# Visual Exploration, Query, and Debugging of RDF Graphs

**Wolfgang Weiss**
JOANNEUM RESEARCH
Forschungsgesellschaft mbH
Wolfgang.Weiss@joanneum.at

**Michael Hausenblas**
JOANNEUM RESEARCH
Forschungsgesellschaft mbH
Michael.Hausenblas@joanneum.at

**Gerhard Sprung**
FH JOANNEUM
University of Applied Sciences
Gerhard.Sprung@fh-joanneum.at

## ABSTRACT
Semantic Web engineers create and explore the content and structure of RDF graphs in order to build Semantic Web applications. Currently, some Semantic Web development and visualisation tools for Semantic Web engineers are available. However, the tasks of exploring, debugging, and querying RDF graphs have been neglected in the past. This paper evaluates a hybrid visualisation technique of RDF graphs for Semantic Web engineers. A prototype has been implemented representing RDF graphs in different textual views and in a graph visualisation. To examine the usefulness of our approach, a usability test, based on typical use cases of Semantic Web engineers, has been done.

### Author Keywords
RDF, visualisation, Semantic Web, Semantic Web engineer, usability

### ACM Classification Keywords
H.5.2 [Information interfaces and presentation (e.g., HCI)]: Evaluation/methodology. D.2.2 [Software engineering]: User interfaces. H3.3 [Information Storage and Retrieval]: Information filtering. I.2.4. [Artificial Intelligence]: Representations.

## INTRODUCTION
The aim of our work is to examine appropriate visualisation and user interface techniques of RDF graphs for Semantic Web engineers, who develop applications for the Semantic Web and have to explore, debug, and query the underlying data model. This paper is based on our research done in [1] and [2].

The Semantic Web extends the current Web, as it brings a machine processible structure to the meaningful content of Web pages. It provides a well defined meaning using metadata represented in the Resource Description Framework (RDF). The RDF model is a graph consisting of nodes and links. Well defined meaning of the content is achieved through using schema languages and ontologies. Ontologies provide a common understanding of a particular domain. They are necessary for the communication between human beings, and to achieve interoperability among different software systems.

The design process of Semantic Web content and ontologies is long-winded and complex. The developers, called Semantic Web engineers have to describe concepts and relationships among the concepts. The task of authoring RDF graphs and ontologies is supported by several tools, which provide good editing capabilities. However, the tasks of exploring, debugging, and querying unknown RDF graphs have been neglected in the last years. Semantic Web engineers have to understand the structure and the content of RDF graphs and have to find errors in existing RDF graphs and ontologies.

To enable Semantic Web engineers to build the long anticipated "Semantic Web killer applications", tools are needed that support the developers in exploring, debugging, and querying RDF graphs and ontologies. These tools have to increase the cognitive support for the Semantic Web engineers to enable a more effective and efficient development cycle, resulting in a reduction of the development costs and improving the quality of Semantic Web applications.

### Requirements
The basic requirements have been collected in the realm according of our project experience over the last years. The requirements include that (i) the application should be Web-accessible to reduce the complexity of the software distribution, (ii) it should combine textual and graphical presentations of the RDF graph, and (iii) the graphical presentation of the RDF graph should be easy to understand and easy to handle, it should present the RDF graph as it is. The textual views are necessary to navigate through the RDF graph. However, it is not the aim to visualise the entire graph at once, because this would result in a cluttering and confusing visualisation, also known as big fat graphs [15]. Moreover, we do not propose to develop a new graph visualisation algorithm. The target users of this application are Semantic Web engineers who create, modify, validate, or even explore RDF graphs and RDF-based ontologies.

The hypothesis according to the objectives and requirements is as follows: *"a graph-based visualisation in combination with a textual representation of the RDF graph meets the requirements of Semantic Web engineers for effectively and efficiently developing, debugging, and exploring RDF graphs and RDF-based ontologies".*

According to former research in our realm [3], three main use cases were identified, representing the work of Semantic Web engineers. These use cases will be used to verify the hypothesis and to examine the usefulness and improvement opportunities of the application.

1. **Debugging:** An inconsistency in an ontology has been found by the Semantic Web engineer. The Semantic Web engineer has to find the problem.
2. **Expressiveness:** An ontology for a new requirement is needed. Therefore, the Semantic Web engineer has to determine whether the vocabulary of an existing ontology meets the requirements and is expandable, or if the ontology has to be engineered from scratch.
3. **Semantic Web content:** The Semantic Web engineer has to embed an RDF graph into an existing HTML Web site. This graph or an existing Semantic Web content has to be validated by the developer.

## RELATED AND EXISTING WORK[1]

RDF [17] is a common framework for exchanging information between applications without loss of meaning. RDF presents metadata about Web resources and identifies these resources by using Uniform Resource Identifiers (URIs). The intention of RDF is to provide a simple way to make statements about Web resources. A statement consists of a resource, a property, and a value, also called an object-attribute-value triple. The triples form the RDF data model, which can be presented as a graph. RDF Schema [22] (RDFS) defines further modelling primitives in RDF to describe other resources. In contrast to RDF and RDF Schema, OWL [23] provides much more expressiveness, which is required for the Semantic Web.

Semantic Web engineers mainly work on the data model level of the Semantic Web. They usually develop RDF graphs and ontologies, but also explore and investigate the structure of even large and unknown RDF graphs and ontologies to build new Semantic Web applications. In contrast to Semantic Web engineers, domain experts often have little computer skills. They are experts in a certain knowledge domain, e.g. a biologist, and therefore they also have to develop ontologies.

