Faculty of
Computer Science

universität
wien

Department of Distributed and
Multimedia Systems

Technical Report TR-20080704 July 2008

Small-Scale Evaluation of Semantic Web-based Applications

Diman Todorov, Bernhard Schandl,

Small-Scale Evaluation of Semantic Web-Based Applications

Diman Todorov, Bernhard Schandl
University of Vienna
Department of Distributed and Multimedia Systems
{diman.todorov|bernhard.schandl}@univie.ac.at

Abstract

The evaluation of user interfaces for Semantic Web-based applications is hard: Often, such applications are designed on top of the very generic triple model and can make only limited assumptions on the structure of the data it operates on. Many concepts, like ontologies and resources, are unfamiliar to users, and the resources for such evaluations are limited. However to achieve a certain level of interface quality, user assessments are a necessary step in UI design. In this paper we describe the experiences we have made with a small-scale of such an application, which revealed issues that can be applied to a wide class of applications.

1 Introduction

More and more applications use Semantic Web technology (mainly RDF(S), ontologies, and reasoning) to express the information they process. Such applications can be found on the web, where they can make use of the increasing amounts of data available and integrate them to infer new knowledge. Furthermore, these technologies provide a sound basis for desktop applications, i.e. applications that are executed and used on a user's personal desktop machine. In such environments, RDF and consortia are well suited to describe and annotate for instance personal user data, as the upcoming research area of the Semantic Desktop indicates [16].

Most semantic systems have in common that the vocabularies used to describe data are extensible, and often they are not specified in a formal way. While this fact is one of the greatest strengths of the RDF family, it also imposes severe consequences for application design and development: tools must be designed so as to accommodate changing and developing data models and data formats.

This issue becomes especially apparent in the context of user interfaces. Most "traditional" applications use a closed-world data model: the processed data is known at design time, and appropriate user interfaces can be designed, evaluated, and optimized based on this model—a rigid data model is a solid basis for user interface design. On the contrary, an application operating with an open data model can only guess what the data it operates on will look like. Often the only known factor is the underlying meta model of RDF; URIs, resources, triples, and so forth. Thus the user interface for such an application must be designed sufficiently generic, yet suitable and understandable for the user.

Naturally such user interfaces are hard to evaluate for multiple reasons. Semantic systems are not yet widespread, hence the average user is not familiar with their inherent concepts. It is the task of the developer to deliver a usable system, thus research in improving the surface of semantic systems is worth the effort. However, UI research and development without evaluation is only one half of the journey. Often user interfaces are designed by the developers of the system and undergo only a very restricted evaluation, if any. The problems and obstacles that hinder a proper UI evaluation process are well known, thus we seek for small-scale evaluation methods that do not claim to cover all aspects, but still yield valuable results and help to improve the system's interface.

We have developed the *Semplorer*, a generic interface for browsing, searching, and manipulating repositories of unstructured digital objects that can be accessed via SemDAV [18]. SemDAV can be regarded as an extension to traditional hierarchical file systems, where files are enriched with metadata descriptions based on RDF. In the following we give a short system overview and describe important aspects of this user interface. Then the approach we applied for evaluating the interface is outlined, followed by a discussion of the concrete findings that we gained. We conclude this paper with a brief overview of related work.

2 Application Description

2.1 SemDAV: Semantic Data Asset Management

SemDAV can be seen as a semantic extension to file systems. It disbands their strictly hierarchical structure and combines the basic characteristics of files with a variety of description metaphors based on RDF. The basic idea of SemDAV is to provide a rich storage infrastructure for desktop applications that takes over the functions of the file system, but additionally enables rich data annotation and application interoperability through its extended information model. SemDAV defines the *sile* as a basic information unit. As the detailed description of the sile data model is out of the scope of this paper, we give a short introduction.

Instead of a path and a file name, siles are uniquely identified by a URI. Similar to a file, a sile is an object with content of arbitrary kind and size. However, instead of placing it into a hierarchy of directories, it can be described (*annotated*) by various means: user-defined *tags* (keywords) can be attached to siles. They can be described using *categories* that are similar to ontology classes, and *attributes*, which consist of a name and a typed value. Finally, siles can be connected by typed *relationships*.

All annotations are represented using RDF triples, an example is given in Listing 1. The depicted sile's content is described in lines 3 to 5, and it is annotated by a category (line 6), attributes (lines 7–9) and relationships (lines 10 and 11).

Listing 1: Java source code represented as sile

```
 1 <urn:uuid:2a7d71a4-...-9d47a8cd1c6c>
 2   rdf:type sw:Sile ;
 3   sw:content-size "2942"^^xsd:nonNegativeInteger ;
 4   sw:content-type "text/java"^^xsd:string ;
 5   sw:content "cGFja2F...fQ0K"^^xsd:base64Binary ;
 6   sw:cattype java:Class ;
 7   java:name "WebDAVServlet"^^xsd:string ;
 8   svn:revision "772"^^xsd:positiveInteger;
 9   java:comment "to␣be␣optimized"^^xsd:string ;
10   java:defines <urn:uuid:3a1970f8-...-c9ac23936ef5> ;
11   java:package <urn:uuid:006ea420-...-ed692052e2f0> .
```

The representation of siles and their annotations using RDF has certain implications. On the one hand, sile metadata can be handled by every RDF processor without further adoptions. Query languages like SPARQL can be used to query sile data, and generic RDF browsers represent siles in a decent manner.

