Semantic Announcement Sharing

Valentin Zacharias, Mike Sibler
Forschungszentrum Informatik FZI
D-76131, Karlsruhe, Germany
{zach,sibler}@fzi.de

Abstract

The paper stems from the idea that maybe the painstainkingly slow adoption of the Semantic Web into the mainstream www can be accelerated by taking clues from these tiny Semantic Weblets already present today.
We have identified RSS as one particularly successful Semantic Weblet, formed an opinion on why it was successful and have than tried to include all its success factors into a new Semantic Web application.

1 Introduction

We start by giving a short explanation of what we understand as Semantic Web and Semantic Weblet. We then describe RSS and its history in section two, in section three we try to explain why it is successful. Section four describes a Semantic Weblet that we hope could repeat the success of RSS. Section five is a more detailed lock at the protocol we are proposing for the use within this Semantic Weblet and after a very brief description of a server side editor for this protocol this text concludes with a future work section. Finally Appendix A gives one example for the appliance of the proposed protocol.

1.1 Semantic Web

For us the idea of the Semantic Web is nicely described in the Semantic Web Activity Statement of the W3C[HP,1999]:
"The Semantic Web is a vision: the idea of having data on the web defined and linked in such a way that it can be used by machines not just for display purposes, but for automation, integration and reuse of data across various applications".
Or even shorter in our own words: The Semantic Web aims to make the Internet more user friendly by making it easier to understand for machines.

1.2 Semantic Weblets

Semantic Weblets are small communities of sites that offer information that is better usable to computers than plain html plus the odd keyword now and then in mainstream www sites. They give computer understandable information about things like additions to the site, real live locations of blogs or social networks. There are quite a few of these, like BlogChalking[BlogChalking,2004], Xhtml Friends Network[XH,2004], GeoURL[GeoUrl,2004], ATOM[Atom,2004], FOAF[FOAF,2004], Creative Commons Metadata[CC,2004] or RSS; though most of them aren't used wildly and their future prospects are unclear.
Although some of these are going to perish, we believe that there will be more of such initiatives, that some will thrive and gain users and that, should there ever by "The Semantic Web", it will have grown from- and still consist of these specialized weblets.

2 RSS

RSS or "Really Simple Syndication" is a family of protocols for describing the content of a webpage in a way that allows for simple, automated syndication.

2.1 Architecture

At the center of the RSS weblet are sites that have an interest in their content being syndicated. These sites than post a file in a specific format - an RSS file - on their webpage. In this file there is a definition of one or more channels, where each channel has some metadata, like a name or the URL of an image. Each channel contains some items and their description, consisting at least of an title and a link. A frequent use case for RSS would be a RSS file on a Blog page, containing one channel having the same name as the Blog with the items being the titles of and links to the Blog entries of maybe the last ten days.
If such an RSS file is discovered by a RSS aware application, it "understands" which articles are advertised by this site, knows a bit about the articles and has some information about who's is advertising this information (via the channel description). Also some RSS formats include dates and Global Unique Identifiers, allowing a visiting application to filter out older items and items published to more than one channel.
There are two different use cases, although they are similar from a technical point of view: one is the syndicating user and the other the syndicating website. In the first use case somebody interested in a couple of websites, but tired of looking at all of them every day to find that most of them have no updates, gets a News Reader that, after the Urls of the interesting websites RSS files have been entered, checks them automatically and presents the user with a list of the new articles from all pages. Examples for RSS News Reader applications are Sharp Reader[SharpReader,2004], Straw [Straw,2004] or Bloglines [BlogLines,2004]. In the second case a web site aggregates the items from different channels, maybe processes the items in some

way and displays them to somebody visiting the site. Such a site could offer a page showing the newest articles from channels usually posting about Africa or it could use Machine Learning techniques to find trends in the Blog community. Google News[GoogleNews,2004] would be a prime example for such a site, but they seem to rely mostly on Information Extraction techniques. RootBlog [RootBlog,2004] and BLOGG.de [BLOGG.de,2004] are genuine RSS syndicating sites.
Since this point is often misunderstood: please note that RSS is traditionally a "pull" architecture. After somebody posts a new article to her website, she just updates her RSS file, nothing else is done. The syndicating programs read this RSS file regularly and will detect the change in due time. Recently however some RPC standards have emerged surrounding RSS, one of them being a "ping" to alert a syndicating site of a change to a RSS file.

