declarations) and easy to use.

SWOOP recognized our test ontology individuals and provided us with a clear representation. The HTML-like view helped us to quickly navigate between components, which it can be considered a plus – see figure 2. Maybe the most important feature that this utensil presents was the versioning system control of the ontology, in other words, on each change of the original document we first had to apply changes before they can be reflected on the ontology. Anytime, when errors were discovered, using this version control we could rollback to the previous good state of ontology. Another good aspect was the partitioning module that splits the ontology in well defined class/property hierarchies allowing us to see the out modeling delimitations/gaps of our test ontology. The SWOOP query tool is not so advanced but, for simple queries, it does it work well and really quick.

Also the backend embedded reasoner debugging feature helps tracking errors in an elegant manner. The built-in ontology examples database (FOAF, AKT-Portal, NATO, OWL-S Grounding, etc.) included by SWOOP offers helps while you are at studying step. Although the expressivity offered was not too high and the set of restrictions you can enforce to objects was a little bit limited, this tool is a good instrument for developing small to medium ontologies which can afterwards be exported to more sophisticated software, because the set of export formats is quite large.

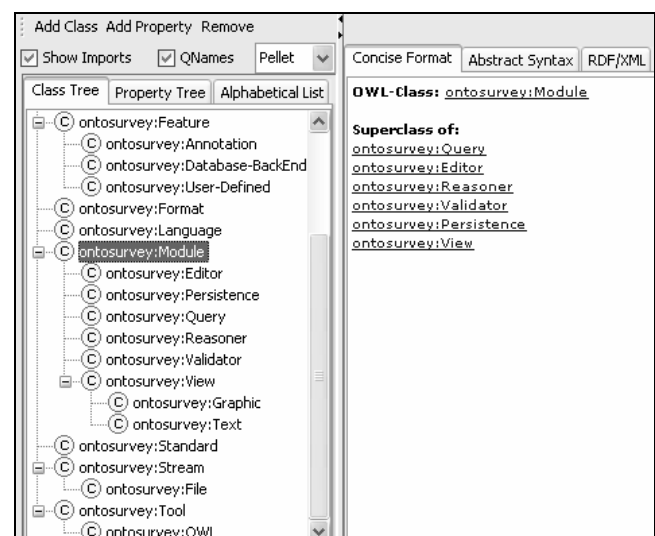


Figure 2. Browsing the test ontology within SWOOP

OilEd is a simple ontology editor that supports the construction of OIL-based ontologies. The basic design has been closely influenced by similar tools such as Protégé and OntoStudio. OilEd inherits only the main facilities, the rest being a little bit restricted. Because of the fact that import/export formats (Simple RDFS, DAML+OIL) are rather limited, the tool is less flexible, comparative with other instruments, such as Protégé. We first had to save our ontology as text RDF/XML source format and then to import it into OilEd tool. We find it to be kind of difficult when dealing with relation/attribute declaration and applying some restrictions. From the view perspective, the hierarchy viewer within OilEd is rather primitive. In particular, it's not very good at dealing with situations when there are cycles in the hierarchy (e.g., lots of equivalent concepts). We also noticed that simple metadata for classes can be recorded. The creation date and creator are automatically logged by OilEd and cannot be changed via the interface. Additional information about the status of the class can be added as free text. OilEd itself does no reasoning at all, but it relies on FaCT reasoner which must be additionally installed by the user. To conclude, OilEd still lacks many features that would be required of a fully-fledged ontology development environment.

OntoStudio (OntoEdit) offers the power and flexibility of Eclipse platform, embedding main standards, usability and scalability. As we are used to Eclipse "views", OntoStudio contains a schema view which allows searching within the concept hierarchy. This is especially useful when modeling large ontologies. Also to be noticed, the (directed) graph view of ontology (i.e. concepts are represented by nodes and the edges substitute the attributes or relations that a concept might include) gives us the ability to get a general view of the entire architecture of ontology. When it comes to import/export formats, the tool supports the standard ontology formats and, more important, database schema import, making the conversion from static architecture to flexible Semantic Web ontologies easier. We can note that the other studied tools do not offer this feature. The real power of this tool is the object-oriented modeling-like perspective, where we can declare and use the concepts and their attributes while creating different relations in a well-designed manner. The best thing about this tool is that it doesn't restrict the user to perform only some specific RDF/RDFS/OWL-like declarations. Our impression on this instrument is that it best fits to the necessities of large ontology development. We found this tool to be an advanced and powerful ontology modeling framework and we conclude that it is best suited for enterprise engineering.

Protégé, at first sight, is more closed to the users which are not accustomed to the development of ontologies. With the provided help via wizards it gave us a head start for creating new ontologies and we managed to model very quick and easy our test ontology. This instrument is more Graphical User Interface (GUI)-oriented platform and with a few clicks it reveals the pleasure working with a simple yet complex Semantic Web tool. Since we mentioned the

GUI, we have to state that it has an improved look and feel consistent with the underlying ontology model. With the help of the rich user-interface, it hides all the logic like constraints, giving opportunity to user to focus on quick development. This does not mean that it gives us a limited palette of components we can access. Contrary, we can model complex resources without loss of expressivity. The query module is more complex and user-friendly than others and, probably, the most important are the examples presented first, in natural language, and then the "how-to" of query modeling using different GUI components. The last but not the least important feature is the Java source code generation plug-in which places ontology interpretation and application development one step further towards architecture focus. A screen-capture of Protégé interface is available in figure 3.