The opposite—converting arbitrary RDF data to the sile model—is also possible. Although not every RDF graph can be mapped to the sile model, most of the annotations described above are based on standard RDF functionality. For example, datatype properties are mapped to sile attributes, and object properties can be mapped to sile relationships. Thus, existing RDF data can be translated to siles with practicable effort.

One benefit of this representation format is that this makes it possible to integrate data from external sources into SemDAV. For instance, messages stored on an IMAP server can be translated to RDF [5] and can then be displayed as siles. These can then be further annotated or brought into relationship with siles from other sources. Consequently, SemDAV can be regarded as a semantic data integration solution.

We have implemented a SemDAV prototype as a client-server application in Java, cf. Figure 1. The *repository* acts as the server component and hosts siles with their content and annotations. Additionally the server can integrate data from external sources, as described above. Finally, it performs reasoning and logging on the data, which is used to gain further information about the stored siles [19]. The server exposes its functionality via two interfaces. First, it provides access to the raw data via a combination of a REST interface and the SPARQL protocol, allowing access to the sile content and to the RDF store, respectively. Second, it exposes high-level operations (like *CreateSile*, *AddTag*, or *DeleteAnnotation*) via an XML-RPC based interface. The server can use different RDF stores to host the data; currently we have developed adapters for Jena and Sesame.

A comprehensive API for the development of SemDAV client aplications is provided. This interface can be used by applications to store data. It is mapped

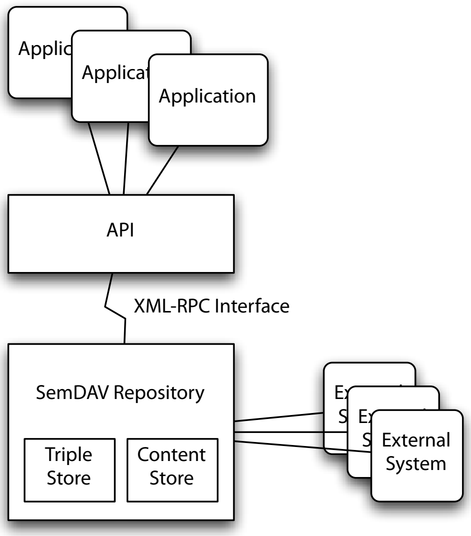
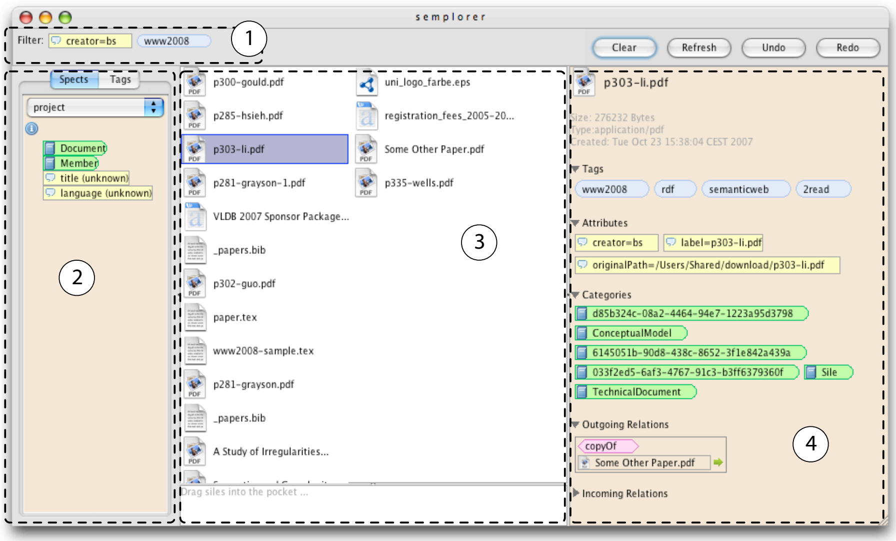


Figure 1: SemDAV Architecture



Filter: creator=bs www2008

project
Document
Member
title (unknown)
language (unknown)

p300-gould.pdf
p285-hsieh.pdf
p303-li.pdf
p281-grayson-1.pdf
VLDB 2007 Sponsor Package...
_papers.bib
p302-guo.pdf
paper.tex
www2008-sample.tex
p281-grayson.pdf
_papers.bib
A Study of Irregularities...
uni_logo_farbe.eps
registration_fees_2005-20...
Some Other Paper.pdf
p335-wells.pdf

Drag siles into the pocket ...

p303-li.pdf
Size: 276232 Bytes
Type:application/pdf
Created: Tue Oct 23 15:38:04 CEST 2007

Tags
www2008 rdf semanticweb 2read

Attributes
creator=bs
label=p303-li.pdf
originalPath=/Users/Shared/download/p303-li.pdf

Categories
d85b324c-08a2-4464-94e7-1223a95d3798
ConceptualModel
6145051b-90d8-438c-8652-3f1e842a439a
033f2ed5-6af3-4767-91c3-b3ff6379360f
Sile
TechnicalDocument

Outgoing Relations
copyOf
Some Other Paper.pdf

Incoming Relations

Figure 2: The Semplorer interface: active filters (1), spects/tags (2), siles (3), sile details (4)

either to a repository running on the local machine, or to a remote instance, in which case the API wraps the necessary XML-RPC calls. As a first demonstration application we have developed a user interface for the management of siles called *Semplorer* which resembles similarity to file browsers like Windows Explorer or Apple Finder. This user interface was the subject of the small-scale evaluation and is described in more detail in the following.