### Applications for Semantic Web Engineers

Denny [4] conducted a survey about current ontology editors, covering tools with ontology editing capabilities that can be used to build ontology schemas and instance data. The author describes ontology building as "a not very linear process", because it is necessary to approach the task from several perspectives at once. It is an iterative process, using top-down and bottom-up techniques. In the survey the author asked each respondent about his or her desired future enhancements. The most important enhancements are (i) a higher level of abstraction for knowledge modelling, (ii) a better visual and spatial navigation among different concept trees or concept graphs as well a better understanding of the ontology and (iii) the need for reasoning and problem solving facilities.

Protégé[2] is an open, platform independent environment for creating and editing ontologies and knowledge bases for Semantic Web engineers. The application is extensible by its plug-in architecture which is categorised in backend plug-ins, slot widget, and tab plug-ins. Jambalaya [5] is an interactive plug-in for Protégé, with a suite of tools for viewing ontologies with graph metaphors. It uses a "Simple Hierarchical Multi-Perspective" (SHriMP) visualisation technique which enhances people to browse, explore and interact with complex information spaces. The visualisation tool combines a hypertext following metaphor with animated panning and zooming motions over nested graphs to provide continuous orientation for the user. According to Mutton and Golbeck [7] "this visualisation technique is full of large boxes, overlapped edges, and obscuring much of the associative structure".

IsaViz[3] is a visual tool for browsing and authoring of RDF models. Resource nodes are represented by ellipses, literals as rectangles and properties are displayed as lines with arrows. The tool uses the GraphViz[4] library. As described in [5], the user interface has issues in its interactivity and it is difficult to customise. Furthermore, the application has performance problems in parsing and generating large graphs. On the other hand, it offers facilities for styling the graph by using a stylesheet concept and the capability for exporting the graph into a Scalable Vector Graphics (SVG) [15].

The Tabulator project[5] [6] is a generic data browser for RDF data on the Web, addressed to end-users and Semantic Web engineers. The authors describe that a graph visualisation with circles and arrows "is very intuitive for humans and useful when trying to understand the structure of data", but "it is not an appropriate way to look at data with many nodes and many different properties". Therefore, a tree visualisation, called "outliner mode", is used. Additionally Tabulator provides several other views, such as a Map View, a Table View or a Calendar View as different tabs. Results of queries can be used to visualise

[1] Further research can be found in [1].

[2] Protégé: Ontology Editor and Knowledge Acquisition System, http://protege.stanford.edu/

[3] IsaViz: A Visual Authoring Tool for RDF, http://www.w3.org/2001/11/IsaViz/

[4] Graphviz: Graph Visualization Software, http://www.graphviz.org

[5] The Tabulator project: http://www.w3.org/2005/ajar/tab

geographical data in a map or temporal data in a calendar. Providing a general view of arbitrary data in combination with specific views for particular special data such as time and space is a sweet spot between completely generic and completely domain-specific applications. Although domain-specific applications will always be important and will do better at specific tasks than the general one.

### Debugging Techniques

Error detecting and debugging is an important aspect when developing ontologies. Different techniques for solving errors of OWL ontologies are implemented in the hypertextual ontology development environment Swoop[6]. To diagnosing unsatisfiable concepts in OWL ontologies two families of reasoner-based techniques are used: the so called glass box and black box techniques [25]. The glass box technique [27] extracts and presents information from the internals of the reasoner. This debugging approach is tightly integrated with the reasoning procedure and provides precise results. This technique is used to find the root cause of a contradiction or to determine the relevant axioms that are responsible for the root cause. The black box technique uses the reasoner as an oracle for a certain set of question, such as satisfiability or subsumption in standard inference. This approach helps to isolate the problems, is independent from the reasoner and needs less memory and computational cost than the glass box technique. Debugging technologies are rather complex and they have its limitations. Therefore, the authors point out that it is necessary to provide manual editing and debugging technologies to the user such as undo, redo, version history or a comparator for ontology entities (cf. [25], [26], [27], [28]).

### Visualisation Techniques

A new visualisation technique for large hierarchical ontologies, called CropCircles was introduced by [8] and [9]. The inspiration for the visualisation technique comes from Treemaps (cf. [10]) which is a space filling visualisation. Changes were made in the representation and layout. Circles represent nodes in a tree and every child circle is nested inside its parent circle. The aim of this visualisation technique is (i) to give a topology overview, (ii) to quickly read labels of children, (iii) to easily detect duplications and (iv) to encourage users to look at the data and gain insights from the visualisation. An empirical evaluation has shown that CropCircles performed well against Treemaps and SpaceTrees (cf. [11]) in topological tasks, like finding certain nodes in a class hierarchy.

The aim of the Cluster Map visualisation technique is it to bridge the gap between complex semantic structures and the simple, intuitive user-oriented presentation. Cluster Maps are used to visualise instances of ontologies and its taxonomy. It is therefore appropriate for end-users. Additionally, geometric closeness is related to semantic closeness (cf. [12] p. 45-48).

OntoSphere 3D [13, 14] is a collection of three-dimensional visualisation techniques; it is available as a plug-in for Protégé for visualising ontologies. A novel approach was developed for inspecting and editing ontologies in a three-dimensional space, which is quite natural for humans. The visualisation tool supports operations like zooming, panning and rotating. Additionally it increases the “dimensions” by using different colours, shapes and transparency of nodes and edges. OntoSphere 3D is scalable in visualising ontologies, but there are still open issues in navigating through the visualisation. It is designed to particularly meet the demands of domain experts who have little technical skills in the field of Semantic Web.

The data model of RDF is a graph and the most popular visualisation is a graph with nodes and edges. In [15] this visualisation is called a “Big Fat Graph”. Using graph visualisations of RDF data especially for end users has a number of drawbacks. For example this visualisations are flat and every node is treated as a primary node. Often, users give too much weight to accidents of the graph layout algorithm. Semantic Web engineers should use appropriate user interface techniques for Semantic Web applications. Though, graph visualisations have their place.