2.2 History

RSS was invented in 1999 as RDF Site Summary 0.9 by the My Netscape Network team as a way to get more content for their portal. It was a simple RDF based protocol that allowed for the definition of channels and channel items, consisting of just a title and a link. Within a matter of months it was superseded by RSS 0.91, now called Rich Site Summary. As the name change suggests, it was no longer a RDF standard, but now based on XML, also it allowed for more elements like a description element, giving a natural language description of an item. Since then there are two lines of protocols, one being the XML line, starting with 0.91, continued with 0.92 and 0.92 now at 2.0. The other development line continues the idea of using RDF and the current version is 1.0. True to the idea of RDF this line of protocols allows for the inclusion of elements from other name spaces, such as the frequently used Dublic Core[DublinCore,2004] elements.

2.3 RSS's Success

Taking the statistics of one particularly large RSS syndicating site[Syndic8,2004], we can see that RSS started out pretty slow, with only 2500 RSS feeds in September 2001 but has grown to almost 120000 feeds in May 2004; this makes RSS easily the largest and most successful of the above mentioned Semantic Weblets.
Recently however RSS started to suffer from it's success: there are simply too many RSS channels around, even reading all items from the channels that sometimes carry interesting news becomes infeasible. There are discussions about using more sophisticated search technology or categorizations within the RSS file to alleviate this problem.

3 Success Factors

We identified six major factor for RSS's Success, these are:

Simplicity
Web friendliness
Openness
Plurality sources
Motivated sources
Plurality of syndicators

We will examine each of these in the following paragraphs.

3.1 Simplicity

RSS is a very simple format. A computer scientist familiar with XML or even an amateur used to HTML can read an RSS file with a text editor, understand what it means and could even change it to reflect the information he wants to publish.
Also, since this protocols are build on top of XML or RDF respectively, editors and API's are available, sometimes even at no cost. This made it easy for programmers around the world to make and share programs around RSS; and not just news readers for the syndicating side, but also web based editors for RSS or RSS support already incorporated into the Blog Software [MovableTYpe,2004]. With this software available it then became even easier to create and update your own RSS files, possible even for people not familiar with HTML.

3.2 Web Friendliness

An RSS file is just like a HTML file on your server, if your webspace allows for HTML files it will allow for RSS files. You can transfer it to the server using your cherished FTP application or even edit it on the server using WebDAV. You can control access to it, using the chmod commands or .htaccess files in the same way you've done it for years.
If you use a newsreader application it will use HTTP to contact the server on port 80 and retrieve a file – something you can do even if you are behind a strict firewall.
Finally, even creating RSS files from an application can be done using the same server side scripting techniques you already use, be that PHP, Perl, Python or Java Servlets.
In short: RSS is only a slight evolutionary addition to the current web, it easily fits into the current architecture of internet applications and allows programmers to reuse a great deal of their past experiences and existing libraries.

3.3 Openness

Everybody can use RSS in each and every way she pleases – as already stated in section 3.1 this was one of the necessary preconditions for a Open Source community to evolve around RSS.
Although it should not be concealed that this also lead to the division of the RSS protocol into the RDF and XML lines.

3.4 Plurality of Sources

As is evident from the previously cited number of 120000 different RSS feds there is a large number of different sources and its importance for the success of RSS should be evident: after all, what good is syndication if there are not many sources to syndicate?
But plurality here means more than just pure numbers, it also means that there are not 119997 unimportant RSS feds and 3 big ones – since in such a case the big ones would try to get the users to visit them exclusively and would therefore not support such a system.

In the RSS Semantic Weblet there is also a plurality of the messages involved: the news posted on the different feds are not all the same, on the contrary: the content is often unique- even if its only a diary entry - and chances are, if you hadn't read this particular Blog, you would have never heard of it.

3.5 Motivated Sources

Even if it is very simply to create an RSS file, there is still some effort involved and the content providers need some reason to undertake it. In the case of RSS the motivation stems from the wish to have more visitors – in order to increase revenue or to better spread ones message. It helps that the RSS files often contain only a small portion of the actual news, so an user will still visit the source website and look at the ads placed there.