2.2 Semplorer: The SemDAV User Interface

The Semplorer is a user interface that allows to browse, search, and manipulate siles and their annotations. It is the counterpart of a file browser: it does not provide functionality for specific content types (this is delegated to corresponding applications, like text processors), but provides a generic interface to access the annotations as described in the previous section.

The GUI (Figure 2) is split into four major parts. In the center of the screen, siles are displayed. Their representation is oriented towards the usual file representation, using an icon and a text label. The icon is selected based on the sile's content type; for later versions we plan to use a designated relationship type for the icon; thus any picture (e.g. a thumbnail preview) can be used as visual representation. The label is derived from the `rdfs:label` attribute of the sile.

When a sile is selected by the user, its annotations are displayed in the right section of the window. Below the sile's label and information regarding the content, all annotations are displayed using widgets with different shape and color. We use the different widget styles to indicate different types of annotations (see Figure 3): tags are displayed using blue widgets with round borders. Attributes are represented by yellow boxes, and categories are green boxes where the right edge is rounded. Finally, relationships are displayed by magenta widgets with arrow-like left and right borders.

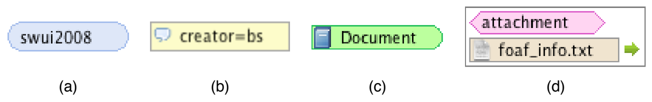


Figure 3: Representation of sile annotations: (a) Tag, (b) Attribute, (c) Category, (d) Relationship

On the left hand side of the window, various organizational metaphors are displayed. In the screenshot, the elements of a small project management ontology are displayed; these can be used to annotate and relate siles. Using the buttons in the top section of this area, users can switch between ontologies, or to an alternate view that displays all tags present in the repository.

Searching for siles is done by applying *filters*. Filters are entities that restrict the set of displayed siles to those that match certain criteria. To apply a filter, any annotation can be dragged into the top area of the window (in the screenshot, the attribute/value pair *creator=bs* and the tag *www2008* are used for filtering). Every widget present on screen can be used for filtering, regardless of its origin. For instance, tags can be dragged from the left pane, where all available tags are displayed, or from the right pane, where the tags of a specific sile are displayed. Also, siles from the center pane can be used as filters: in this case, the filter is restricted to siles that have a relationship with the dragged sile. Elements can be removed from the filter by moving the mouse over them until a minus sign appears—clicking on this sign removes the filter element.

All active filter elements are AND-combined; for the sake of simplicity more complex queries that include OR and NOT operators have not been considered. However we plan to implement an advanced interface for such queries in the future.

After every modification of the filter, the set of siles displayed in the center area is updated accordingly. There is a small area on the bottom of this area; any widget (siles or annotations) can be placed here. This area is not affected by changes to the filter or the selected sile; it allows the placement of frequently used items for later reference.

Annotation operations can be performed as simple as applying a filter. Any annotation widget present on screen can be dragged on any available sile; the corresponding annotation is then written to the repository. Siles can be related by dragging one sile onto another one, after which a context menu appears. From this menu the user can select the type of the relationship based on the object properties defined in the loaded ontologies. Annotations can be removed by clicking on the minus sign that appears when the mouse is moved over a widget. By clicking into the text area of an annotation widget, it can be edited: in this manner, the value of attributes or the label of tags can be modified.

As far as it is possible in the Java language the Semplorer is also integrated into the user desktop. Files can be dragged into the Semplorer's central area, the files are then converted to siles and stored into the SemDAV repository. Also, web links can be dragged into this area and are then stored as references to these web resources. By doing so, one can define relationships between files on the desktop and arbitrary web resources. For certain types of files and web resources, the Semplorer additionally tries to extract further information: for instance, when a link to an HTML page is dropped, the application tries to extract RDFa data, if present, and stores the embedded triples as sile annotations.

3 Evaluation

In the SemDAV project the resources for usability evaluation are limited. Aquiring an external usability expert was not an option, so cheap evaluation methods that could be performed by inexperienced evaluators were needed. There are not many methods that fulfill these constraints; we considered *heuristic evaluation* [11–13], *think aloud evaluation* [13] and *paper prototyping* [20]. Nielsen [13] advises to apply at least two methods for acceptable accuracy. We have chosen to combine a heuristic evaluation and a think aloud evaluation because of their simplified administration in comparison to that of a paper prototyping evaluation. In the following we outline the three methods as well as the rationale behind our selection.

3.1 Evaluation Methodologies

Historically, *heuristic evaluation* has evolved from guideline reviews [10]. Reviewing a user interface with respect to a potentially very large set of guidelines is one of the earliest methods of usability assessment. The main problem of a guideline review is that the number of guidelines easily goes into the thousands. Reviewing a user interface for its conformance to a large set of very specific guidelines is not only a tedious task but, in consequence, also a very expensive process. Molich and Nielsen [10] have distilled 10 heuristics by consolidating guidelines from various sources.

