Working Paper Series
ISSN 1177-777X

A Review of User Interface Adaption in Current Semantic Web Browsers

Emmanuel Turner, Annika Hinze, Steve Jones

Working Paper: 02/2011
February 2011

Department of Computer Science
The University of Waikato
Private Bag 3105
Hamilton, New Zealand

A Review of User Interface Adaption in Current Semantic Web Browsers

Emmanuel Turner, Annika Hinze, Steve Jones
University of Waikato, Hamilton, New Zealand
{eturner,hinze,stevej}@cs.waikato.ac.nz

8 February 2011

Abstract

The semantic web is an example of an innumerable corpus because it contains innumerable subjects expressed using innumerable ontologies. This paper reviews current semantic web browsers to see if they can adaptively show meaningful data presentations to users. The paper also seeks to discover if current semantic web browsers provide a rich enough set of capabilities for future user interface work to be built upon.

1 Introduction

This working paper examines the current (December 2010) state of the art in Semantic Web Browsers. The purpose of this research is twofold. Firstly it will be established that there is sufficient research and working examples of current semantic web browsers to build upon. Secondly it argues that current semantic web-browsers do not select appropriate data presentations for an innumerable corpus such as the semantic web. An innumerable corpus has an innumerable number of subjects expressed in innumerable ontologies.

If these two points can be established then it shows that a gap in currently available browser software exists around data presentation which can further research can explore and that there is enough existing research in place that an exploration into an adaptive semantic web browser has a good chance of success.

In answering the hypothesis the following questions are explored:

1. What is the current state of the art in semantic web browser capability?
2. How do current semantic web browsers allow for alternative data presentations?

Question 2 assumes that a good semantic web browser takes advantage of the innumerable ontologies present on the semantic web by providing alternative data presentations to best suit the subject and the user. The ideal semantic web browser for the innumerable corpus is one that is able to adapt to meaningfully present data from any ontology and will also be able to adapt that data presentation to suit an individual user.

The format of the paper outlines the case study methodology in Section 2, describes and evaluates each browser in turn (Section 3) and then the browsers are discussed in comparison with each other (Section 4). Finally, a summary is made and conclusions drawn about the hypothesis (Section 5).

2 Methodology

The methodology followed is a case study that takes the form of a series of software reviews that focus on gaining knowledge that may inform the hypothesis. Following good case study methodology, firstly each semantic web browser is reviewed independently then the browsers are compared and contrasted against each other. Finally conclusions are drawn from the data gathered in terms of the hypothesis.

The semantic web browsers that are reviewed are: BrownSauce (Steer, 2003), Disco (Bizer & Gauß, 2007), Exhibit (Huynh, Karger, & Miller, 2007), Marbles (Becker & Bizer, 2009), ObjectViewer (Lerner & Self, 2004), Tabulator (Tabulator Team, 2008), and Zitgist DataViewer (OpenLink Software, 2009). These browsers reflect both the current state of the art and/or have some interesting properties that are worth investigating. There are many other semantic web browsers but these are not reviewed because they were either unavailable to the researcher or have no significant properties that, from a preliminary examination, are not already explored by the semantic web browsers above. Notable semantic web browsers that were not reviewed are: Longwell (MIT, 2005), /facet (Hildebrand, van Ossenbruggen, & Hardman, 2006), BrowseRDF (Oren, Delbru, & Decker, 2006), Explorator (de Araújo & Schwabe, 2009), IsaViz (W3C, 2007), OpenLink Data Explorer (OpenLink Software, 2006), Noadster (Rutledge, van Ossenbruggen, & Hardman, 2005) and MindRaider (Dvorak, 2008).

Software was reviewed on a Windows Vista 32-bit computer, quad-core with 4GB of RAM. The screen resolution is Full HD 1920x1080 at 32-bit colour. Where applicable and not otherwise stated the default HTML web browser used is Firefox Portable version 3.6.8. The semantic web browsers software reviewed was the current version as at August 2010.

Each case study begins with a description of the browser followed by an evaluation of the browser against key criteria. (The criteria are outlined in the next section.)

The description briefly introduces the semantic web browser project to help contextualize the later data. Then setup of where the browser logic actually runs is explained. Following this are two screenshots of the browser showing real subjects. One screenshot will show Tim Berners-Lee's FOAF file (http://www.w3.org/People/Berners-Lee/card#i), the other screenshot shows a subject that shows the data presentation capabilities of the semantic web browser. The second screenshot will be either Berlin from GeoNames (http://sws.geonames.org/2950159/about.rdf) or a test calendar file (http://www.w3.org/2002/12/cal/test/bus-hrs.rdf) .Next is a description of the user interface. Following this is a description of which data presentations the browser allows and how the data presentations are selected.

This evaluation section scores the semantic web browser against a set of criteria formed from the hypothesis. The evaluation section of each case study is divided into two parts. First the capabilities of each browser is analysed and secondly the data presentation handling is evaluated.

2.1 Browser Capability Criteria

This set of criteria scores each browser for its capabilities. This evaluation seeks to establish whether current semantic web browsers provide a mature enough platform upon which to build further research upon. These criteria are scored using a range of --, -, 0, + and ++. A total score is given for each semantic web browser. Higher scores (more +s) are considered better.

Eases of Use: A candidate scores well for potential ease of use and setup. The highest scores go to web server hosted semantic web browsers (++) followed by local applications and HTML web browser plugins (+). Candidates that must install a local server score the lowest (--). This criterion should be seen as having a lesser importance than the others.

Supported Data Sources: A candidate scores maximum marks for supporting multiple unbounded sources that can be supplied by the user a run-time (++). Fewer marks are awarded for only displaying data from a single user provided data source at a time (+). The lowest marks are awarded to a candidate that does not allow users to specify the data source, having the data source hard coded at author-time (--).

Data Formats: Candidates score well for supporting most of the common document data formats used to exchange semantic web data (++). Candidates score poorly for supporting only a single document standard data format (-). A browser would score -- if only proprietary data formats are supported.

2.2 Data Presentation Criteria

Data presentation criteria evaluate each of the case study subjects on how well they meet criteria relating to the second hypothesis. These criteria are a quality judgment on how semantic web browsers handle alternative data presentations in response to different users and different ontologies. In order to solve the problems of dealing with an innumerable ontology, a semantic web browser would be in the highest in each of the data presentation criteria.

Ontology Adaption: This criterion gauges how a particular semantic web browser adjusts the data presentation when displaying subjects from different onotlogies. The allowable values, in worst to best order, are:

1. None

   The candidate does not change the display based on different ontologies.
2. Fixed

   The candidate has a fixed number of alternative data presentations that match particular ontologies and these are hardcoded at compile time.
3. Extensible

   The candidate has a template system for extending the alternative data presentations. Templates may be added after compile time. The skills set for producing templates are within reach of a power user.
4. Innumerable

   The candidate will adaptively produce a data presentation for every different ontology encountered. Each data presentation will aim to be more meaningful for an average user than a generic static display.

User Adaption: This criterion gauges how a particular semantic web browser adjusts the data presentation based upon its knowledge of the current user. The allowable values, in worst to best order, are:

1. None

   The candidate makes no attempt to provide data presentations that are more meaningful to the individual user.

2. Fixed

   The candidate will change the data presentation based upon placing the user into one of a fixed number of groups that are specified at compile time (e.g. librarian, enrolment officer, genealogist).

3. Open

   The candidate changes the data presentation to suit an individual user based upon a user model. There are no fixed categories of users – the user model allows each user to be treated uniquely.

User Affordance: This criterion gauges the user interface affordances provided by a particular semantic web browser to change to an alternative data presentation. The allowable values, in worst to best order, are:

1. None

   The candidate has does not provide alternative data presentations so no user affordance is provided, or indeed, is necessary. A candidate that is placed here will have no ontology adaption and no user adaption.

2. Manual

   The candidate provides an affordance for users to change to an alternative data presentation but this must be deliberately operated by the user.

3. Automatic

   The candidate selects the best data presentation and automatically uses that one. No user interface affordance is necessary.

3 Results

The Results section examines each of the featured semantic web-browsers in turn to examine what they offer the user. This section focuses on exploring the unique attributes of each browser in isolation. Later sections will compare and contrast the features of each semantic web browser and will examine each against the hypothesis. The purpose of this section is to provide the raw data that the later Findings and Conclusions sections will draw upon.

3.1 BrownSauce

BrownSauce (Steer, 2003) is a web browser based system for viewing RDF documents. The homepage for the project is at: http://brownsauce.sourceforge.net. The current version as at August 2010 is 0.1.2. The name is a reference to the old HotSauce hypertext program.

