Is Participation in the Semantic Web Too Difficult?

Stefan Haustein and Jörg Pleumann

University of Dortmund, Computer Science VIII / X,
Baroper Str. 301, D-44221 Dortmund, Germany,
{stefan.haustein, joerg.pleumann}@udo.edu

Abstract. As long as there is not a sufficient base of RDF-annotated pages, the benefits of participating in the Semantic Web are barely visible. This is true in particular for content providers like individuals or small institutions. These potential participants can't afford the additional work necessary for the Semantic Web, yet they're needed for the Semantic Web to reach the critical mass that will make it a success. This paper discusses problems that may prevent small content providers from participating in the Semantic Web, as well as a possible way to lower the barrier for entry using tools like our own Information Layer system.

1 Introduction

Roughly a decade ago, a handful of more or less technically interested people started participating in the collaborative effort of creating something later to be called the World Wide Web. Among others, there were two significant reasons for the success of the project: Participation was simple, and the results of the work were immediately visible to the creator.

As an example, in order to build a basic web presence for a university department, it was sufficient to place a few HTML files in a directory structure and then start an HTTP daemon delivering the content on request of a client. Since HTML was easily understood, pages could be generated without the assistance of specialised tools – at least by people who were familiar with SGML, TeX or other structured text formats. Since an HTML user agent could also be used to display files residing in the intranet or even on the local harddisk, WWW technologies were – as a side effect – also used as a means of discussion or personal documentation, which resulted in quick and wide-spread adoption of the whole idea.

Now, ten years from then and with the World Wide Web truly deserving its name, we are at a point that is similar to some extent: The emergence of the Semantic Web. Theoretically, this more formal and machine-readable add-on to the existing web could undergo an evolution quite similar to its predecessor. One would just need to annotate existing HTML pages with the desired RDF code, RDF being, again, a language that is easily understood and quickly written down. However, there are some problems in this approa ch that might turn out to be an obstacle for the success of the Semantic Web.

The biggest problem is redundancy: Adding RDF annotations to HTML pages generates redundant information, since parts of the content have to be duplicated in a machine-readable manner. The usual problem of maintaining consistency between the two versions arises, and it gets even worse once the RDF information is moved into a separate file. As a result, there is a significant amount of additional work necessary for participating in the Semantic Web, but there is no immediate benefit for the participant: RDF i nformation is primarily meant to be consumed by computer programs – other people's computer programs, to be accurate –, and it is usually not useful for the original provider of the content. Thus, the Semantic Web relies on the "network effect" even more than the original Web did: It becomes useful only when a large enough number of participants exists. Unfortunately, given this currently limited usability, it is hard especially for individuals and small institutions to take the initial migration step t o the Semantic Web: The barrier for entry is too high.

2 Tools to Lower the Barrier

One way to lower the barrier might be tool support. Programs like Protégé [1], OntoBroker [2] or Ontology Builder [3] ease ontology and RDF(S) data management. They could be seen as the Semantic Web equivalent to HTML editors, and they help to solve the language issue. The redundancy issue, however, persists as long as one still wants a plain HTML version of the pages, viewable with a regular browser. The latter is, of course, a requirement for a smooth transition from the traditional Web to the semantic one.

To get rid of the dilemma caused by the redundant information, it seems to be a promising solution to hold the "semantically-relevant" information in a fine-grained storage, say, a relational database, and generate HTML as well as RDF output on-the-fly by using templates for both targets. This approach is somewhat similar to the blend of database and XML-generating front-end that is quite common these days (e.g. Cold Fusion[4], PHP, Enhydra etc.). It would also allow to address additional targets, say, WML or different HTML versions, without additional effort.

However, if we start modelling the tables for, say, a university department's Web presence, another problem becomes obvious: Assume we need at least tables for persons, their research topics, projects, and publications. Since most of the associations between these tables are n:n and thus require separate association tables in a relational database, the example results in quite a lot of tables (10, to be accurate), each of which potentially contains only a very small subset of all the possible instances at run-time.

In this case, the benefit for the content provider, that is, the dynamic generation of RDF, HTML or WML from a single set of data, obviously does not outweigh the extra effort inherent in maintaining all these tables. The barrier for entry to the Semantic Web is still too high.

3 Design Goals for a Semantic Web Server

So what are the requirements for an easy-to-use Semantic Web "server" aimed at our target audience of individuals and small institutions? This section discusses a set of design goals for a tool that may allow small content providers to easily participate in the Semantic Web.

Build on existing knowledge: First it should solve the language issue. Users having a background in AI may be expected to be familiar description logics. For mainstream acceptance, though, building on a recognised standard for conceptual modelling might be the better alternative. We have chosen to adopt the ideas of [5] here, who propose UML as a language for ontology modelling. Most students of computer science or related engineering disciplines can be assumed to be familiar with UML and modelling tools like Together or Rational Rose.