Originally, the *think aloud method* was a method for psychological research. In such an evaluation users are asked to continuously verbalize their thoughts while they are using the application being evaluated. The result of a think aloud evaluation is information on what users are doing and why they are doing it. The particular value of this information stems from the fact that it is collected while the user performs the tasks she is verbalizing. Because of its qualitative nature and its relatively informal structure the think aloud method has several pitfalls, one of which is that an evaluator may find it hard to distinguish between information on what the user is doing and rationalization theories of the user. An example described in [13] is a user spending more time than expected looking for an input field in a form. When she finally finds the field she might suggest to place the field elsewhere. While users are good at demonstrating usability issues, they are not experts in solving them. For this reason their suggestion for the new location of the field should be taken with a grain of salt.

Another caveat is that an evaluator may have to frequently prompt a user to continue thinking aloud. Thinking aloud is unnatural and may seem strange to users, consequently it may happen that the verbalization stream stops. In such a situation the evaluator has to prompt the user to continue talking. However it requires a certain level of experience to prompt for further information without influencing the evaluation result. If, for example, a user is hesitating while looking at some message, the evaluator might ask her to explain why. If the user has not yet noticed the message, the prompt would influence her attention focus. Other misconceptions and difficulties with this method have been described by Boren and Ramey [3].

A more traditional evaluation method for investigation of user interface design questions is *paper prototyping*. This method is well described in literature (most notably in [20]), so we will only briefly outline it and compare it to the two other techniques we have discussed. In the most general form of paper prototyping, all possible user interface states are sketched on sheets of paper. The evaluator asks a test subject to manipulate the paper interface and imitates a computer by presenting sequences of user interface states to the test subject. The whole process is logged by a second evaluator and may also be videotaped.

One of the advantages of paper prototyping is that it can be applied without extensive theoretical background; of course the quality increases with the evaluator's experience since more elaborate and more detailed scenarios and questions can be used. Another strength of this method is that it does not require a running prototype, hence it can support the design process before the user interface is actually implemented.

3.2 Heuristic Evaluation of the Semplorer

Because the team involved with the development of the SemDAV application had no prior experience with a heuristical evaluation, the obvious first step was to introduce the method. The evaluators were acquainted to the guidelines and their interpretation in a tutorial session, and the interpretations encompassed by every guideline were elaborated. The understanding of each guideline was ensured by the demonstration of a violation in commonly available software. Finally, a heuristic evaluation exercise [10] was discussed. The questions that were asked during this discussion anticipated the biggest difficulties the evaluators had during the evaluation. One question was how much time to spend on the evaluation, another question was how many problems the evaluators were expected to find. The third big problem was the format of the evaluation protocol. Care was taken to leave the answers to these questions as open as possible in order to ensure that the evaluators were not influenced by constraints.

In retrospect this may not have been an optimal decision. We recommend to not only present and describe an example of an evaluation, but to actually perform a simple evaluation in the course of the introductory tutorial. While this makes the preparation and administration of the tutorial more time consuming, the extra effort has manifold returns in terms of a common understanding of the method that is shared by all participators.

The most critical part of a heuristic evaluation is the neccessary familiarization of the assessment team with the evaluation method. The guidelines by themselves are worded in a very general manner which is prone to misunderstanding. For example, consider Nielsen's first guidline:

> Simple and natural dialogs: *Dialogs should not contain information that is irrelevant or rarely needed. Every extra unit of information in a dialog competes with the relevant units of information and diminishes their relative visibility.* [13]

We discovered that several of our evaluators misunderstood what was meant by the word "dialog". Nielsen's intended meaning of this term is the way the user interface communicates with the user to help her achieve a goal. Some evaluators however assumed that this guideline only applies to dialog boxes. A good way to aleviate problems like this is a short discussion of possible interpretations in the tutorial session.

Even with a good instruction, detecting conflicts with the heuristics is a difficult task. Studies performed by Nielsen and Molich [11] have shown that, depending on their experience and knowledge of user interfaces, single evaluators discover about 20% of all usability problems. Thus the method only yields acceptable results if several persons independently perform the evaluation. The overlap of discovered problems between individual evaluations is very low: the cumulative results of 3 to 5 evaluators account for about 50 to 80 percent of all evaluation problems [11]. Still, the quality of the results highly depends on previous experience with usability inspection methods. It is unclear however whether this estimation also holds for more complex applications that require a longer time to familiarize.

In our evaluation, we found 50 issues with 5 overlaps. This amounts to 45 issues among 4 evaluators. Comparing it to heuristic evaluations of other applications, there is a reason to believe that the real number of issues is larger. A rough interpolation on benchmark results of the method [11] make it safe to assume that we have found between 50 and 75 percent of all usability problems. Nevertheless, the evaluation yielded considerable results considering the low volume of invested effort.

4 Usability Challenges

In the following we outline difficulties we have met while designing the Semplorer user interface, most of which were also reflected by the evaluation results. The scope of the problems varies—some apply specifically to the sile model and its user interface, while others have a more general validity. All of the identified difficulties can be derived from the attempt to provide a method for communication between users and semantic applications: as with any novel technology, the interface has to be based to the greatest possible extent on concepts with which users are already familiar, while at the same time it must introduce new data manipulation possibilities in order to harness the new technology to its full extent.