## IMPLEMENTATION

### Architecture

The prototype[7] of this project was implemented as an Ajax Web application using Java. It was deployed on the servlet container Apache Tomcat. Ajax is a bundle of existing technologies to make Web applications more interactive. To get access to the prototype, the user needs a Web browser with Scalable Vector Graphics (SVG) [16] support. The SVG support of the browser has to provide at least rendering, animation, and scripting of Scalable Vector Graphics to be able to use all implemented features of this prototype. A number of libraries and frameworks have been used to implement the prototype. Figure 1 gives an overview of the architecture.

[6] SWOOP - Hypermedia-based OWL Ontology Browser and Editor: http://www.mindswap.org/2004/SWOOP/

[7] The prototype is available at: http://sw.joanneum.at:8080/visr/

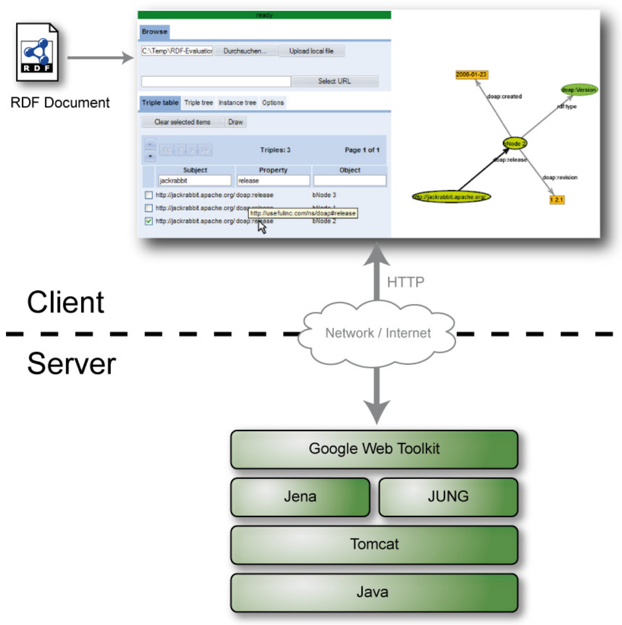


**Figure 1: Architecture of the Prototype**

Jena[8] is an open source framework for building Semantic Web applications implemented in Java. The API of the framework offers a statement-centric and a resource-centric approach to access RDF [17]. The Jena Semantic Web framework is used to read and process the RDF triples on the server of an uploaded file. The Java Universal Network / Graph Framework (JUNG)[9] is a graph visualisation library, which provides a number of layout algorithms and mechanisms to manipulate graphs. In the prototype, the layout of the graph is generated with the “self-organising map layout for graphs” (cf. [18]). An evaluation has shown that this algorithm generates the best results for a graphical representation that is used in this prototype. However, this layout algorithm generates a different layout in every single run. Therefore, it is necessary to animate the graph visualisation for the user. The animation helps the user to follow how the layout changed since the last run. The Batik SVG Toolkit[10] is an open source Apache project for Java applications or applets that want to use images in the SVG format. The project consists of several modules, which can be used independently, or together to generate, view, or convert SVG content. The Google Web Toolkit (GWT)[11] is an open source, Java-based development framework that makes programming Ajax applications easier. It is used to implement the user interface and the services, which exchange data between the client and the server.

## USER INTERFACE

Visualising abstract information helps us to reveal patterns, clusters, gaps, or outlier in statistical data, stock-market trades, or document collections. Humans have remarkable perceptual abilities to scan, recognise, and recall images rapidly and are able to detect changes in size, colour, shape, movement, or texture. This is the reason why visual interfaces have appealing opportunities. Therefore, Shneiderman [19] summarised the basic principles of information visualisation as the Visual Information-Seeking Mantra: “Overview first, zoom and filter, then details-on-demand”. This Visual Information-Seeking Mantra was also the starting point for designing our user interface.

The user interface provides functions to upload RDF files and to visualise and investigate RDF data. The RDF data is illustrated in a graph visualisation with nodes and edges, and in different textual representation. The textual visualisation allows the users to navigate and browse through the graph. To avoid a cluttered visualisation, only selected triples are visualised in the graphical view. The graphical view helps the user to understand the RDF content.

As illustrated in Figure 2, the application has a number of control and display elements for user interaction. A status bar (1) informs the user whether the application is busy or ready to use. (2) is the upload widget to upload local files, stored on the hard disk of the user, or to select remote files, for example with the HTTP protocol. The application accepts RDF/XML files. The user has the choice of three different textual representations (3). The triple table extracts the triples in a table. The triple tree creates a hierarchical view of the graph and the resource tree extracts URI Refs, blank nodes, literals, and properties from the graph. The 4th tab of this widget allows the user to set options of the visualisation. Finally, (4) shows the graphical representation of the graph. This visualisation represents the RDF graph as it is - without abstracting or obscuring information.