3.6 Plurality of syndicators

If there is only a small number of syndicators, than they will try to get the syndicated information exclusively in order to lock out potential competitors. This may be achieved by forcing content providers to enter the information directly into the syndicators website or by relying on Information Extraction algorithms instead of data exchange protocols. This last approach makes it harder for potential competitors to offer the same service by forcing them to first program and optimize the extraction algorithms, significantly raising the amount of money they have to spend up front in order to compete.

4 Sharing real life events

Starting from RSS we wanted to create a Semantic Web application that, replicating the success factors of RSS, has the potential to become as successful. This of course does neither mean, that RSS shows the only possible road to a successful Semantic Web application nor that our idea will become an automatic success (chances are, it will not; since in the absent of funding for this project we lack the resources to develop release grade software or advertise it).

4.1 Identifying the domain

Three of the identified six success factors for RSS depend on the domain it which it is used, therefore in order to build something similar we had to find another domain for which these criteria are satisfied.
We decided to use the domain of announcements for events. Whenever there is some kind of event planned, a Rolling Stones concert of just a local market, the organizers will want to advertise it as broadly as possible with the least amount of money possible. Naturally they will turn to the internet to announce it. For smaller events and for somebody not living in the close vicinity or not following the local media, the internet may even be the only way to learn about such an event.
Seeing that today most events are announced on smaller sites, like the site of a local hunting club, we believe that we can speak of plurality of sources in the above described sense. The motivation of the sources is also not in doubt: the organizers want more visitors for their events, everyone syndicating their information may help them archive that. That leaves us with "Plurality of sources" as the last criteria for the domain, but if we understand that events even today are not just syndicated by large specialized event sites (that usually sell tickets), but also by smaller non profit local event sites, tourism sites and sites with general local information we see that this criteria is satisfied as well. We also believe that an RSS style protocol for events would lead to the equivalent of news readers for events, a personal event scout. People would feed it with the location of the syndication files of some of their favorite acts, some filtering criteria (within 100km of my location) and it would alert them to newly published events that met the criteria.

4.2 Architecture Overview

For the overall architecture we decided to stick very close to the example of RSS. Like in RSS we have the event data stored in files on the server, these files will be accessed via HTTP and can be uploaded vie FTP or can be created on the server using well known server side scripting / programming methods. We also decided to keep the pull architecture, forcing the syndicating sites to regularly visit the pages to detect changes themselves.

4.3 Protocol Overview

We choose RDF and RDFS as basis for our protocol, because of their relative simplicity, the availability of quite a few (free) tools that support them and our believe that RDF/RDFS can be used to create an effective, language independent and easily extendible categorization for events.
As in RSS our event files are organized in channels and some information about the channels is given. The channels contain events and each event attributes like title and description and consists of the main parts: ülace, category, time and purchase details. There are possibilities to model recurrent events (for instance weekly market) and complex events containing other events (for instance the Tour de France); but even though this is still very simple we made sure that somebody who only wants to model one simple event need not worry about those.

4.4 Software Support

The simplicity of our protocol in the sense described in 4.1 can not be completely assessed without a look at the tools supporting it. We identified four areas where tool support could help this protocol spread.

API

The basis for any other tools is an easy to use, easily available API for RDF. Luckily such API's are readily available in virtually every language, like Java[Kaon, 2002][Jena, 2000], PHP[RAP, 2004], Python [Pyrple, 2004] and soon maybe even for Smalltalk [Sambuca, 2004]. Most of the APIs are available for free.

Server Side Editor

A server side editor would be installed on the website of one of the syndicators. A user could enter the URL of her current event describing file that would then be retrieved by the editor, afterwards she could use the editor to change the data contained in the file and once she's finished changing it, she could download the new

file to her computer and than use ftp to put it on her website. A slightly better version of this editor would be a script running on the website of the event announcement source, bypassing the download/upload step. But not every webspace allows for custom CGI's and for somebody who's only announcing new events sporadically the time saving may not be worth the hassle of installing such an editor.
Such editors are of course not yet available, but we have created a prototype server side editor, showing both the feasibility and simplicity of building one.