4.1 Navigating Large Data Sets

Siles are a semantically enriched version of the file concept, but the sile model does not include a counterpart of hierarchical directories. If we imagine a file system and subtract from it its directory structure, we end up with a number of information entities that easily goes into the tens of thousands [1]. Hierarchical directories are an efficient method for bringing a certain amount of order into this information cloud, they are however often too restrictive to depict all facets of relations between data. When the large numbers of objects in a semantic repository are annotated with an even larger number of attributes, organizational means have to be provided that are comparable to traditional approaches in terms of responsiveness, but excel them in expressive power. This challenge applies to any attempt to organize information, although its severity depends on the domain of the application.

Devising a high-performance information storage and retrieval model is much easier for applications with a narrow domain. Such applications benefit from the fact that their users already own a mental model of the information they are working with, which designers can attempt to translate into the application's data model. The mental model for browsing arbitrary information in a semantic paradigm is not understood, and probably not even shaped yet. This thesis is supported by research such as the evaluation of the merits of numerical volume indicators in the mSpace user interface [24].

4.2 Designing for Diversity

The SemDAV user interface tries to be as generic as possible. It is designed to be a suitable tool for tasks that cope with information organization. Sauermann et al. claim that users are unwilling to adopt a generic interface [17], but the success of projects such as Haystack [15] make this claim disputable. When only one generic application is developed, it is possible to experiment with many different extentions at a relatively low implementation overhead. Enabling the development of low cost functional prototypes is important and beneficial at the current, mostly experimental stage of semantic applications research.

The genericity of an application is not limited to technological aspects: it should also encompass the intended user group. An information retrieval application should make as few assumptions about its users as possible. It should address not only e.g. scientists from any domain, but also users with varying computer expertise, handicapped users, children and older people. Even users from the same category may exhibit different search behaviors that must be accommodated. An existing type information system which fulfills these requirements (next to playing a role in forming a cultural identity and being a social center) are libraries. Stephanidis et al. [23] further elaborate the analogy between libraries and information systems. They argue that the foundational paradigm of user interface design, *knowing the users*, is not directly applicable in applications with a large audience. He also provides insights into the challenges faced in information system user interface design as well as methods to overcome them [22].

4.3 Visualization and Navigation of Ontologies

The majority of tools (Protege, Swoop, TopBraid, etc.) visualize ontologies using a tree view; however, in general ontologies are not trees. A graph G is a tree if it is connected and it is not connected if any edge is removed from G. If a class U is a subclass of both classes, X and Y, which in turn are a subclass of `owl:Thing` then the resulting class hierarchy graph is not a tree. The graph will remain connected regardless which edge is removed. Tools utilizing the tree view circumvent this problem by rendering two nodes for the class U, one which appears as a subclass of X and one which appears as a subclass of X. This node ambiguity may not be an obstacle for a user who is familiar with the structure of an ontology, but it may be confusing for less experienced users [4].

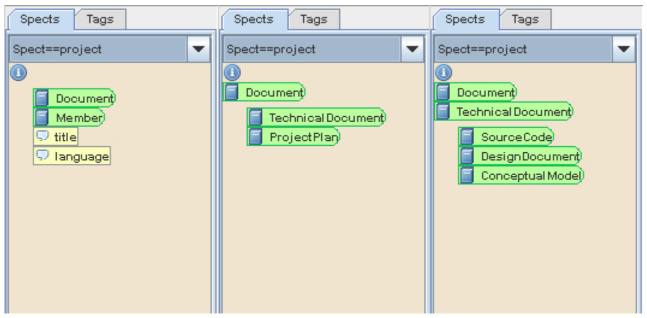


Figure 4: Browsing spects.

In the Semplorer we experimented with a different approach for ontology navigation, the *spect viewer*. The spect viewer displays all children nodes of the current node slightly indented. Beyond the current node a trail of already visited nodes is displayed. Every time the user clicks on a child node, the viewer navigates to that node and expands its children. If the user clicks on a node in the trail, the viewer steps back to the selected node. Figure 4 shows three spect viewers showing different hierarchy levels of the same ontology. The viewer on the left shows the spect as it initially appears to the user. The second pane shows the state after the user has clicked on the *Document* category—the category is now in the trail, and the viewer displays the sub-categories of *Document*. The third viewer shows the next level; here a trail of two categories has been walked and the sub-categories of *Technical Document* are displayed. Clicking on the "i" symbol resets the viewer back to the root node. If the interface is in the state depicted on the right and a user clicks on the *Document* node, the interface traverses up the hierarchy to the node representing the *Document* class, as shown in the middle screenshot.

The results of the usability evaluation however have been divergent with regard to this aproach. The two problems pointed out by evaluators were firstly that the parent/child relationship is not made sufficiently explicit, as e.g. the relation between the categories *Document* and *Technical Document* is not clear from the visualization. Secondly there exists no "bird eye view" of the ontology: searching for a particular category is difficult because the user has to remember the trail she had to walk to reach it.