Avoid redundancy: Second, the server should reduce the workload posed on the administrator. In particular, the redundancy between RDF code and HTML needs to be avoided. Instead, there should be a simple way to dynamically generate HTML and RDF pages from a common fact base. While it is relatively easy to provide a generic mapping from a fact base to HTML pages, this mapping needs to be sufficiently configurable to match the user's preferences or a given corporate design.

Provide additional benefits: Third, the Semantic Web server should provide "added value" that helps in lowering the entry barrier for the planned target audience of individuals and small institutions. In conjunction with an integrated storage and user management, for example, it can provide significant advantages over usual content management systems, such as HyperWave[1], Zope[2] or OpenCMS[3] These systems, which are widely used for managing a set of HTML pages, usually have a fixed set of metadata for annotating the pages. Here, ontology-based Semantic Web solutions would provide much more flexibility.

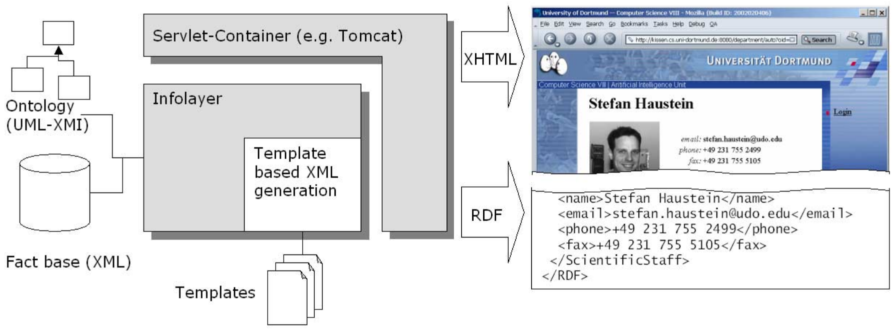
Immediate gain needs to outweigh extra cost: Finally, while the initial migration step will generate some extra effort, the system needs to ensure that this cost is outweighed by the gain for the content provider. This gain should not count too much on the network effect of the Semantic Web, because this effect might take some time to really pay off. Instead, the gain has to be immediately visible to the content provider.

4 The Information Layer System

In order to prove that the ideas of storing content in a fine-grained database and generating output on-the-fly actually works and provides significant advantages even without counting the "network effect" of the Semantic Web, we have chosen to modify our own "Information Layer" [6] system with respect to the design goals listed above. The reasons for building on our own system instead of modifying, for example, Protégé were mostly of pragmatic nature: It provided a solid basis, an d we know it well enough to make adoptions in a predictable time frame.

The Information Layer system was originally conceived as an integrated information platform for software agents and human users, and its design took into account the data redundancy issue that turns out to be an obstacle for the Semantic Web now. The system stores data in a simple object-oriented XML format the structure of which is determined by an ontology, and it features an XML-based template mechanism that is suitable for generating HTML output as well as more "formal" output consumable by software a gents. Obviously, when information is already machine-readable for agents, it is not a

[1] http://www.hyperwave.com
[2] http://www.zope.org
[3] http://www.opencms.org


Ontology (UML-XMI)
Fact base (XML)
Servlet-Container (e.g. Tomcat)
Infolayer
Template based XML generation
Templates
XHTML
RDF
University of Dortmund — Computer Science VIII - Mozilla (Build ID: 2002020406)
File Edit View Search Go Bookmarks Tasks Help Debug QA
http://kissen.cs.uni-dortmund.de:8080/department/auto?oid=
Search
UNIVERSITÄT DORTMUND
Computer Science VIII | Artificial Intelligence Unit
Login

Stefan Haustein

email: stefan.haustein@udo.edu
phone: +49 231 755 2499
fax: +49 231 755 5105

```
  <name>Stefan Haustein</name>
  <email>stefan.haustein@udo.edu</email>
  <phone>+49 231 755 2499</phone>
  <fax>+49 231 755 5105</fax>
 </ScientificStaff>
</RDF>
```

Fig. 1. Architecture of the Information Layer system

big leap to bring this information to the Semantic Web – we simply made use of the template mechanism to add the ability to output RDF data as well.

The architecture of the system, which is depicted in figure 1, consists of these main components or ideas:

- The Information Layer runs as a Java servlet. It makes use of the ontology, the fact base and the templates to generate its output.
- The ontology is described in UML, following the the argument raised in section 3. It can be read into the system from an XMI file, which allows to utilize an existing CASE tool for ontology modelling. No extra RDFS editor is necessary.
- A subset of the Object Constraint Language (OCL) [7] is used as a query language in the system. While other languages would have been possible (and actually have been used in the system before), the OCL suits the UML-based approach to ontology modelling very well.
- The system generates a user interface of HTML pages for browsing the fact base and viewing individual entries on-the-fly. It also generates form-based HTML pages to modify the fact base, so that, again, no additional tool is needed to modify the facts stored in the system. The HTML forms take into account all the rules imposed by the ontology, so that, for example, an 1:n relation will always have the proper cardinalities at both ends.
- There is an option to upload arbitrary files (PDF, MPG, ...). While this feature might look a bit odd at the first sight, it is a typical feature of content management systems and we incorporated it as an example of the added value mentioned in section 3. Of course, the content of the uploaded files is opaque to the system, which is kind of controversial to the idea of providing fine grained information in RDF-format. Yet, the system supports the addition of detailed meta-information.