### *Triple Table*

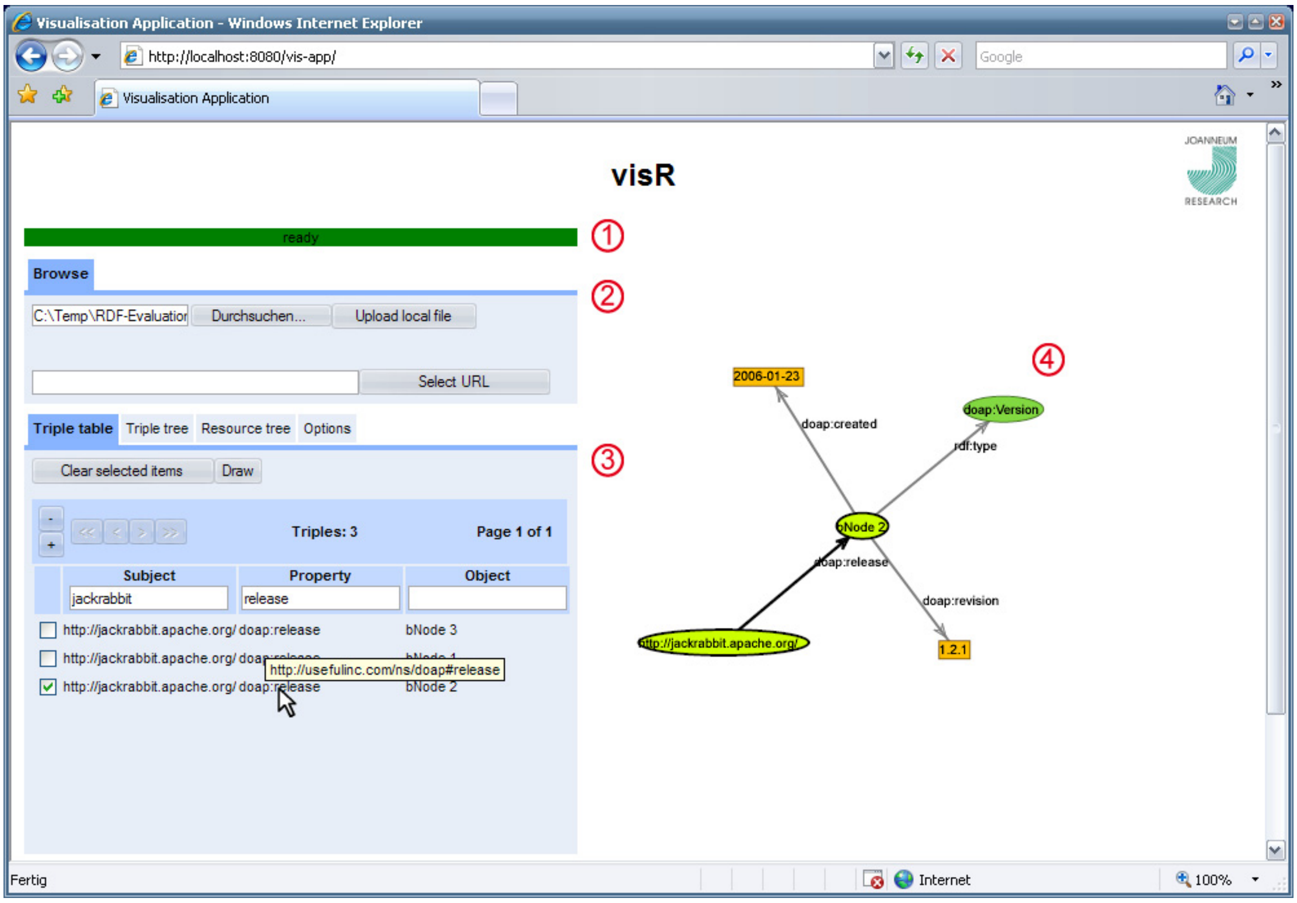
The screenshot of Figure 3 illustrates the triple table. This table shows the RDF triples (subject, property, and object). It loads only a certain number of triples from the server and displays it. The number of visible triples can be increased and decreased by (1). This feature is useful when the RDF graph contains more than 100 triples. If the graph contains more triples than can be displayed in a single view the triples are split up in several pages. The user has the possibility to switch (2) between the pages. The number of currently available triples and pages is displayed in the header (3) of the table. To find the right information, it is possible to *“filter”* (cf. [19]) the triples by simply using the text boxes (4) of the subject, property, or object. The user

[8] Jena: A Semantic Web Framework for Java, http://jena.sourceforge.net/

[9] JUNG - Java Universal Network / Graph Framework, http://jung.sourceforge.net/

[10] Batik SVG Toolkit, http://xmlgraphics.apache.org/batik/

[11] Google Web Toolkit, http://code.google.com/webtoolkit/



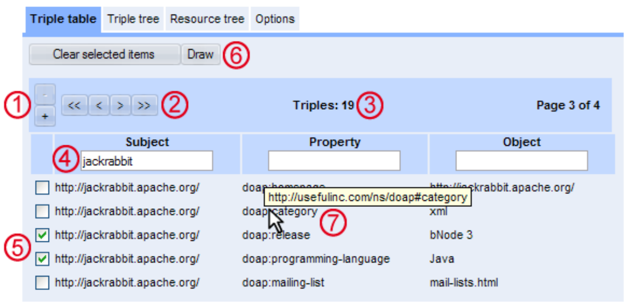
Triples: 3 — Page 1 of 1

| | Subject | Property | Object |
|---|---|---|---|
| | jackrabbit | release | |
| ☐ | http://jackrabbit.apache.org/ | doap:release | bNode 3 |
| ☐ | http://jackrabbit.apache.org/ | doap:release | bNode 1 |
| ✓ | http://jackrabbit.apache.org/ | doap:release | bNode 2 |

http://usefulinc.com/ns/doap#release

**Figure 2: Screenshot of the Prototype**

can select any triple by clicking the check-box of a triple (5) to visualise it in the graphical view. The buttons of (6) clear the selection of selected triples or redraw the graph in the graphical view. Additionally, when the user moves the mouse over an item of the triple, *detailed information* is displayed in a tool tip (7).