BrownSauce is written in java and runs a local web server that is then accessed via a web-browser. All code is executed in the web server and the browser is only used to display the resulting HTML/CSS pages.

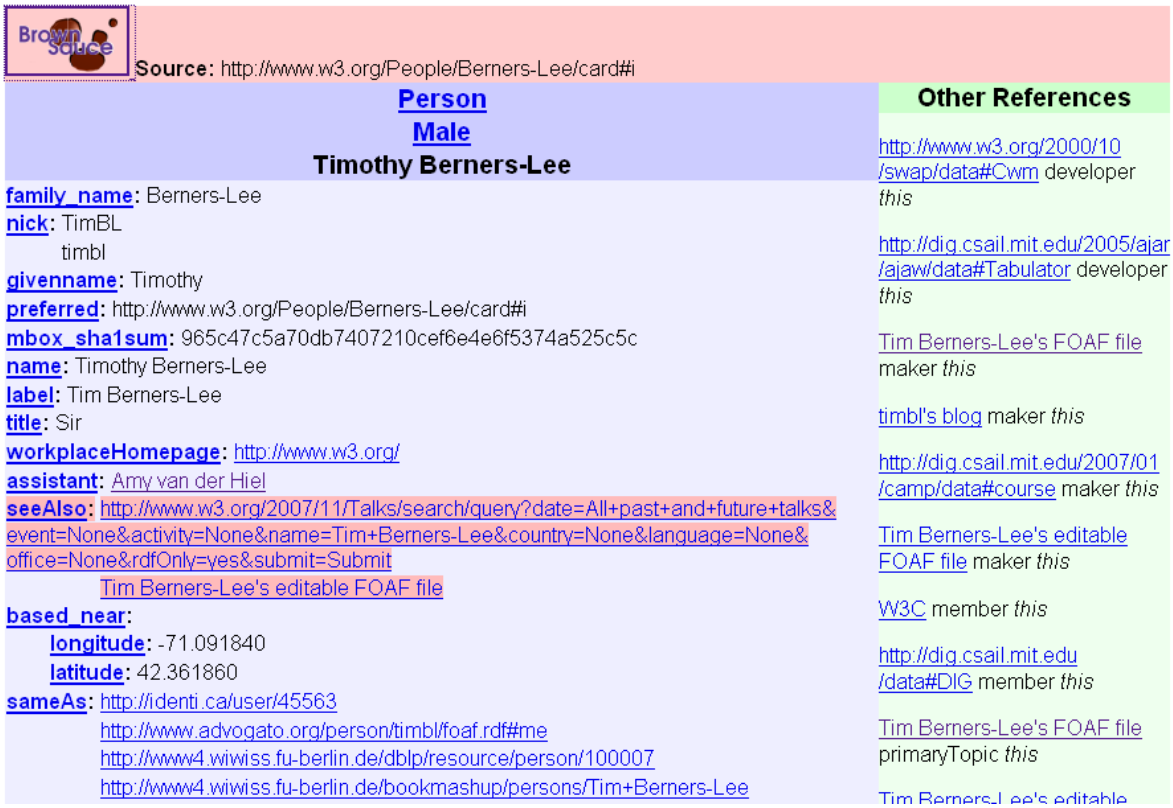

Figure 3.1: BrownSauce screen shot showing data for Tim Berners-Lee. (Author's own)

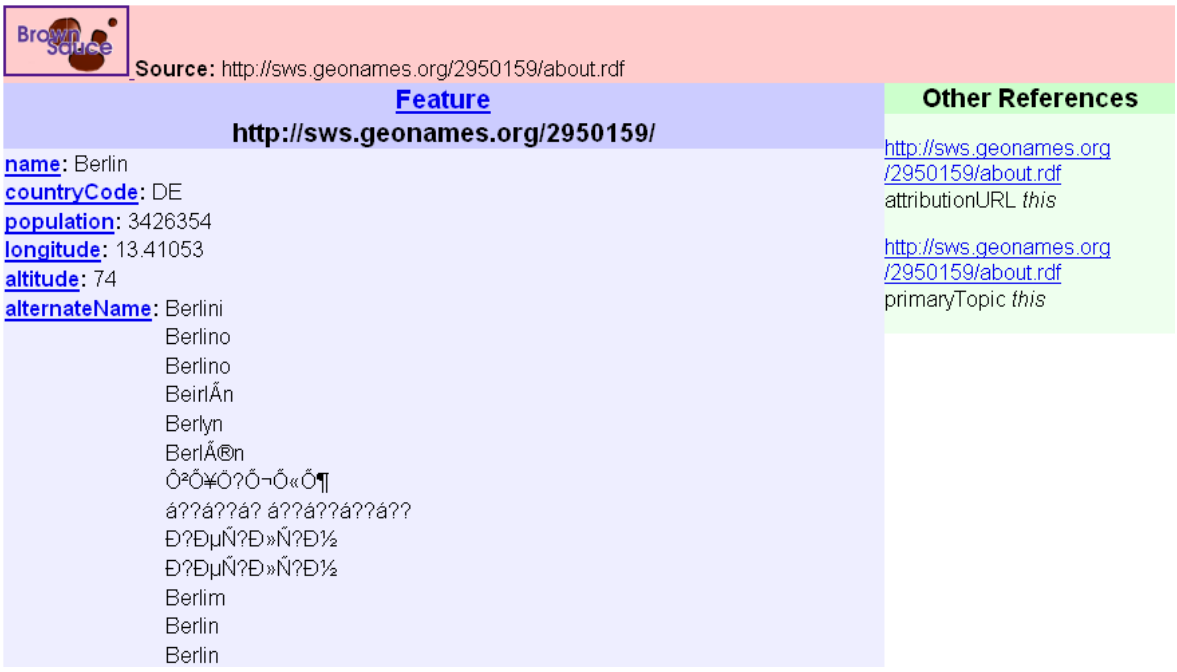

Figure 3.2: BrownSauce screen shot showing Berlin from GeoNames (http://sws.geonames.org/2950159/about.rdf).

BrownSauce starts with a plain web-form where a semantic web URI can be submitted. The display that shows a semantic web document has the current URI at the top followed by a banner showing the label and data-types of the current subject. To the right is a panel containing links to known subjects that relate to the current subject. The main area (to the left) displays the subject's data.

The current subject's data is displayed as an indented text list of RDF predicate-object pairs. Much of the data is hyperlinked where-ever possible and the hyperlinks use plain-text labels rather than URIs when labels are available. No images are shown. No other data presentations are available.

It is possible for the person running the web server to edit BrownSauce's CSS file to change the presentation – though such changes are globally applied to all semantic web subjects subsequently viewed via BrownSauce. Following this is a commentary on the searching and filtering facilities provided by the browser.

Table 3.1: BrownSauce summary.

Browser	Runs	Data Sources	Data Formats	Data Presentations	Presentation selection

BrownSauce	Local Web Server	One at a time	RDF (Any)	Indented text list	Hard-coded at compile-time

3.1.1 BrownSauce Evaluation

The browser capabilities of BrownSauce do not score highly when evaluated against the criteria established in the methodology. BrownSauce requires the installation of a local web server which does create potential ease of installation and use problems (--). BrownSauce displays data from only single semantic web document at a time and has no ability to aggregate data from multiple sources. The single data source can be specified at run time by a user (+). BrownSauce does have good support for RDF data formats (++). Brownsauce scores a total of + (1).

Brownsauce does not score well in the Data Presentation Criteria. There is no facility to adapt to either the ontology of the current subject or to the user. Since no data presentation alternatives are available there is no need for a user affordance to change data presentations.

3.2 Disco

The Disco Hyperdata Browser (Bizer & Gauß, 2007) is a simple browser for navigating the unbounded Semantic Web. Disco will take a Semantic Web URI and load the data found there. It will provide links to data linked within the Semantic Web document. Disco is described online at http://www4.wiwiss.fu-berlin.de/bizer/ng4j/disco/. The Disco version reviewed is current as at August 2010. A live version of the Disco browser is available here: http://www4.wiwiss.fu-berlin.de/rdf_browser/.

Disco is a web-server hosted application that can be accessed from any modern web-browser. It is written in Java. No code is run on the local web-browser client so it requires no installation of special software or changes to browser configuration.

Figure 3.3: Edited screenshot of Disco browser showing Tim Berners-Lee's FOAF file. (Edits: Cuts made show relevant features) (Author's own).