4.4 Visualizing Queries

Instead of navigating through directory hierarchies, siles are managed using filters, which are dynamically evaluated and used to narrow down the user's view to their current context. A filter abstracts over the complexity of the query language. The most natural abstraction over the query language are boolean expressions, which are easier to formulate than a complex query. However it has been shown that users who are unfamiliar with boolean algebra find it difficult to

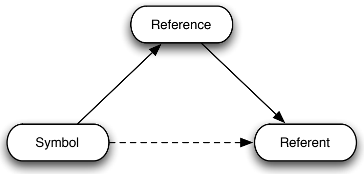


Figure 5: Semiotic triangle [14]

formulate boolean queries [2]. The filter component of the Semplorer is limited to connecting concepts, such as attributes or tags, with an *AND* relation. We have decided in favor of this restriction because AND is the easiest operator to use [25]. A similar method of information retrieval is mSpace [24], where also only the AND operator is used, despite the user interface is designed differently to the Semplorer.

While there has been some effort to visualize boolean queries in a user friendly manner [8, 25] there is no convincing solution known to the authors. We plan to survey and evaluate alternative abstractions over SPARQL which have more intuitive user interface translations, and we will also introduce a mechanism to persist queries between sessions, which reduces the need to remember and to manually reconstruct frequently used filter constellations. A set of reasonable default queries may alleviate initial problems with the formulation of boolean queries.

4.5 Semantics of Resource Names

The relation of a name to the thing it names is both, problematic and well researched [9]. There are several theories that attempt to model the relationships between symbols, concepts and concrete things, all of which agree upon the idea that a symbol and its referent are two different things. Ogden and Richards have introduced the semiotic triangle as a model of meaning (Figure 5). This metaphor expresses the idea that a symbol refers to a referent by the means of a reference, or, more concretely, an expression in a natural language uses a thought or an idea to refer to an object: When someone uses the word "horse" she is communicating a thought which entails a possibly real horse. The exact way of how the angles of the triangle refer to each other has been subject of a very large scoped linguistic and philosophical research.

The semantics of the sile model, and the meta models of the Semantic Web as a whole, fit well into this construct. The symbol is represented by the name of a sile, the referent is the URI which references to the content of a sile (Figure 6). The only flaw in this translation is that a sile does not necessarily have a content—it is possible that the meaning of the syle is entirely stored in the metadata attached to the sile. Thus an optimal reference model for siles has yet to be devised.

Even this incomplete model shows that the relation between siles (or re-

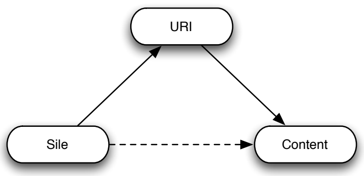


Figure 6: Semiotic triangle, applied to the sile model

sources) and their names (or labels) is of great interest to a user interface designer. This edge of the triangle is the bridge between the mental model of a user and the machine representation of the semantically enriched data, and a reference without a human readable name would be of little value.

If we consider siles as concepts in the linguistic sense, it is natural to assume that siles and their names should not have implications on each other. In a natural language the word "horse" can refer not only to a specific horse but to every animal of that species. Following the analogy of natural languages, several siles can thus have the same name, and one name can symbolize several siles.

One question which arises is whether two things denoted by the same name have anything in common. There are two competing reference theories which answer this question differently; the theory of realism and the theory of nominalism. The former assumes that two things bearing the same name share more than just their name. In its most simple, traditional form the theory of nominalism assumes that the only thing referrents have in common are their names. The semantics of siles adhere to the latter model: it is perfectly legal for siles to have common names but share no further properties or characteristics.

The ambiguity of meaning has been treated only very recently for Semantic Web issues by Garcia et al. [6]. Their research however only deals with synonyms in ontology mappings. The authors of this paper are not aware of any works treating problems of ambiguous labels for semantic resources.

4.6 Introduction of New Vocabulary

Semantic technology is described in domain specific terminology, and—unfortunately—the vocabulary used by semantic systems researchers is only understood by semantic researches. One of the reasons is that many of the terms have been adopted from other fields of research but their meaning was changed. The word "ontology", for example, originally meant "the study of being". In the field of Semantic Web research however an ontology is best described as a special form of a taxonomy, or as a formalized conceptualization.

It can be assumed that the majority of the users of semantic applications are not familiar with the language used by the scientists who created the application. An average user will be hopelessly overwhelmed by a message saying that the ontology currently in use does not allow the class *Person* to have a $1:n$ cardinality for the property *name*. The user's situation would be even worse if she had background knowledge in philosophy.

In cases where a system has a well-defined application domain an obvious way to circumvent this wording problem is to reformulate messages: For example, the application might translate the message to "A person may only have one name". Downey has recognized this problem in the evalution of the SEEK user interface [4]: during the evaluation one of the test subjects noted that the term "annotation" was too overloaded and thus not appropriate. If the application however targets towards more than one domain the approach of explicit wording becomes infeasible.

A pragmatic solution for applications with a universal purpose is to translate as much terminology as possible into a language understood by the majority of the intended audience. In cases where this is not possible, there are two options. The first option is to introduce new terms which are consistent with the rest of the application, in which case users will have to learn only the new terms: having to learn $n - k$ new words (where n is the total number of words and k is the number of known words) is still an improvement over having to learn n new words. The second option is to omit functionality which requires the understanding of foreign terms. As often, any solution for this problem will be a tradeoff between the expressive power of the application on the one hand, and the steepness of the learning curve for new users on the other hand.