Triples: 19 — Page 3 of 4

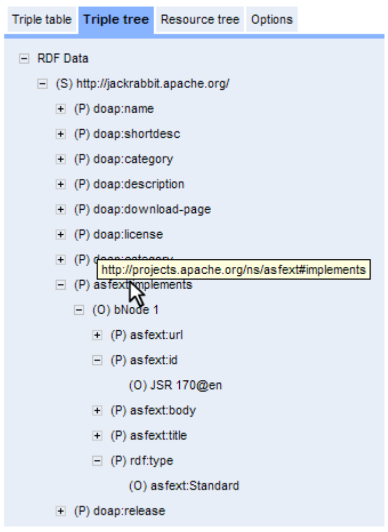
| | Subject | Property | Object |
|---|---|---|---|
| | jackrabbit | | |
| ☐ | http://jackrabbit.apache.org/ | doap:homepage | http://jackrabbit.apache.org/ |
| ☐ | http://jackrabbit.apache.org/ | doap:category | xml |
| ✓ | http://jackrabbit.apache.org/ | doap:release | bNode 3 |
| ✓ | http://jackrabbit.apache.org/ | doap:programming-language | Java |
| ☐ | http://jackrabbit.apache.org/ | doap:mailing-list | mail-lists.html |

http://usefulinc.com/ns/doap#category

**Figure 3: Screenshot of the triple table widget.**

*Triple Tree*

The triple tree, illustrated in Figure 4, is a hierarchical representation of the RDF graph. Users can browse through the graph just like in a tree of a file browser. This representation loads the whole RDF graph at once. It is ideal for small compact hierarchies which can be represented on a single screen without having to scroll (cf. [10] p. 3). The triple tree is currently in an experimental stage. It was designed to find out which textual representation best fits the needs of Semantic Web engineers.

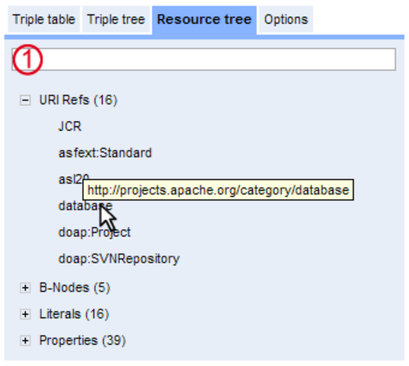


```
⊟ RDF Data
  ⊟ (S) http://jackrabbit.apache.org/
    ⊞ (P) doap:name
    ⊞ (P) doap:shortdesc
    ⊞ (P) doap:category
    ⊞ (P) doap:description
    ⊞ (P) doap:download-page
    ⊞ (P) doap:license
    ⊞ (P) doap:category
    ⊟ (P) asfext:implements
      ⊟ (O) bNode 1
        ⊞ (P) asfext:url
        ⊟ (P) asfext:id
            (O) JSR 170@en
        ⊞ (P) asfext:body
        ⊞ (P) asfext:title
        ⊟ (P) rdf:type
            (O) asfext:Standard
    ⊞ (P) doap:release
```
http://projects.apache.org/ns/asfext#implements

**Figure 4: Screenshot of the triple tree widget.**

### *Resource Tree*

Also the resource tree (see Figure 5) of the prototype is an experimental view and was designed to find out the needs of Semantic Web engineers. It splits the RDF graph up into URI references, blank nodes, literals, and properties. Properties are also URI references, but, they are displayed separately. This view loads the whole RDF graph at once, but in contrast to the triple tree, it has a text box to *search* and *filter* inside the RDF graph.

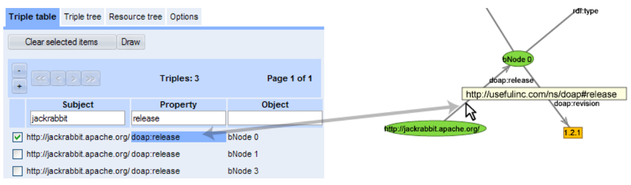


```
⊟ URI Refs (16)
    JCR
    asfext:Standard
    asl20
    database
    doap:Project
    doap:SVNRepository
⊞ B-Nodes (5)
⊞ Literals (16)
⊞ Properties (39)
```
http://projects.apache.org/category/database

**Figure 5: Screenshot of the resource tree widget.**

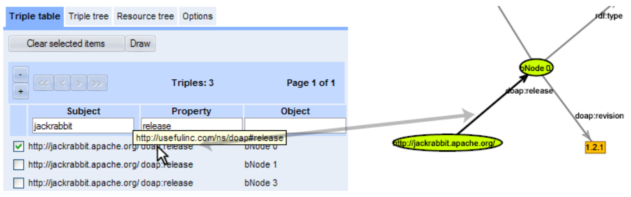
### *Graphical Representation*

The graphical representation, a SVG, includes graphical entities, such as nodes and edges, as well as animation and scripting. The scripting of the SVG provides user interactions on the graphics. For example, the user gets *detailed information* when the mouse pointer is moved over an edge or a node (see Figure 6), which is also known as *"details on demand"*. Furthermore, the corresponding nodes or edges are highlighted in the textual representation at the same time, which is also known to *"relate"* items (cf. [19]). When the user moves the mouse over a triple or a resource in the textual representation, the corresponding resources are highlighted in the graphical representation (illustrated by Figure 7). The used visualisation algorithm produces a new layout at each run. Therefore, it is necessary to animate the transition of the graph when the visualisation is reloaded.



| | Subject | Property | Object |
|---|---|---|---|
| | jackrabbit | release | |
| ☑ | http://jackrabbit.apache.org/ | doap:release | bNode 0 |
| ☐ | http://jackrabbit.apache.org/ | doap:release | bNode 1 |
| ☐ | http://jackrabbit.apache.org/ | doap:release | bNode 3 |

**Figure 6: Details on demand and highlighting corresponding resources of the textual and graphical representation.**



| | Subject | Property | Object |
|---|---|---|---|
| | jackrabbit | release | |
| ☑ | http://jackrabbit.apache.org/ | doap:release | bNode 0 |
| ☐ | http://jackrabbit.apache.org/ | doap:release | bNode 1 |
| ☐ | http://jackrabbit.apache.org/ | doap:release | bNode 3 |

**Figure 7: Highlighting corresponding resources of the textual and graphical representation.**

## USABILITY TEST

The aim of usability testing is to get feedback from actual users performing real tasks. The process of usability testing is sometimes called "usability engineering". This relates to the term in the concept and to the importance to software engineering and it is a critical part of product development.