Figure 3.4: Edited screenshot of Disco browser showing Berlin from GeoNames (http://sws.geonames.org/2950159/). (Edits: Cut made show relevant features). (Author's own)

Disco displays data in a property value table. The third column of the property-value table gives a reference to the provenance (data source) of the triplet. RDF documents are added to the sources section as they are explored during a browsing session. Disco is able to incorporate data from multiple RDF documents at once. Disco can read most valid RDF formatted documents that conform to the Linked Data specification.

Disco will display properties as plain text, links or images as applicable. There are no other data presentations so therefore there is no affordance provided to allow users to change the presentation. The Disco homepage explicitly states that Disco is a lower-level semantic web browser suitable debugging and demonstrating linked data.

Table 3.2: Disco summary.

Browser	Runs	Data Sources	Data Formats	Data Presentations	Presentation selection
Disco	Web Server	Multiple Unbounded	RDF (Any)	Predicate-object table	Hard-coded at compile-time

3.2.1 Disco Evaluation

Disco scores well against the browser capability criteria. Disco runs locally in a web browser so no software installation is needed (++). Disco can aggregate the data from multiple sources into a single presentation and the data sources can be specified at runtime by the user (++). Disco also supports a wide variety of RDF file formats (++). Disco scores a total of ++++++ (6).

Disco scores poorly against the data presentation criteria. Disco does not change the data presentation dependent upon the ontology of the current subject. Disco does not use knowledge of the current user to adapt the data presentation. Disco has no alternative data presentations so there is no affordance to switch to an alternative.

3.3 Exhibit

Exhibit (Huynh et al., 2007) is a framework for creating interactive web pages based on semantic web documents. Exhibit is part of the Simile project and is available online at: http://www.simile-widgets.org/exhibit/. The Exhibit version reviewed is 2.2.0.

Exhibit is deployed in any modern HTML browser and uses standard technology (HTML, CSS and JavaScript) to work. Exhibit does not require the installation of special client-side software or browser extensions. Exhibit does not run any code on the web-server; all code is contained in client-side run JavaScript files that are downloaded from the web-server.

Figure 3.5: Exhibit showing keyword search, facets, timeline and map. (Screenshot of: http://www.simile-widgets.org/exhibit/examples/presidents/presidents.html, taken 26 Aug 2010)

Figure 3.6: Exhibit showing a timeline overview (left) and a detail view of a person (right). (Author's own)

An Exhibit based browser is a custom written piece of HTML code that uses extended XML attributes to instruct the Exhibit framework where to display widgets and the meanings of interface elements. The Exhibit framework allows easy creation of an interface to data that affords ways for users to search and browse the JSON document in ways useful to them.

Exhibit allows the browsing of a single data file that is available in JSON format. It does not natively aggregate data from many sources. JSON (JavaScript Object Notation) is a lightweight text file data format that is meant to be readable and writeable by both humans and machines. JSON can be used as an alternative to RDF for semantic web data. (For more information on JSON, see http://json.org)

Exhibit display widgets (called "views") can display data in List, Timeline, Graph, Map, Tabular and Custom HTML forms. The map view allows data to be overlaid on the map with a variety of different graphics to represent different data. Map data can also become hotspots that either display a bubble of HTML information on click, or go to a detail page. Timelines are scrollable and show data items as duration bars that are clickable to access details. Graphs also allow clickable detail hotspots in their plots.

An Exhibit based browser is actually a custom written database front-end application that uses semantic web technologies as an underlying data-store. All of Exhibit's display widgets are set at author-time, the data-source is set at author-time and facet's are set are author-time. While Exhibit is not strictly a generic semantic web browser it is included because it does show the possibilities for semantic web technology.

Table 3.3: Exhibit summary.

Browser	Runs	Data Sources	Data Formats	Data Presentations	Presentation selection
Exhibit	HTML Web Browser	Single author-time specified	JSON only	List, Timeline, Map, Graph, Table, Custom HTML form	HTML Author-time

3.3.1 Exhibit Evaluation

Exhibit does not score well in the browser capability criteria. An Exhibit browser is delivered by a web server and runs within a web browser. No special server-side support for scripting languages or databases is needed because it is driven totally by HTML, CSS and JavaScript in the local HTML web browser (++). Exhibit supports only a single data source that is hard coded at the author time; the time when the Exhibit-based site is created (--). Exhibit supports only the JSON data format (-). Exhibit scores a total of – (-1).

Exhibit does not score highly against the data presentation criteria. Exhibit has support for a fixed set of different data presentations based on the ontology of the current semantic web subject. However, the ontology based data presentations provided are hard coded into Exhibit at author time. Exhibit has does not adjust the data presentation based on knowledge on the current user. Affordances are generally provided in the user interface for manually switching between alternative data presentations.

3.4 Marbles

Marbles (Becker & Bizer, 2009) is a linked data semantic web browser that can aggregate data from many sources into a single display. The marbles homepage is available at: http://marbles.sourceforge.net/. The version reviewed is current as at August 2010.

Marbles is a web-server based application that formats semantic web data for consumption with a modern HTML web browser. Marbles is written in Java.

Marbles produces data presentations by transforming data through Fresnel Lenses and Formats. Fresnel (http://www.w3.org/2005/04/fresnel-info/) is an effort to provide a standardized language for representing the presentation of semantic web data. Fresnel Lenses specify the ordering of predicate-object pairs and Fresnel Formats specify how semantic web resources are visually presented.

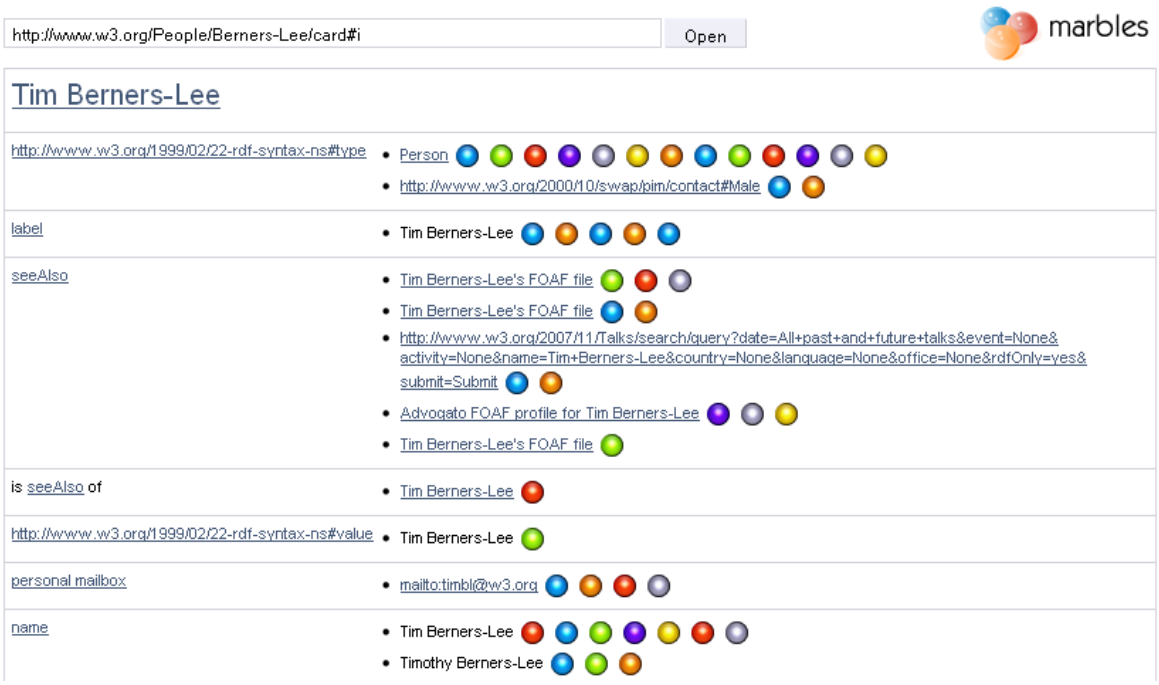

Figure 3.7: Marbles screenshot showing Tim Berners-Lee's FOAF file. (Author's own)