The choice of terminology in the SemDAV user interface follows this paradigm. The application uses terms that are used in common language, such as "category" or "filter". In some cases it was safe to assume that users were familiar with very similar concepts to the ones we employed, hence we tried to choose words that are phonetically similar to the concepts already known to users but still accentuate the difference. The word "sile", for example, is very similar to "file", and "slink" is very similar to a "link" on the web, but nevertheless, slightly different. Unfortunately there are still more concepts which are unknown to the majority of the intended user population, like for example the word "ontology". In such cases we have deliberately used neologisms in order to avoid ambiguities.

5 Related Work

5.1 The EPOS Project

In the course of the EPOS project [17], a semantic user interface was evaluated. Over a longer period of time, qualitative and quantitative was collected: the former data was collected using a structured questionnaire, which users were asked to fill out on a daily basis; the latter was collected in the form of clickstream traces and quantized explicit user feedback. However, the claim that "participants agreed that the PIMO reflects their personal mental models" stands in certain contradiction to the theory that mental models are seldom conscious and users are not explicitly aware of them [21].

5.2 ESWC 2006 evaluation

The organizers of the 3rd European Semantic Web Conference[1] decided to use the state of the art in the field of semantic web applications for the local information infrastructure. The rationale behind was that by using their own applications the conference participants would gain more insight into practicability problems of semantic technologies. Although not being a small-scale evaluation, this scenario is particularly interesting because practically all of the users could be safely assumed to have a clear understanding of the concepts underlying the applications. Heath et al. [7] have assessed the practicability of the installed infrastructure by asking conference participants to answer open-ended questions on a local website. The authors remark however that their evaluation was more akin to a formative feedback than to a traditional UI evaluation, and they are aware that this approach is not a substitute for formal usability testing.

5.3 The SEEK project

Usability issues in the Scientific Environment for Ecological Knowledge (SEEK) project were assessed using a paper prototype [4]. This method was favored because it is known to yield good results at a low cost. Paper prototypes are easy to make, and evaluating them requires only a few test subjects from the intended user population. The SEEK project is very domain specific, which means that the user population was very small: the evaluation team had access to three potential users.

In contrast to the paper prototyping evaluation method known from literature [20], the author uses it as a mean of providing common ground between users and developers, which is necessary to discuss the shape of the user interface. While this approach may have its own merits, the standard method focuses on observing users rather than discussing the interface with users. The rationale behind favoring observing over involving is the assumption that users rarely have the expertise to make valuable constructive suggestions. Observing users of an application provides thorough insight into the usability weaknesses.

Some users may even be able to voice usability problems, whereas their suggestions for improvement are often inadequate. An example supporting this claim can be found in the result section of the SEEK evaluation: one user remarked that the term “annotation” was too ambiguous, and he suggest using “keywords” or “descriptors” instead. The authors point out that “keywords” is an unfortunate choice, because it is used elsewhere in a different meaning (see [4], page 9).

5.4 Numeric Volume Indicators in mSpace

mSpace is a domain specific information browser, intended to become a generic information navigation tool. Its interface is based on a multi-column view; every column represents one aspect of an information space. If the user browses

[1]`http://www.eswc2006.org`

information about movies, one column might show all directors and allow users to discriminate results but chosing a director. The order of columns can be rearranged.

The study performed with the mSpace interface [24] investigates the interpretation of numeric volume indicators (NVIs). Numeric volume indicators are numbers printed after every category indicating the volume of the category. As mentioned before, the mSpace user interface discriminates search results by applying a conjunction of category membership constraints. For example, when browsing music the user might first select an era, which would display all composers of this era. Subsequently the user can chose a composer to find out which pieces this composer has written. The composer constraint however may be omitted so that a selection of an era yields all pieces composed in this era. This flexibility makes it unclear what the most obvious interpretation of an NVI might be: is the volume of an era the number of composers or the number of pieces in this era?

The authors administrated an evaluation to answer two questions: first, what is intuitive interpretation of NVIs?, and second, how can this interpretation be influenced with various cues?. They designed the experiment to answer these two particular questions. The 20 participants of the study were matched in groups of 5 based on their computer experience. Each of the groups was presented with one of four interface conditions and was asked to interpret an NVI. After the users had given their answer they were engaged in a discussion introducing the other three interface conditions.

While the concrete outcome of the study is undecisive, the study presents a very valuable finding: the mental model for NVIs is not completely understood. The mental model for NVIs in the context of Semantic Web applications may not even be formed yet. The authors of the study recommend that user interface designers allow the users themselves to chose an interpretation for numeric volume indicators.

6 Conclusions

In this paper we have discussed our experience with a small-scale evaluation of a Semantic Web-based application. This application is designed as a generic information browser for annotated resources, called siles, and is designed similar to file browsers known from desktop operating systems. This evaluation, even if it was carried out at very low cost, greatly improved the design process as it revealed a number of design issues that we consider as relevant for a larger class of applications.

We plan to continue development of our application based on the evaluation results and will extend our evaluation by publishing a prototype for download. In this software we will integrate mechanisms for implicit feedback, that we can gain from analyzing anonymzed activity logs, and explicit feedback through online evaluation forms.

7 Acknowledgements

Parts of this work are funded by FIT-IT grant 812513 from Austrian Federal Ministry of Transport, Infrastructure, and Technology.