The installation of an Information Layer based system is rather straightforward and includes the following steps:

1. Build a simple base ontology with the UML tool of your choice, or just use the sample ontology available from the InfoLayer homepage as a starting point.

2. Install Apache/Tomcat or any other Web server that is capable of running Java servlets.
3. Copy the Information Layer servlet files into the Web services directory of the server and adjust the configuration in the `servlet.xml` file to your local environment.

After these steps, the system is already in a state that allows you to add content using the generic Web interface. This content is immediately available to both the HTML and, with proper templates installed, the RDF world. The right side of figure 1 shows how staff data belonging to the university department scenario mentioned before might look like.

To improve the InfoLayer installation further, the following two steps might be performed in an arbitrary order or even in an iterated manner until a point is reached where the system perfectly suits the needs of the user:

– The ontology can be extended, either by adding new classes or by insertings new attributes or relationships into existing classes.
– The default look and feel of the system can be customized to match, for example, a corporate Web design by using HTML templates.

Please note that the temporal frame of the latter two steps is not fixed. One can, in the university department scenario, start with managing publications only, and add other concepts like projects, topics, persons or courses later. This, together with the fact that the InfoLayer servlet integrates well with an already existing Web presence, ensures a smooth migration from the traditional Web to the semantic one.

5 Experiences with the System

The InfoLayer system has been in development for several years now. The most up-to-date version is currently being used as a prototypical Web presence for MuSofT, a Germany-wide project that develops multimedia teaching material for software engineering education. The goal of this web presence is to manage and distribute the learning objects contributed by the various project partners. To allow efficient retrieval of material, LOM[4]-conforming metadata is provided using the system's ontology capabilities. Since the whole HTML user interface is dynamically generated from the ontology and the fact base, the initial extra work to get the system running has already paid off: Changes to the fact base, which occur whenever a learning object is added, changed or removed, are immediately visible to both the human and the machine-readable worlds. Even changes to the metadata, that is, to the ontology itself, can be made easily without having to modify the rest of the system afterwards. This application also makes use of the content management ability provided by the file upload feature.

Another project that utilizes the Information Layer in its current form is a database for Java-enabled small devices like cell phones and personal digital assistants[5]. Here, the ontology descibes the devices, their capabilities, vendors, available protocols and

[4] http://ltsc.ieee.org/wg12/index.html
[5] http://www.kobjects.org/devicedb

known bugs. Again, changes to the fact base are quite frequent, but do not require the duplicated effort of updating a human and a machine-readable version, which makes the website very easy to maintain.

Previous versions of the Information Layer system are still in use for MLnet teaching information server[6] and in other internal projects. For more details about the information layer software and it current applications, please refer to the homepage at `http://infolayer.org`.

6 Conclusion and Outlook

The Semantic Web is a great vision. However, for a broad adoption, simple tools that allow participation without a background in AI are still rare. Protégé and similar tools seem to aim in this direction. We would like to contribute our own Information Layer system. While other tools focus on the ontology building process, we mainly tried to address simplicity. This way we hope to improve availability of structured information suitable for the Semantic Web. We did not put a focus on advanced features like full DAML+OIL support, nor do not have a priority here in the future.

References

1. Stanford University: Using Protégé-2000 to Edit RDF. (2000) http://protege.stanford.edu/protege-rdf/protege-rdf.html.
2. Decker, S., Erdmann, M., Fensel, D., Studer, R.: Ontobroker: Ontology based access to distributed and semi-structured information. In Meersman, R., other, eds.: Semantic Issues in Multimedia Systems, Kluwer Academic Publisher, Boston, 1999. Kluwer Academic Publisher, Boston (1999)
3. Das, A., Wu, W., McGuinness, D.L.: Industrial strength ontology management. In: International Semantic Web Working Symposium (SWWS), California, USA, Stanford University (2001) 17–37
4. Brooks-Bilson, R.: Programming ColdFusion. O'Reilly (2001)
5. Cranefield, S., Purvis, M.: Uml as an ontology modelling language. In: Proceedings of the Workshop on Intelligent Information Integration, 16th International Joint Conference on Artificial Intelligence (IJCAI-99. (1999)
6. Haustein, S.: Utilising an ontology based repository to connect web miners and application agents. In: Proceedings of the ECML/PKDD Workshop on Semantic Web Mining. (2001) to appear, accepted for publication.
7. Object Management Group: OMG Unified Modeling Language Specification, version 1.3. http://www.omg.org/technology/documents/formal/unified_modeling_language.htm (2000)

6 http://kiew.cs.uni-dortmund.de:8001/