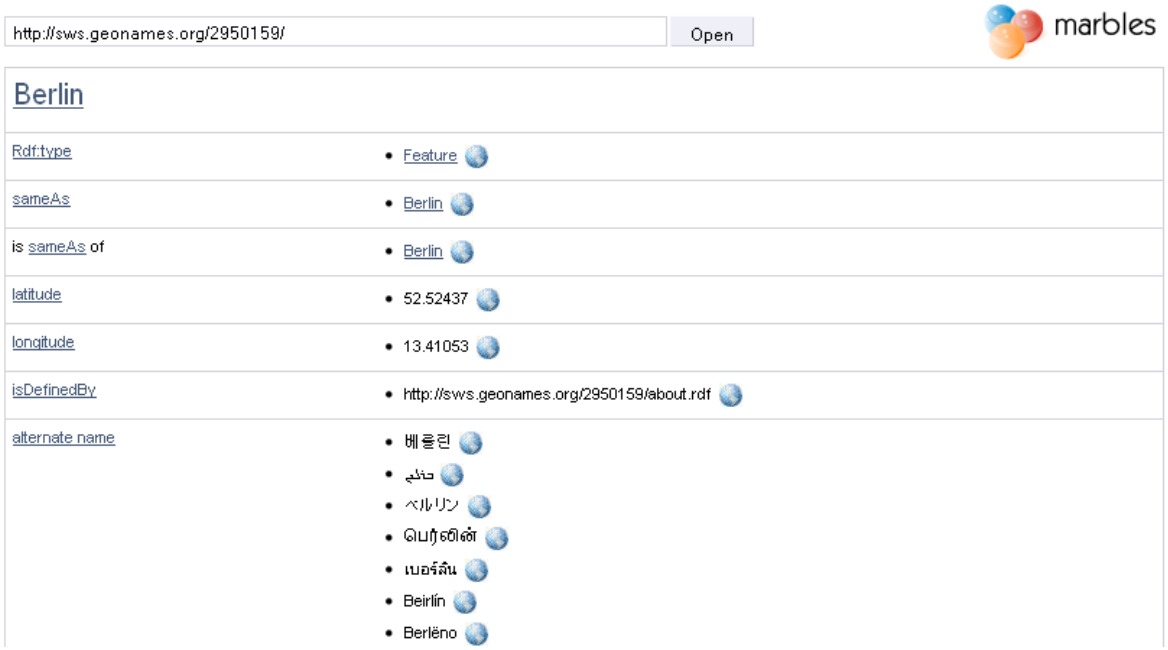

Figure 3.8: Marbles screen shot showing Berlin from GeoNames (http://sws.geonames.org/2950159/). (Author's own)

The top row of the Marbles display provides a text input box to submit a semantic web URI. Underneath that, the label for the current semantic web subject is displayed. Following this is a predicate-object table. The coloured circles to the right of the value labels relate to the source document of that predicate-object pair. Hovering the mouse cursor over a coloured circle with display the source's URI. The list of source documents, and their retrieval status, is listed at the end of the HTML page.

Marbles displays both text and images. Marbles automatically follows references to linked data in order to complete the text in the display. Both attributes and values are hyperlinked where applicable. Marbles will ensure semantic web data links redirect back to display in the Marbles browser, while links to HTML files are left to display in HTML mode.

The impact of the use of Fresnel was not apparent in the version reviewed. The display appeared generic and uncustomised. There does not appear to be the facility for multiple presentations.

Marbles does have switchable data presentations for full versions, photos only or mobile versions. The view is selected depending on the URI used to access the Marbles browser. No user interface affordances are given to switch the view.

Table 3.4: Marbles summary.

Browser	Runs	Data Sources	Data Formats	Data Presentations	Presentation selection
Marbles	Web Server	Multiple Unbounded	RDF (Any)	Predicate-object table	Fresnel.

3.4.1 Marbles Evaluation

Marbles scores very highly against the browser capability criteria. Marbles runs on a web server with all resources delivered to the local web browser as standard HTML/CSS/JavaScript (++). Marbles allows the user to specify a data source URL at run time and will aggregate the data from multiple data sources into a single data presentation (++). Marbles also has good support for RDF data formats (++).Marbles scores a total of ++++++ (6).

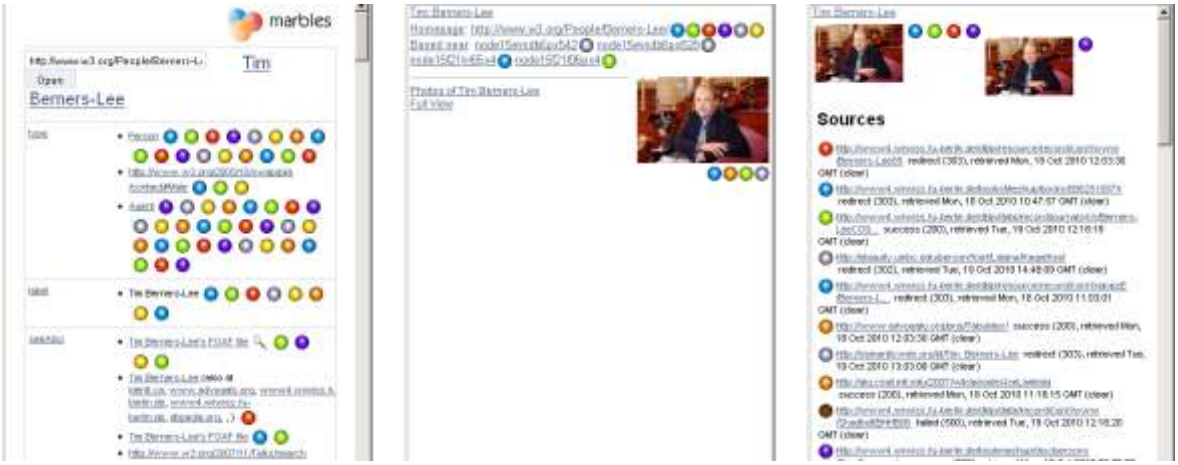

Figure 3.9: Marbles alternative data presentations (from left to right) Full view, Summary view, Photo view. (Author's own)

Marbles scores poorly against the data presentation criteria. Marbles does not change the data presentation in response to subjects expressed in different ontologies. Marbles has a selector to change the display type based on a user preference, but not a user type. The display types are full, summary and photo. The selection of data presentation is done manually by the user.

3.5 ObjectViewer

ObjectViewer (Lerner & Self, 2004) is a linked data browser that is capable of browsing the unbounded semantic web linked data cloud. ObjectViewer takes a given URI and loads the data contained there. ObjectViewer is available online from: http://projects.semwebcentral.org/projects/objectviewer/. The current version is 1.1.

ObjectViewer is a Java Swing application that runs all code on the desktop machine. It uses the Jena framework (http://jena.sourceforge.net/) to read and parse RDF data.

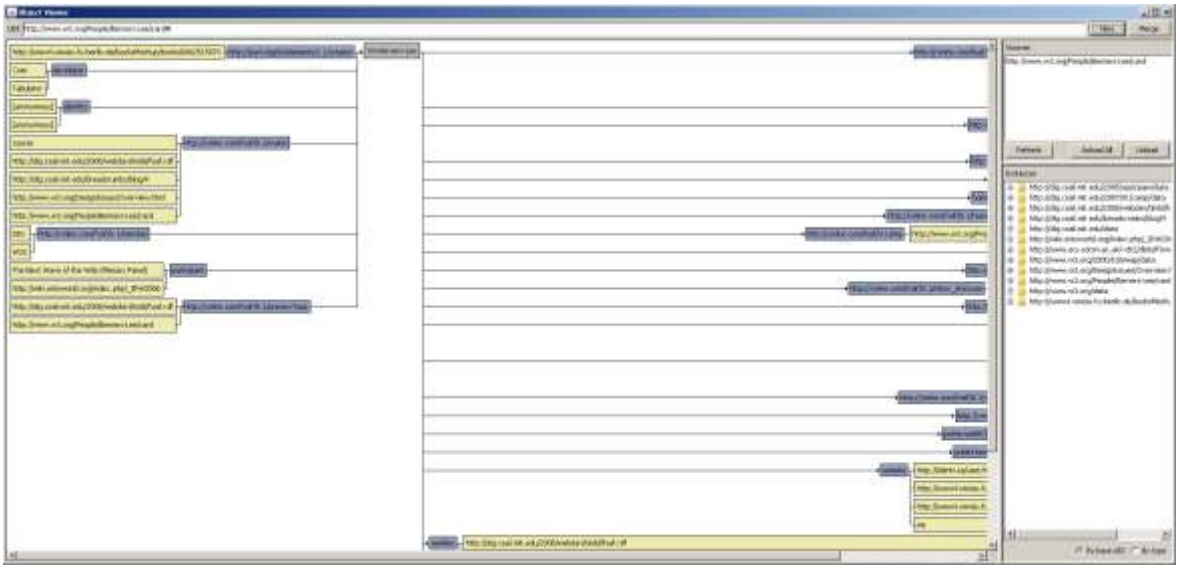

Figure 3.10: ObjectViewer window showing address bar, graph viewing area, sources list and instances list. (Author's own)