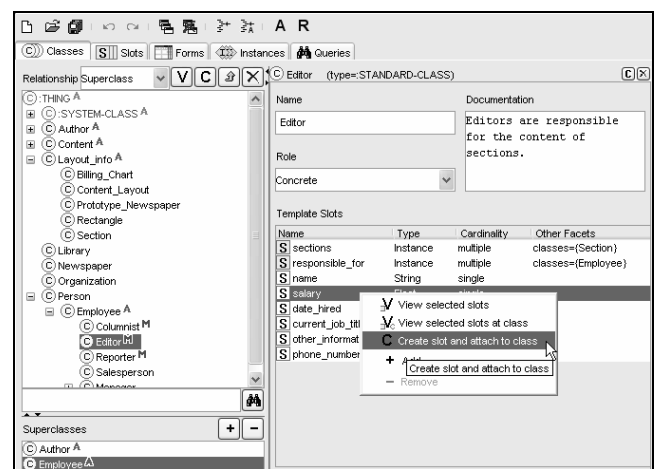


Figure 3. Ontology editing and management with Protégé

Survey Results

At the end of our study, we noticed that every tool has some nice, but particular, features and we consider that choosing the right tool is related with the type of user (beginner/expert), with the type of ontology (small/large) and, not least, with the features that an developer might require (for example, graph view, quick development, using certain data formats, etc.).

Keeping these observations in mind, our opinion is that for "home use" (individual or academic study) the best suited tools are Protégé and SWOOP, while – if we need to develop large ontologies with complex architectures – best solution is provided by OntoStudio, a tool for professional designers.

IV. CONCLUSIONS

In this paper, we had drawn several remarks and results concerning a survey on Web ontology editing tools. The tested tools have different characteristics regarding ontology modeling and thus different features and perceptions. Some of them are intended for simple ontology development, others are complex pieces of

software that are capable of general-purpose ontology building regardless of content focus. We also distinguished plug-in able utensils that can be extended an improved.

Also, a test ontology regarding ontology management software application was designed in order to study different characteristics and features provided by certain instruments. This specific ontology was ported to every instrument to distinguish the level of expressivity and the easiness of semantic modeling that every tool offers.

From our knowledge, only one study – see [10] – was performed. The most important difference is that our survey concerns practical deployment of a test ontology via a suite of tools and is focused on some important technical details and features not revealed by other papers.

We assume our survey can be important for all practitioners from the domain of Semantic Web technologies, as well for the interested specialists belonging to knowledge management, artificial intelligence, and computational linguistics areas. Designing accurate ontologies is a difficult and time-consuming task. Thus, a good developing tool helps to amortize this effort.

Further directions of interest will focused on a broader range of ontology editing tools (such as Altova SemanticWorks, not covered by the actual survey) and Semantic Web frameworks (for example, Jena [14] or Redland [19] platforms), especially in the case of knowledge querying and rules modeling – details in [4].

Also, we intend to develop more complex test ontologies to be used in the future versions of the survey.

REFERENCES

- [1] F. Baader et al. (eds.), *The Description Logic Handbook*, Cambridge University Press, 2003.
- [2] T. Berners-Lee, J. Hendler, O. Lassila, „The Semantic Web”, *Scientific American*, 5, 2001.
- [3] T. Bray et al. (eds.), *Extensible Markup Language (XML) 1.0 (Third Edition)*, W3C Recommendation, Boston, 2004: <http://www.w3.org/TR/REC-xml>
- [4] S. Buraga, *XML Technologies* (in Romanian), Polirom, 2006.
- [5] S. Buraga, *Semantic Web* (in Romanian), Matrix Rom, 2004.
- [6] D. Brickley, R. Guha (eds.), *Resource Description Framework (RDF) Schema Specification*, W3C Candidate Recommendation, Boston, 2000: <http://www.w3.org/TR/rdf-schema>
- [7] M. Daconta, L. Obrst, K. Smith, *The Semantic Web*, Wiley Publishing, 2003.
- [8] J. Davies, D. Fensel, F. van Harmelen (eds.), *Towards the Semantic Web*, John Wiley & Sons, 2003.
- [9] Y. Gil et al. (eds.), *The Semantic Web – ISWC 2005. Proceedings of the 4th International Conference*, LNCS 3729, Springer-Verlag, 2005.
- [10] L. Laera, V. Tamma, „The Hitchhiker’s Guide to Ontology Editors”, *AgentLink News*, Issue 18, August 2005.
- [11] F. Manola, E. Miller (eds.), *RDF (Resource Description Framework) Primer*, W3C Recommendation, Boston, 2004: <http://www.w3.org/TR/rdf-primer/>
- [12] M. Smith et al. (eds.), *OWL Web Ontology Language Guide*, W3C Recommendation, Boston, 2004: <http://www.w3.org/TR/owl-guide/>
- [13] A. Pease, „Formal Representation of Concepts: The Suggested Upper Merged Ontology and its Use in Linguistics”. In A. Schalley, D. Zaefferer (eds.), *OntoLinguistics*, 2005.
- [14] * * *, *Jena*: <http://jena.sourceforge.net/>
- [15] * * *, *OilEd*: <http://oiled.man.ac.uk/>
- [16] * * *, *OntoStudio*: <http://www.ontoprise.de/>
- [17] * * *, *pOWL*: <http://powl.sf.net/>
- [18] * * *, *Protégé*: <http://protege.stanford.edu/>
- [19] * * *, *Redland RDF Application Framework*: <http://librdf.org/>
- [20] * * *, *SWOOP (Semantic Web Ontology Overview and Perusal)*: <http://www.mindswap.org/2004/SWOOP/>
- [21] * * *, *World-Wide Web Consortium’s Technical Reports*, Boston, 2006: <http://www.w3.org/TR/>