### Usability Test Setup

The aim of this usability test is to verify the hypothesis and to examine the usefulness of the combination of a graphical and a textual representation of RDF graphs for Semantic Web engineers, according to the previously defined use cases. The evaluation was separated in four parts: (i) introduction, (ii) user tasks including observation, (iii) user tasks including measurement of time and task completion, and (iiii) a questionnaire. It was carried out with seven participants. Each subject has experience in RDF and RDF Schema. As illustrated in the Table 1, the average skill level / experience of the participants is 3.5, according to a self assessment of the part four of the usability test.

| Min | Max | Average | Median |
|---|---|---|---|
| 3 | 4 | 3.5 | 3.5 |

**Table 1: Self-assessment of each participant about their skill level of RDF(S), on a scale between 1 (low) and 5 (excellent).**

Each usability test lasted approximately 1 hour and 20 minutes. The setup of this usability test is partly based on a mixture of expert interviews and user interviews (cf. [20] pp. 5-44). The users are Semantic Web engineers, who are the focus group of the application. The Semantic Web engineers are also software engineers and experts in designing user interfaces for applications.

**Part 1**
The introduction covered a refresh of RDF and RDFS, and an explanation of the application, how the application works and the functionality of each widget.

**Part 2**
The second part of the usability test was the user tasks including the observation of the participant where any specific behaviour was noted. Additionally a screen movie of all tasks done by the users on the computer, including the voice of the users, was recorded to analyse the behaviour of the users and the usability of the application after the test.

The users had to do following tasks: Do simple tasks, such as uploading a file and visualising certain triples. The participants had to describe the property "implements" of a description of a project[12] (DOAP) file from the Apache Jackrabbit[13] project. Then the users had to find an error in this RDF graph, in which the date of the latest version had an error. The problem of this RDF graph was that the property "created" in the latest version was wrong (see also Listing 1).

```
@prefix :
<http://usefulinc.com/ns/doap#> .
:release
   [a :Version;
   :name       "Apache Jackrabbit 1.2.1";
   :created    "2006-01-23"@en;
   :revision   "1.2.1"@en
   ];
:release
   [a :Version;
   :name       "Apache Jackrabbit 1.2.2";
   :onDate     "2007-02-21"@en;
   :revision   "1.2.2"@en
   ];
```

**Listing 1: Excerpt of the used DOAP File.**

The next task was to find out the expressiveness of two ontologies. The first ontology was a schema for metadata for photos[14], similar to Exif [24] data. The users had to find out whether it is possible to represent geographical information with this schema. The answer was: "no". Then, the users got another schema for representing geographical information[15], where the participants hat to find out how to use this schema in an RDF document.

**Part 3**
The third part of the usability test covered again user tasks but the task completion and duration to complete the each task was measured. These user tasks consisted of 13 tasks where each subject had to answer to "multiple choice" and "fill-in blank" questions (Questions: 2.1, 3.2, 5.1, 6.1) such as:

- Which "rdf:Properties" are defined in this RDF Schema?
    - Answer 1: trackList
    - Answer 2: Satus
    - Answer 3: releaseType
    - Answer 4: trackNum
- Complete following sentences: The resource "madeFrom" is of "rdf:type" ________ it has a "rdfs:domain" of ________ and it has a "rdfs:range" of ________.

and to open questions (Questions: 1.1, 1.2, 2.2, 3.1, 4.1, 5.2, 6.2, 7.1, 7.2), where the participants had to answer in short sentences.

- This is a simple RDF Schema, describe all "rdf:Properties".
- This RDF file describes some institutions and their location. Find out the geographic location of the resource "Joanneum-Research".
- This is an ontology about music instruments and its players. Describe the resource "m:Contrabass" and its super classes.

To perform the test RDF files, RDF Schemas and OWL ontologies containing 12 - 1600 Triples were used.

**Part 4**
The last part of the usability test covered the feedback questionnaire, which was adapted from "Questionnaire for User Interaction Satisfaction" [21] pp. 172–182 and it contained 15 questions. Each subject had to answer to questions about the:

*1. User interface*
- The readability, concerning the characters and signs on the screen.
- The helpfulness of tooltips and emphasis of certain elements.
- The simultaneously visible information.

[12] Description of a Project: http://usefulinc.com/doap/

[13] Apache Jackrabbit - RDF DOAP file: http://jackrabbit.apache.org/doap_Jackrabbit.rdf

[14] Photography Vocabulary: http://purl.org/net/vocab/2003/11/photo.rdf

[15] Basic Geo Vocabulary: http://www.w3.org/2003/01/geo/wgs84_pos.rdf

- Order of elements on the screen.

*2. Functionality*
- The combination of textual and graphical representation of RDF data.
- Missed features.

*3. Scalability and Responsiveness*
- Reaction time of the system.

*4. Learnability*
- Whether the usage of the system easy to learn.
- Whether it is possible that domain experts use this application.

*5. Other*
- Level of difficulty of the tests.

The questionnaire also included the paper prototype. The users had to assess whether a facet oriented view, also known as mulit-pane browsing, would be more useful than the currently available views for the RDF graph. The facet oriented view provides a "browsing through the graph" metaphor.