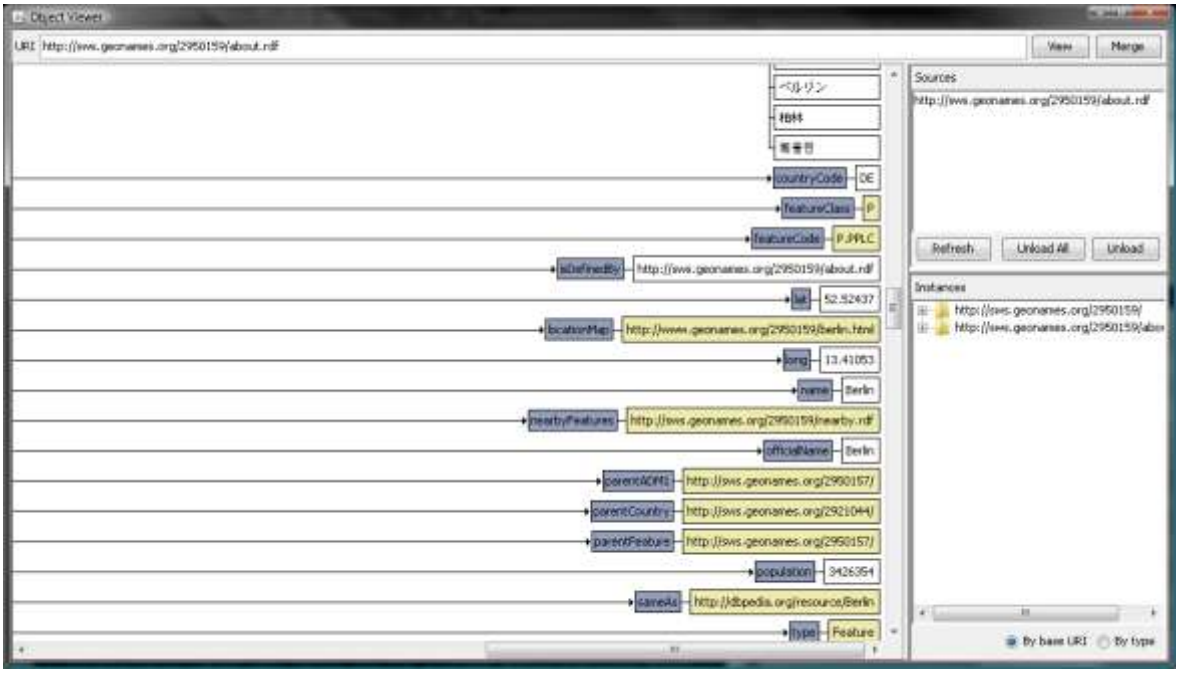

Figure 3.11: Object Viewer screen shot showing Berlin from GeoNames (http://sws.geonames.org/2950159/). (Author's own)

The ObjectViewer window has an address bar at the top where the user can input a linked data URI. The view button will clear the current graph and load the new document into it. The Merge button will load a new document and merge its contents into current graph. To the right of the window is a list of source RDF documents and a list of current instances discovered. The largest pane of the ObjectViewer window displays a graph starting from the current RDF node.

While ObjectViewer does work, there are many facilities missing that are common in standard HTML browsers that may also be of use in a semantic web browser. The ObjectViewer browser includes no status indicators to show that data is being transferred, no history facilities (though the Sources pane is of some utility) and no Back button. There is no bookmarking and no homepage facility. Printing does not exist and there is no facility to open a new window or tab for simultaneous browsing of multiple semantic web subjects.

ObjectViewer only displays RDF data in a graph form. The graph is interactive, with yellow labels acting as links to other graphs. The graph display works well for smaller data sets, but quickly becomes very large to scroll around when the RDF graph becomes more complex.

Table 3.5: ObjectViewer summary.

Browser	Runs	Data Sources	Data Formats	Data Presentations	Presentation selection
ObjectViewer	Desktop Java	Multiple Unbounded	RDF (Any supported by Jena)	Graph - interactive	Hard-coded at compile-time

3.5.1 ObjectViewer Evaluation

ObjectViewer scores highly against the browser capability criteria. The ObjectViewer semantic web browser runs as a desktop java application so it is cross platform but it is not the easiest software to install (+). ObjectViewer can aggregate data from multiple sources that are specified at run time (++). ObjectViewer also has good support for RDF data formats via its use of the Jena libraries (++). ObjectViewer scores +++++ (5).

ObjectViewer scores very poorly against the data presentation criteria. ObjectViewer does not change the data presentation in response to subjects expressed in different ontologies. There is also no change in the data presentation in response to user types. Since there are no alternative data presentation then there is also no need for a user interface affordance to switch between them.

3.6 The Tabulator Extension

Tabulator (Tabulator Team, 2008) is a high level browser for browsing the linked data cloud. When it comes across Semantic Web Data, Tabulator will format and display the data found there. The Tabulator homepage is: http://dig.csail.mit.edu/2007/tab/. It appears that in recent versions, the Tabulator Extension has become known as simply The Data Browser Extension. The version tested is 0.8.7 running in Firefox Portable 3.0.19 because Tabulator is not compatible with more recent browser versions.

Tabulator is implemented as a browser extension for Firefox browsers. It is therefore written in JavaScript, HTML, CSS and XUL. As a browser extension it runs completely on the local desktop and does not require server support. It works by registering itself as a handler for Semantic Web documents where the Tabulator extension takes over displaying semantic web documents.

The Tabulator display has the current semantic web subject at the top, with some display toggle options next to the subject header. Following this, the default display of data is to present a predicate-object table. Values that are links are either displayed as label if the label information is available or as a URI. Linked values are expandable in place using a small triangle icon to the left of the value's text. Between this triangle and the value's text label is a round coloured icon. Hovering the mouse cursor over the circle shaped icon will display the data's source URI with information on whether or not that data has been fetched. The colour of the circle represents the fetch status; Green for successfully fetched, blue for not yet fetched, red for a failed fetch and yellow for a fetch in progress. Data available from multiple sources will have multiple source icons, one per source. Clicking a source icon will (re)fetch the data. This means that Tabulator is able to aggregate the data from multiple sources into a single display.

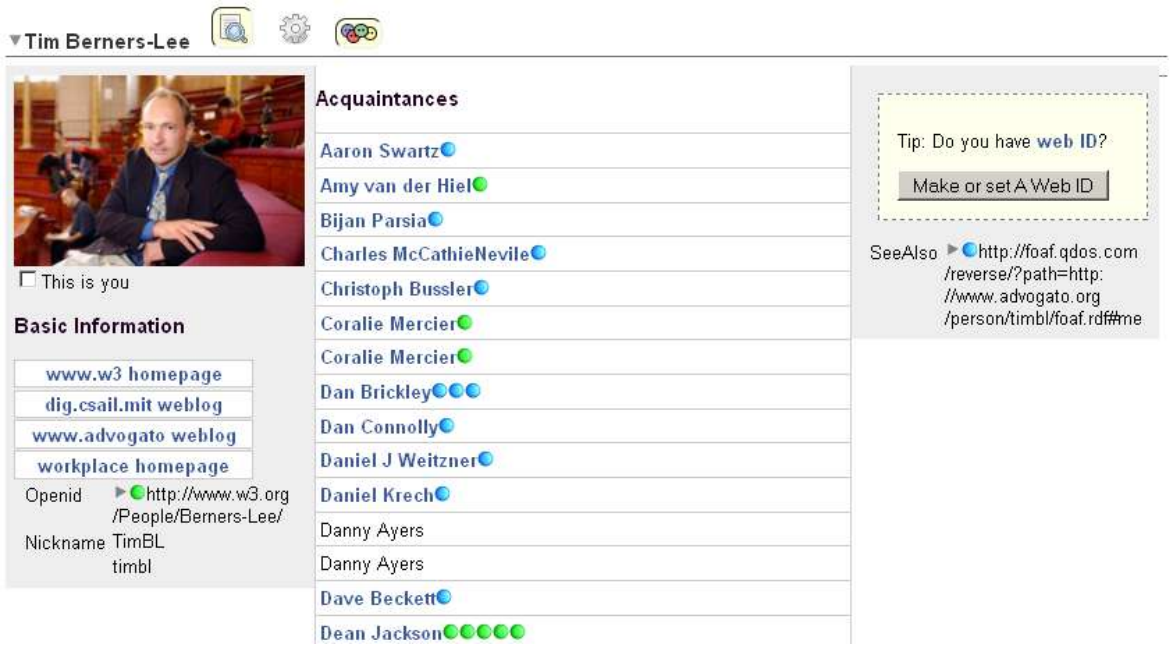

Figure 3.12: Tabulator Extension showing a Friend Of A Friend (FOAF) profile in Friends view. (Author's own)