References

[1] Nitin Agrawal, William J. Bolosky, John R. Douceur, and Jacob R. Lorch. A Five-Year Study of File-System Metadata. In *Proceedings of the 5th Conference on File and Storage Technologies (FAST '07), San Jose, CA*, 2007.

[2] Ricardo A. Baeza-Yates and Berthier A. Ribeiro-Neto. *Modern Information Retrieval*. ACM Press / Addison-Wesley, 1999.

[3] M. Ted Boren and Judith Ramey. Thinking Aloud: Reconciling Theory and Practice. *IEEE Transactions on Professional Communication*, 43:261–278, 2000.

[4] Laura Downey. Designing Annotation Mechanisms with Users in Mind: A Paper Prototyping Case Study from the Scientific Environment for Ecological Knowledge (SEEK) Project. In *Proceedings of the Semantic Web Personalization Workshop at the ESWC 2006*, 2006.

[5] Davide Eynard, John Recker, and Craig Sayers. An IMAP Plugin for SquirrelRDF. Technical report, HP Labs, 2007.

[6] Jorge Gracia, Vanessa Lopez, Mathieu d'Aquin, Marta Sabou, Enrico Motta, and Eduardo Mena. Solving Semantic Ambiguity to Improve Semantic Web based Ontology Matching. In *Proc. of the 2nd Ontology Matching Workshop (OM'07) at the 6th International Semantic Web Conference (ISWC 2007)*, November 2007.

[7] Tom Heath, John Domingue, and Paul Shabajee. User Interaction and Uptake Challenges to Successfully Deploying Semantic Web Technologies. In *Proceedings of the 5th International Semantic Web Conference (ISWC 2006)*, 2006.

[8] Steve Jones, Shona McInnes, and Mark S. Staveley. A Graphical User Interface for Boolean Query Specification. *Int. J. on Digital Libraries*, 2(2-3):207–223, 1999.

[9] John Lyons. *Semantics*, volume 5 of *Handbücher zur Sprach- und Kommunikationswissenschaft*, chapter General Foundations, pages 1–24. Walter de Gruyter Berlin New York, 1991.

[10] Jakob Nielsen. Improving a human-computer dialogue. *Communications of the ACM*, 33:338–348, 1990.

[11] Jakob Nielsen. Finding Usability Problems Through Heuristic Evaluation. In *CHI '92: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 373–380, New York, NY, USA, 1992. ACM Press.

[12] Jakob Nielsen. The usability engineering life cycle. *Computer*, 25:12–22, 1992.

[13] Jakob Nielsen. *Usability Engineering.* Academic Press, 1993.

[14] C. K. Ogden and I. A. Richards. *The Meaning of Meaning.* Harcourt, 1989.

[15] Dennis Quan, David Huynh, and David R. Karger. Haystack: A Platform for Authoring End User Semantic Web Applications. In *Proceedings of the 2nd International Semantic Web Conference (ISWC 2003).* Springer, 2003.

[16] Leo Sauermann, Ansgar Bernardi, and Andreas Dengel. Overview and Outlook on the Semantic Desktop. In Stefan Decker, Jack Park, Dennis Quan, and Leo Sauermann, editors, *Proceedings of the 1st Semantic Desktop Workshop*, volume 175, Galway, Ireland, November 2005. CEUR Workshop Proceedings.

[17] Leo Sauermann, Andreas Dengel, Ludger van Elst, Andreas Lauer, Heiko Maus, and Sven Schwarz. Personalization in the EPOS Project. In *Proceedings of the Semantic Web Personalization Workshop at the ESWC 2006*, 2006.

[18] Bernhard Schandl. SemDAV: A File Exchange Protocol for the Semantic Desktop. In *Proceedings of the Semantic Desktop and Social Semantic Collaboration Workshop*, volume 202, Athens, GA, USA, November 2006. CEUR Workshop Proceedings.

[19] Bernhard Schandl and Ross King. The SemDAV Project: Metadata Management for Unstructured Content. In *CAMA '06: Proceedings of the 1st International Workshop on Contextualized attention metadata: collecting, managing and exploiting of rich usage information*, pages 27–32, New York, NY, USA, 2006. ACM Press.

[20] Carolyn Snyder. *Paper Prototyping: The Fast and Easy Way to Design and Refine User Interfaces.* Morgan Kaufmann, 2003.

[21] Rick Spencer. The Streamlined Cognitive Walkthrough Method, Working Around Social Constraints Encountered in a Software Development Company. *CHI Letters*, 2:353–359, 2000.

[22] Constantine Stephanidis, editor. *User Interfaces For All — Concepts, Methods, and Tools.* Lawrence Erlbaum Associates, 2001.

[23] Constantine Stephanidis, Demosthenes Akoumianakis, and Alex Paramythis. User Interaction in Digital Libraries: Coping with Diversity through Adaptation. *Lecture Notes In Computer Science*, 1513:717 – 735, 1998.

[24] Max L. Wilson and m.c. schraefel. mSpace: What do Numbers and Totals Mean in a Flexible Semantic Browser. In *Proceedings of the Semantic Web Personalization Workshop at the ESWC 2006*, 2006.

[25] Degi Young and Ben Shneiderman. A Graphical Filter/Flow Representation of Boolean Queries: A Prototype Implementation and Evaluation. *Journal of the American Society of Information Science*, 44(6):327–339, 1993.