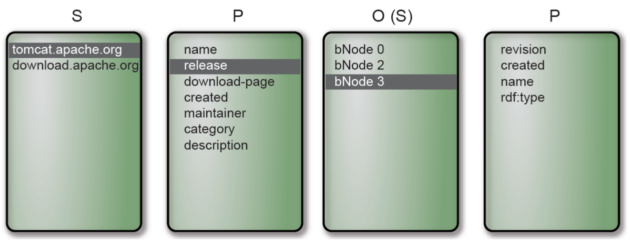


**Figure 8: Paper Prototype.**

### Results

*Results of the Part 3*

All participants were able to solve all questions of the part 3 of the usability test correctly. Figure 9 illustrates the needed time to complete each question. The skill level and experience of each participant was different and some participants had problems in understanding some questions. This fact explains the great variation of the time to complete some tasks, especially of question 5.1.

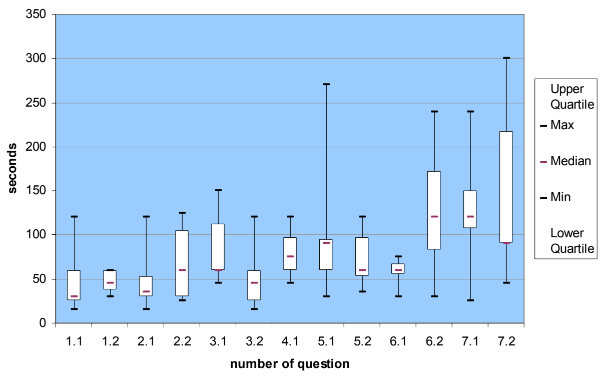


**Figure 9: Duration to complete each question**

*Results of the Parts 2 and 4*

According to the questionnaires and the observations during the second part of the usability test, following results were identified:

1. Usability (Readability, design of the user interface, details on demand, arrangement of the components)

- The results have shown that the combination of a textual and a graph-based visualisation of RDF graphs are useful for Semantic Web engineers.
  - It is easier for the users to understand what a certain RDF graph expresses.
  - The visualisation is the hub of different textual representations of the RDF graph.
- All information on the screen is clear and easy to read. The components, also known as widgets, are easy to access.
- All participants agreed that tooltips, which give information on demand, are helpful.
- The users agreed that the amount of simultaneously visible information and the order of the widgets are suitable.
- The visualisation algorithm should work more precisely, since it sometimes produces overlapping nodes.

2. Functionality

- Users who are familiar with the SPARQL query language claimed to use it in the application.
- The "resource tree" should extract more information of the RDF graph.
- Users want to sort the results in the "triple table" and claimed to use better filtering functionality, such as regular expressions.

3. Scalability and Responsiveness

- The reaction time of the application is in general adequate. However, the time until the trees are generated could be a bit faster.

4. Learnability

- The application is easy to learn and intuitive to use.
- The users agreed that the application would be useful for domain experts, if they have enough experience in RDF.

In conclusion, the usability test has shown that the combination of a graph visualisation and different textual representations of the RDF graph is helpful for users who want to explore or find an error in an RDF graph. Some users used the graph visualisation extensively others used it only to "memorise" certain nodes, but both user groups agreed that the graph visualisation is helpful. The basic functions are easy to use and well arranged. The triple table was very helpful to find the error in the RDF graph in the usability test and the users appreciated the filter functions of this widget. The triple tree was useful to explore ontologies and schemas. The resource tree widget was not helpful for the users; it should extract more or other information. The participants had different opinions about the facet-oriented view of the paper prototype. They believe that it is easier to use, but not better than the "triple tree" which has already been implemented in the application. However, it would make sense to additionally use a facet-oriented view.

## CONCLUSION

The Semantic Web builds on RDF and statements form the RDF data model which is a graph. Information provided by RDF is intended to be processed by applications rather than being only displayed to people. This is one reason that makes developing Semantic Web applications for Semantic Web engineers that difficult.

The Semantic Web engineers mainly work on the data model level of RDF. They usually develop RDF graphs and ontologies, but also explore and investigate the structure of large RDF graphs and ontologies to build Semantic Web application. Currently, several good applications for authoring and editing RDF graphs and ontologies exist. However, the tasks of exploring, debugging, and querying unknown RDF graphs have been neglected in the past. Therefore, new tools for Semantic Web engineers are needed to explore and investigate the structures of large RDF graphs and ontologies and to more easily share common concepts over different development groups. Special attention is paid to the cognitive support of the users to enable a more effective and efficient development cycle.

A prototype[16] has been implemented to examine appropriate visualisation and user interface techniques for Semantic Web engineers. The aim of this prototype is to support the users in understanding the structure and content of RDF graphs as well as to find errors in RDF graphs. The prototype has been implemented as a Web application. It contains three different textual presentations of RDF graphs as tables and trees and a graphical presentation of the RDF graph. The visualisation presents the RDF graph as it is - without abstracting the graph or obscuring information. A usability test has been carried out to examine the usefulness of these presentations. The usability test has shown that the combination of a graph visualisation and different textual presentations of the RDF graph is helpful for users who want to explore or find an error of an RDF graph. The different views, especially the graph visualisation, of the RDF graph makes it easier for the users to understand what a certain RDF graph expresses and the visualisation is the hub of different textual representations of the prototype.

However, this paper illustrates only one aspect of Semantic Web development tools; such a tool should also support the reasoning and querying of RDF graphs and provide more abstract views on RDF graphs and ontologies.

## ACKNOWLEDGMENTS

Parts of the research presented herein were carried out in the "Understanding Advertising" (UAd)[17] project, funded by the Austrian FIT-IT Programme.