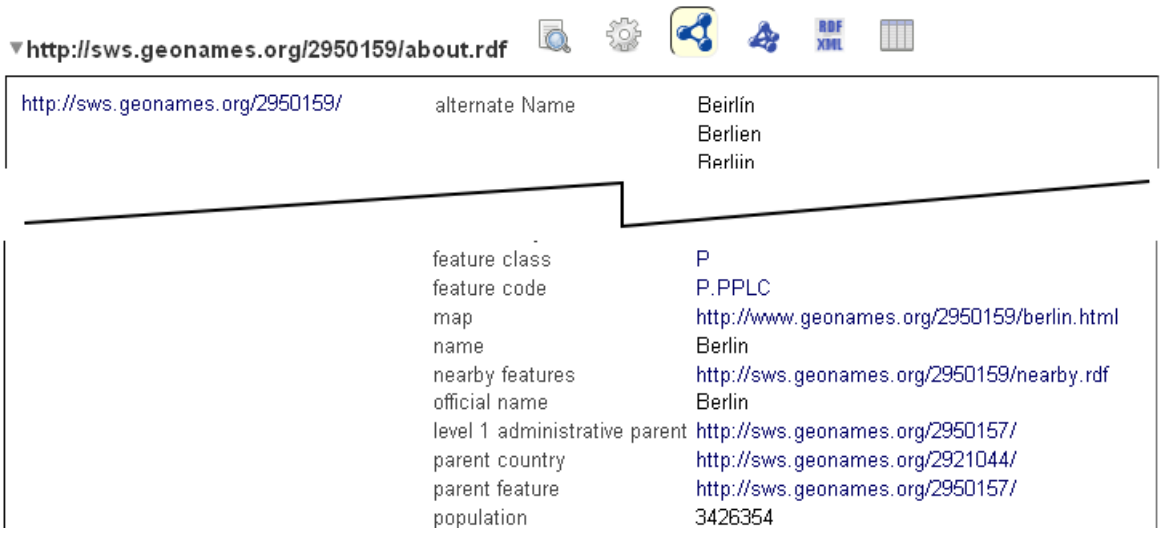

Figure 3.13: Editted Tabulator Extension screen shot showing Berlin from GeoNames (http://sws.geonames.org/2950159/about.rdf). (Edit: cut to show interesting data). (Author's own)

To the right of a subject's label are the display toggle items. Clicking these will either display additional information or show a more tailored view of the subject. Tailored data presentation icons are made available when Tabulator recognizes the data-type of the semantic web subject. The view will always default to the predicate-object table and the user must select the tailored view they want. The image above shows an icon for selecting a specialized data presentation because the current semantic web subject contains FOAF data.

Tabulator currently has support for table, map, friends, calendar and web page data presentations. Tabulator can also show the underlying data as RDF/N3, RDF/XML. Further view-types must be coded directly into the Tabulator extension.

Table 3.6: Tabulator summary.

Browser	Runs	Data Sources	Data Formats	Data Presentations	Presentation selection
Tabulator	Firefox Web browser extension	Multiple Unbounded	RDF (Any)	Table, Calendar, Map, Friends. RDF/N3, RDF/XML, HTML	Run-time manually by user. Available presentations are decided by data-type.

3.6.1 Tabulator Evaluation

Tabulator scores highly against the browser capability criteria. Tabulator runs as an extension in the Firefox web browser and all intelligence driving semantic web browsing is contained locally within the JavaScript of the extension (+). Tabulator is able to take run-time data sources URL provided by the user and aggregate these into a data presentation (++). Tabulator has good support for RDF data formats (++). Tabulator scores a total of +++++ (5).

Tabulator scores just above par against the data presentation criteria. Tabulator provides a fixed number of data presentations that cater for subjects expressed in key known ontologies. Tabulator does not change the data presentation based on knowledge of the user. Switching between alternative data presentations is done manually.

3.7 Zitgist DataViewer

The Zitgist DataViewer (OpenLink Software, 2009) is a high-level semantic web browser that can amalgamate data from many sources to display information from a single semantic web document. The DataViewer homepage is at: http://zitgist.com/products/dataviewer/dataviewer.html. The version reviewed is current as at August 2010.

DataViewer is a web server based product that renders HTML/CSS and JavaScript suitable for display in a reasonably modern HTML web browser. DataViewer uses a templating system that selects a presentation based on data-type. DataViewer will follow linked data hyperlinks in order to complete all text needed to render the web page.

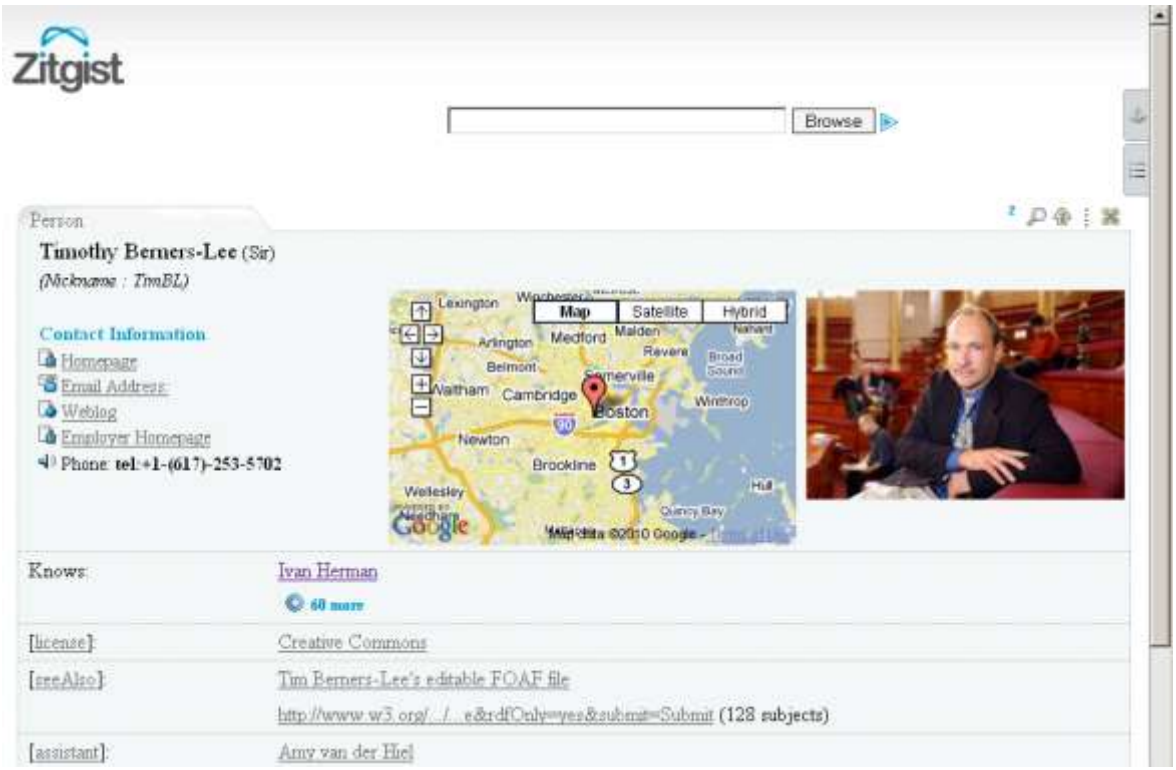

Figure 3.14: Zitgist DataViewer showing a Friend Of A Friend (FOAF) file. (Author's own)

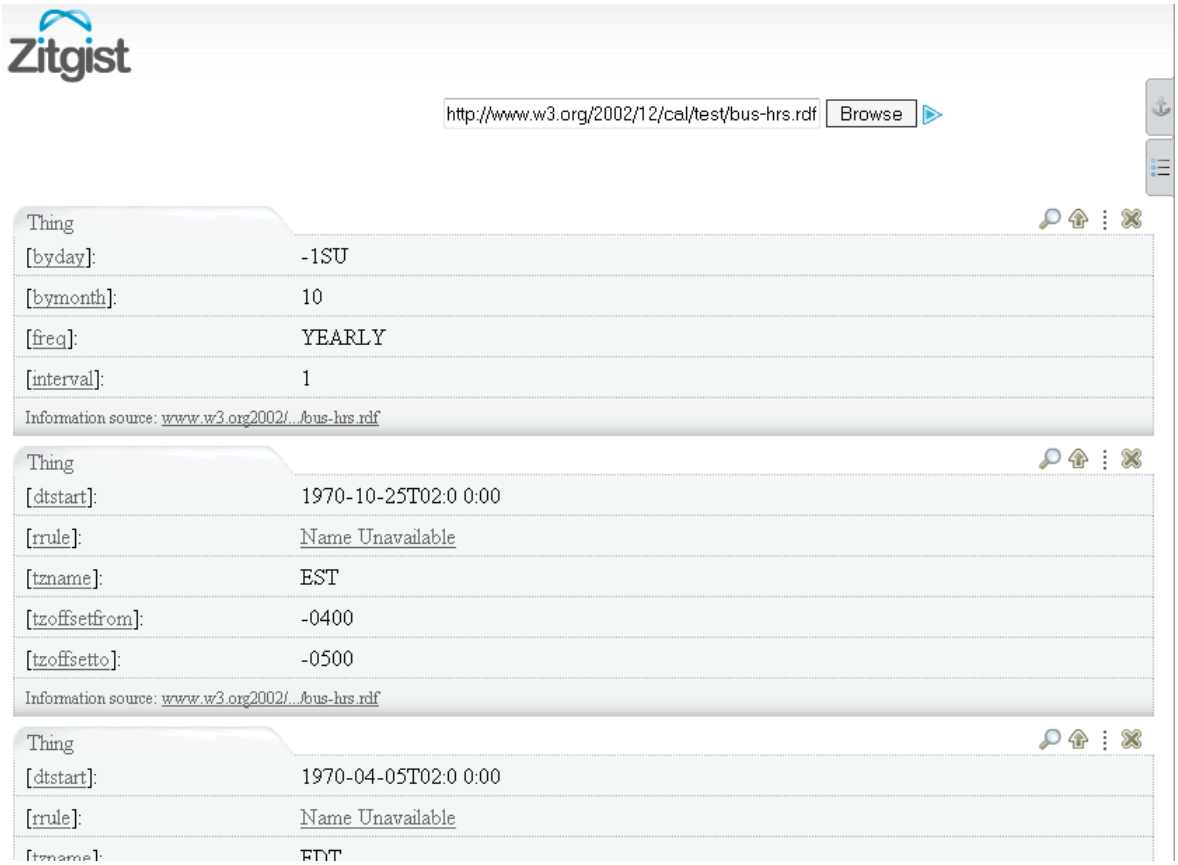

Figure 3.15: Zitgist DataViewer screen shot of a test calendar file (http://www.w3.org/2002/12/cal/test/bus-hrs.rdf). The calendar ontology is not recognized so a generic data presentation is shown. (Author's own)

At the top-centre of the DataViewer display is a text input box for submitting a new semantic web URI. On the right hand edge of the page is an anchor icon that activates the Navigator panel. The Navigator panel allows the user to select a subject type, then a specific instance of that subject from the current document. The “Bullets” tab below the anchor tab opens the Selector panel which allows the user to manually show or hide particular subject types and particular attributes.

Each subject contained within the semantic web document is displayed in a tabbed form. The top of the tab specifies the subject type and the label (where applicable) of the subject. To the right of the tab are controls to open a list of linked subjects, focus the browser on just this subject to the exclusion of others, to scroll to the top of the page and to show/hide the contents of the tab.

Templates are then used to display subjects according to their data type. DataViewer currently includes support for special templates covering the following ontologies: Music Ontology, Description of a Project, Friend Of A Friend (FOAF), Geonames and Semantically-Interlinked

Online Communities (SIOC). All other data is rendered using a default predicate-object table. For example, in the above screenshot, Zitgist DataViewer renders the start of the FOAF data in a business card style format and the location information in a Google map widget.

Table 3.7: Zitgist DataViewer summary.

Browser	Runs	Data Sources	Data Formats	Data Presentations	Presentation selection
Zitgist DataViewer	Web Server	Multiple Unbounded	RDF (Any)	Templates, Predicate-object table	Automatically selected by data-type.

3.7.1 Evaluation of Zitgist DataViewer

The Zitgist DataViewer semantic web browser scores highly against the browser capability criteria. Zitgist DataViewer runs on a web server and the user accesses it via a standard HTML web browser (++). Zitgist DataViewer is able to aggregate data from multiple sources that are supplied at runtime (++). It also has good support for RDF data formats (++). Zitgist DataViewer scores a total of ++++++ (6).

Zitgist DataViewer scores the best against the data presentation criteria. Zitgist is able to provide alternative data presentation dependent upon the ontology of the current subject. The set of data presentation alternatives is extensible via a template system. Zitgist does not change the data presentation based on knowledge of a user. ZitGist DataViewer automatically switches to the most appropriate data presentation alternative and therefore does not require manual intervention by the user.

4 Findings

The Findings section compares and contrasts information from the Results section into a form that makes it easier to draw conclusions from. Firstly a table of the key attributes is presented, followed by evaluation tables and then further commentary is made.

Table 4.1: Comparison of Semantic Web Browsers

Browser	Runs	Data Sources	Data Formats	Data Presentations	Presentation selection

BrownSauce	Local Web Server	One at a time	RDF (Any)	Indented text list	Hard-coded at compile-time
Disco	Web Server	Multiple Unbounded	RDF (Any)	Predicate-object table	Hard-coded at compile-time
Exhibit	HTML Web Browser	Single author-time specified	JSON only	List, Timeline, Map, Graph, Table, Custom HTML form	HTML Author-time
Marbles	Web Server	Multiple Unbounded	RDF (Any)	Predicate-object table	Fresnel.
ObjectViewer	Desktop Java	Multiple Unbounded	RDF (Any supported by Jena)	Graph - interactive	Hard-coded at compile-time
Tabulator	Firefox Web browser extension	Multiple Unbounded	RDF (Any)	Table, Calendar, Map, Friends. RDF/N3, RDF/XML, HTML	Run-time manually by user. Available presentations are decided by data-type.
Zitgist DataViewer	Web Server	Multiple Unbounded	RDF (Any)	Templates, Predicate-object table	Automatically selected by data-type.

Exhibit, Marbles and Tabulator do allow filtering and ordering of which attribute value pairs are displayed – Exhibit at author-time of the configuration HTML file, Marbles via Fresnel and Tabulator by the use of different data presentations. Zitgist DataViewer allows the user to manually filter which attributes are displayed. None of the other reviewed semantic web browsers provides the facility to filter or order the display of predicate-object pairs. The default ordering of the reviewed browsers was to either present data alphabetically, or in the order specified in the source RDF document.

4.1 Evaluation

The following table merges the evaluations of the reviewed semantic web browsers into a single place for ease of comparison. Each of the two criteria groups are presented separately.

4.1.1 Browser Capability criteria

Table 4.2: Evaluation of semantic web browsers against the Browser Capability criteria

Browser	Browser Capability Criteria			Total
	Ease of Use	Supported Data Sources	Supported Data Formats	
BrownSauce	--	+	++	+ (1)
Disco	++	++	++	++++++ (6)
Exhibit	++	--	-	- (-1)
Marbles	++	++	++	++++++ (6)
ObjectViewer	+	++	++	+++++ (5)
Tabulator	+	++	++	+++++ (5)
Zitgist DataViewer	++	++	++	++++++ (6)

The semantic web browsers scoring the highest (6) against the browser capability criteria were Disco, Marbles and Zitgist DataViewer. These three semantic web browsers we easy to use because they ran in a standard HTML web browser. They all supported multiple arbitrary data sources supplied at runtime and the support for semantic web standard data formats was good.

Tabulator and ObjectViewer scored 5. Both Tabulator and ObjectViewer also supported multiple arbitrary data sources supplied at runtime and had good support for standard semantic web data formats. However, both Tabulator and ObjectViewer required client-side installation of software.

Brownsauce scored 1 because it supported only a single data source at a time and required the client-side installation of server software. Exhibit scored the lowest (-1) because it supported only a single data source hardcoded at author time and only supported the JSON file format. However, Exhibit did score highly for ease of use since it runs in a standard HTML web browser.