## REFERENCES

1. Wolfgang Weiss. (2007). *Visual Exploration, Query, and Debugging of RDF Graphs.* Master Thesis, FH JOANNEUM, University of applied sciences.
2. Michael Hausenblas, Wolfgang Weiss. (2007). *The Needs of Semantic Web Engineers.* Position Statement, SWUI 2007.
3. Michael Hausenblas, Herwig Rehatschek. (2007). *mle: Enhancing the Exploration of Mailing List Archives Through Making Semantics Explicit.* In Proc. ISWC, Semantic Web Challenge 2007, Busan, South Korea, November 13th, 2007.
4. Michael Denny. (2004). *Ontology Tools Survey.* http://www.xml.com/pub/a/2004/07/14/onto.html
5. Neil A. Ernst, Margaret-Anne Storey and Polly Allen. (2004). *Cognitive Support for Ontology Modeling.* International Journal of Human-Computer Studies, Volume 62, Issue 5, May 2005, Pages: 553 – 577, ISSN:1071-5819
6. Tim Berners-Lee, Yuhsin Chen, Lydia Chilton, Dan Connolly, Ruth Dhanaraj, James Hollenbach, Adam Lerer, and David Sheets. (2006). *Tabulator: Exploring and Analyzing linked data on the Semantic Web.* SWUI Workshop at ISWC 2006, Athens, Georgia, November 5-9, 2006.

[16] The prototype is available at: http://sw.joanneum.at:8080/visr/

[17] Understanding Advertising: http://www.sembase.at/index.php/UAd

7. Paul Mutton and Jennifer Golbeck. (2003). *Visualization of Semantic Metadata and Ontologies.* In Proc. InfoVis 2003, Seattle, WA, USA, 20-21 October 2003.

8. Bijan Parsia, Taowei Wang, and Jennifer Golbeck. (2005). *Visualizing Web Ontologies with CropCircles.* In Proc. ISWC 2005, Workshop on End User Semantic Web Interaction, Galway, Ireland, November 7, 2005.

9. Taowei David Wang and Bijan Parsia. (2006). *Crop Circles: Topology Sensitive Visualization of OWL Class Hierarchies.* In Proc. ISWC 2006, Athens, Georgia, November 5-9, 2006, Pages: 695 – 708.

10. Keith Andrews. (2002). *Visualising Information Structures: Aspects of Information Visualisation.* Professorial Thesis, IICM, University of Technology, Graz, Austria, November 2002.

11. Catherine Plaisant, Jesse Grosjean, and Benjamin B. Bederson. (2002). *Space-Tree: Supporting Exploration in Large Node Link Tree, Design Evolution and Empirical Evaluation.* In Proc InfoVis 2002, Boston, MA, 27 October - 1 November 2002, IEEE, 57-64.

12. Vladimir Geroimenko and Chaomei Chen. (2005). *Visualizing the Semantic Web: XML-based Internet and Information Visualization.* Springer; 2nd edition, ISBN 1852339764

13. Alessio Bosca, Dario Bonino, and Paolo Pellegrino. (2005). *OntoSphere: more than a 3D ontology visualization tool.* In Proc. Italian Semantic Web Workshop, University of Trento, Italy, 14 – 16 December 2005.

14. Alessio Bosca and Dario Bonino. (2006). O*ntoSphere3D: a multidimensional visualization tool for ontologies.* DEXA 2006, Andrzej Frycz Modrzewski Cracow College, Krakow, Poland, 4 - 8 September 2006, ISBN: 0-7695-2641-1, Pages: 339 – 343.

15. David Karger, M.C Shraefel. (2006). *The Pathetic Fallacy of RDF.* SWUI Workshop at ISWC 2006, Athens, Georgia.

16. Jon Ferraiolo, Fujisawa Jun, and Dean Jackson. (2003). *Scalable Vector Graphics (SVG) 1.1 Specification.* http://www.w3.org/TR/2003/REC-SVG11-20030114/

17. Graham Klyne and Jeremy J. Carroll. (2004). *Resource Description Framework (RDF): Concepts and Abstract.* Syntax, http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/

18. Bernd Meyer. (1998). *Self-Organizing Graphs - A Neural Network Perspective of Graph Layout.* In Proc. Graph Drawing 1998, Montréal, Canada, August 1998, ISBN: 978-3-540-65473-5, Pages: 246 - 262.

19. Ben Shneiderman. (1996). *The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations.* vl, p. 336, IEEE Symposium on Visual Languages, 1996, Pages: 336 – 343.

20. Carol M. Barnum. (2002). *Usability Testing and Research.* Pearson Education. ISBN 0205315194, 2002.

21. Ben Shneiderman. (2002). *User Interface Design.* Third Edition. ISBN 3826607538, 2002.

22. Dan Brickley and R.V. Guha. (2004). *RDF Vocabulary Description Language 1.0: RDF Schema.* http://www.w3.org/TR/2004/REC-rdf-schema-20040210/

23. Michael K. Smith, Chris Welty, and Deborah L. McGuinness. (2004). *OWL Web Ontology Language Guide.* http://www.w3.org/TR/2004/REC-owl-guide-20040210/

24. Standard of Japan Electronics and Information Technology Industries Association. *Exchangeable image file format for digital still cameras: Exif Version 2.2.* (2002). http://www.digicamsoft.com/exif22/exif22/html/exif22_1.htm

25. Bijan Parsia, Evren Sirin, and Aditya Kalyanpur. (2005). *Debugging OWL Ontologies.* In Proc. WWW2005, Chiba, Japan, May 2005.

26. Aditya Kalyanpur, Bijan Parsia, and Evren Sirin. (2005). *Black Box Techniques for Debugging Unsatisfiable Concepts.* In Proc. DL 2005, Edinburgh, Scotland, UK, July 26–28, 2005.

27. Aditya Kalyanpur, Bijan Parsia, Evren Sirin, and James Hendler. (2006). *Debugging Unsatisfiable Classes in OWL Ontologies.* Journal of Web Semantics, Volume 3 Issue 4, 2006.

28. Aditya Kalyanpur, Bijan Parsia, Evren Sirin, and Bernardo Cuenca-Grau. (2006). *Repairing Unsatisfiable Concepts in OWL Ontologies.* In Proc. ESWC 2006, Budva (Montenegro), 11 – 14 June, 2006.