4.1.2 Data Presentation criteria

Table 4.3: Comparison of semantic web browsers against the Data Presentation criteria. An 'x' marks where the browser places.

	Ontology Adaptation				User Adaption			UI Affordance		
	None	Fixed	Extensible	Innumerable	None	Fixed	Open	None	Manual	Automatic
Brownsauce	x				x			x		
Disco	x				x			x		
Exhibit		x			x				x	
Marbles			x		x	*			x	
Openviewer	x				x			x		
Tabulator		x			x				x	
Zitgist Dataviewer			x		x					x

* Marbles does have a fixed selector for choosing between full, summary and photo views – but this is not considered adapting to different fixed user groups.

None of the semantic web browsers did well against the data presentation criteria. None of the candidates adapted to innumerable ontologies. No browsers took into account the individual user – although Marbles potentially could have provided manual facilities to change between fixed user types. Only Zitgist Dataviewer automatically changed to an alternative data presentation.

5 Summary and Conclusions

In this section significant findings are summarised by evaluating against the questions established in the Introduction section. Then each hypothesis is directly answered in turn to provide conclusions to the research. This is followed by additional discoveries, extra to those that directly inform the hypotheses.

What is the current state of the art in semantic web browser capability?

Current semantic web browsers have demonstrated workable solutions that transfer (via HTTP) semantic web documents and parse them into data for later display. Five out of seven of the reviewed semantic web browsers scored highly against the browser capability criteria. This suggests that there is enough existing work in underlying infrastructure so that this research can focus on presentation issues only.

There is a wide interpretation of ideas on where the actual code running the semantic web-browser should reside. The server-based solutions tended to use traditional HTML web browsers as thin clients to display semantic web renderings. Of the client-side solutions, Tabulator runs as an extension within the FireFox HTML web browser. ObjectViewer uses a standalone Java Swing application. While each deployment method has advantages and disadvantages that may better suit them for particular circumstances, this paper has taken the assumption that a system that provides the easiest setup and lowest maintenance is better.

How do current semantic web browsers display data?

BrownSauce displays data in an indented text list. Disco displays data in a predicate-object table. Exhibit displays data in timeline, map, graph, tabular and custom HTML forms. ObjectViewer displays data as a graph. Tabulator displays data in many different data presentations: predicate-object table, map, calendar and friends. Tabulator can also display the underlying RDF or HTML.

There is limited facility for run-time filtering and ordering in the reviewed semantic web browsers. Fresnel does provide a vocabulary for filtering and ordering, but Fresnel is not yet widely implemented. Zitgist DataViewer uses a template system that filters and orders data. Zitgist DataViewer provides the Selector panel that allows the user to manually filter attributes at run-time. However, subjects with many predicate-object pairs will have long displays which make it difficult to locate specific data of interest. Also, ordering of predicate-objects could potentially put the most important information first. This is a gap in the current body of work that this research can explore.

Do current semantic web browsers adjust the data presentation based on ontology?

Over half of the current semantic web browsers adjusted the data presentation based on the ontology of the current subject. In all cases this was a single mapping of data presentation to a fixed set of ontologies. In two cases the set of data presentations could be extended via

templates. No current semantic web browser was able to adjust the data presentation for an innumerable number of ontologies.

Do current semantic web browsers adjust the data presentation based on the user?
No current semantic web browser takes the individual user into consideration when building a data presentation.

What user interface affordances are available to change to an alternative data presentation?
For the semantic web browsers that had alternative data presentations most provided a user interface affordance that had to be manually operated to change to the alternative data presentation. Only one of the current semantic web browsers automatically used an alternative data presentation automatically and without user intervention.

5.1 Summary of Conclusions

1. Many of the semantic web browsers reviewed meet the browser capability criteria which indicates that there is a strong technology base that further research can be built upon.
2. The reviewed semantic web browsers reviewed performed poorly against the data presentation criteria which indicates a research gap that can be further explored.
    a. Adaption of the data presentation based on ontology was limited to a fixed set or extensible via templates and did not provide facilities for adapting to the innumerable ontologies found on the semantic web. Further research could explore adaption to innumerable ontologies.
    b. No semantic web browser took the individual user's data presentation needs into consideration. There is an opportunity for new research into selecting presentations based on user needs, rather than just data type.
    c. Only one semantic web browser automatically attempted to select the best data presentation from the available alternatives. Further research could explore the automatic display of the best available data presentation.

Additional Conclusions

During the course of the research, additional things were discovered that while not directly related to the hypotheses of the research are of potential use for further research.

- There is room for significant improvement in the variety of data presentations possible.
- A semantic web browser could be deployed server-side with an HTML web-browser client, desktop application or hosted in an HTML web browser.
- There is an opportunity for new research into filtering and ordering presentations of predicate-objects in semantic web data - perhaps built upon the Fresnel Lenses language.
- Current view types are hard-coded by humans. There exists a new research opportunity in exploring computer generation of presentations – perhaps in Fresnel Formats language.

Currently semantic web browsers do provide a suitable and mature enough foundation for building further research upon. It is clear that the current state of the art in semantic web browsers do not adapt to present data from an innumerable corpus in meaningful ways. Therefore a gap exists in the current research that that future research can build upon.

References

de Araújo, S. F., & Schwabe, D. (2009). Explorator: a tool for exploring RDF data through direct manipulation. In *Proceedings of the WWW2009 Workshop on Linked Data on the Web (LDOW2009)*. Madrid, Spain. Retrieved from http://ceur-ws.org/Vol-538/ldow2009_paper2.pdf

Becker, C., & Bizer, C. (2009). Marbles linked data browser. *Marbles*. Retrieved November 23, 2010, from http://marbles.sourceforge.net/

Bizer, C., & Gauß, T. (2007, January 15). Disco - Hyperdata Browser. Retrieved November 23, 2010, from http://www4.wiwiss.fu-berlin.de/bizer/ng4j/disco/

Dvorak, M. (2008). MindRaider - Semantic Web Outliner. Retrieved November 23, 2010, from http://mindraider.sourceforge.net

Hildebrand, M., van Ossenbruggen, J., & Hardman, L. (2006). /facet: A browser for heterogeneous semantic web repositories. *The Semantic Web-ISWC 2006*, Lecture Notes in Computer Science, *4273*, 272–285. doi:10.1007/11926078_20

Huynh, D. F., Karger, D. R., & Miller, R. C. (2007). Exhibit: lightweight structured data publishing. In *Proceedings of the 16th international conference on World Wide Web - WWW '07* (pp. 737–746). Banff, Alberta, Canada: ACM. doi:10.1145/1242572.1242672

Lerner, J., & Self, T. (2004). Object Viewer. *SemWebCentral: Object Viewer: Project Info*. Retrieved November 23, 2010, from http://projects.semwebcentral.org/projects/objectviewer/

MIT. (2005). SIMILIE: Longwell RDF Browser. Retrieved November 23, 2010, from http://simile.mit.edu/wiki/Longwell

OpenLink Software. (2006). OpenLink Data Explorer. Retrieved November 23, 2010, from http://linkeddata.uriburner.com/ode/

OpenLink Software. (2009). Zitgist DataViewer. Retrieved November 23, 2010, from http://zitgist.com/products/dataviewer/dataviewer.html

Oren, E., Delbru, R., & Decker, S. (2006). Extending Faceted Navigation for RDF Data. (I. Cruz, S. Decker, D. Allemang, C. Preist, D. Schwabe, P. Mika, M. Uschold, et al., Eds.)*The Semantic Web - ISWC 2006*, Lecture Notes in Computer Science, *4273*, 559-572. doi:10.1007/11926078_40

Rutledge, L., van Ossenbruggen, J., & Hardman, L. (2005). Making RDF presentable: integrated global and local semantic Web browsing. In *Proceedings of the 14th international conference on World Wide Web* (pp. 199–206). Presented at the WWW05, Chiba, Japan: ACM. doi:10.1145/1060745.1060777

Steer, D. (2003, January 28). BrownSauce: an introduction. HP Laboratory. Retrieved from http://www.hpl.hp.com/techreports/2003/HPL-2003-10.pdf

Tabulator Team. (2008, November 24). The Tabulator Extension. *The Tabulator Extension*. Retrieved November 23, 2010, from http://dig.csail.mit.edu/2007/tab/

W3C. (2007, October 21). IsaViz: A Visual Authoring Tool for RDF. *IsaViz Overview*. Retrieved November 23, 2010, from http://www.w3.org/2001/11/IsaViz/