Email Syndicator

An email syndicator is one example of the above described "Personal Event Scout". Users would enter their email address and a description of the events their interested in into such a website and would henceforth get an Email whenever a new event matching their criteria is announced.
Such a service could be run be anyone with an interest in including some commercials in the emails – naturally that would be the specialized event sites offering links to purchase tickets for the events deemed interesting (unless of course, they are free).
With a services like that, it would be even easier for people to syndicate the data from their favorite event sources, allowing them to avoid the hassle of installing some new software on their computer.

Website Event Aggregator

Some scripts that can be easily installed on a website that regularly check some event announcement files, aggregate this information and serve it as html in a customizable way. Such tools are necessary when we have a plurality of syndicators, since many of them will probably not have the resources to build their own tools for syndications.

4.5 Revisiting Success Factor

After having described a high level view of the Event Announcement Sharing Semantic Weblet we now want to revisit the success factors we identified in section three, to verify that our application meets these criteria.
We have already shown in section 4.1 that we have plurality and motivation of sources and plurality of syndicators in the event announcement domain.
We can take Openness for granted, since RDF is a W3C standard, most of the APIs for RDF are free and Open Source and our event announcement protocol is of course available for everyone.
As stated already for RSS, we believe that the kind of architecture described above is "web friendly".
That leaves us with "Simplicity" as last criteria. We believe that our protocol is, albeit a bit more complex than RSS, still sufficiently simple to be immediately understandable for thousands of programmers[1] around the world; together with the readily available RDF APIs this means that there is already a large base of programmers that could program the software for this Semantic Weblet. We have to concede however that not a single instance exists for three of the above mentioned four tool categories that could support the spread of this protocol. This would probably make it too difficult for not-technology savvy people, or syndication sites with few resources to join this Semantic Weblet, hampering it's grow. Still, from our experience in programming a prototype of one of these tools, we believe that no large investment is necessary to create these.

[1] The interested reader can learn more about the protocol in Section 5.

5 The protocol in detail

The simplest kinds of events happen on one location at one time, like the 100th birthday of a hunting club celebrated in the club house. We expect the majority of events to be in this category.
Besides this kind of events there are events that happen not just in one location (for instance the Tour de France), that happen at regular intervals (for instance a weekly market) or events composed from other events (for instance a congress, consisting of different workshops).
All of the above mentioned kinds of event share a number of simple attributes, these are: title, description and a link. These attributes are exact replica of the attributes in RSS 0.9, but for a description of events that should allow searching by location, time and category they are obviously not enough. We have therefore the following other attributes: purchase details, location, category and time frame. We will now examine each of these in more detail.

Purchase details

Purchase details consist of entranceFee[2], purchaseLink and a Boolean "public" – allowing to state that some kind of restriction applies to who may attend (for instance a medical congress may not be open for the public).

Location

To understand how the location is modeled the reader has to note that as already stated the location of some (especially larger) events may not be easily stated, for instance the "Tour de France" is in France and the start of the fourth leg is in some city. Even though most of the events will be in specific locations that can be specified in the usual way with zip code, street and number we wanted to allow for this "loose locations". Therefore the location of an event is a relation to instances of venue, with venue having only the attribute description. The subclasses of venue - country, city and location - allow for consecutively more detail in the description of the location.

Category

The category of an event is given by a relation to an instance of category or one or more of its subclasses. Category as top level class has only the attribute description. As part of the protocol we also defined a small ontology for the subclasses of category, so for instance concert is a subclass of category and has the additional attributes of band name and cast. Although we tried to build this ontology to encompass all possible events, we have to concede that we probably failed, but the beauty of RDFS is that this ontology can

[2] The event description contains for simplicity reasons not all possible entrance fees, but a minimum price – allowing fort the probably most common filtering criteria: Too expensive for me.

be easily extended - even in a decentralized and unorganized way - without breaking any application. If, for example, a music site decides that “concert” is definitely to coarse to be of much use in their context and defines HeavyMetalConcert, JazzConcert etc. as subclasses of concert, even applications unaware of this change, will still be able to return the Heavy Metal Concerts in reply to a query for “concert”.[3]

Having such an ontology allows for sophisticated searches across language boundaries, the easy implementation of query refinement and relaxation and should amend incompatibilities between different versions of this protocol – because many changes can be done by making new subclasses to the category classes.

Time frame

As already mentioned should we also be able to model events that not just happen once, but events that like a weekly market recur a number of times. We decided to use a simple model, only allowing for events that happen every day or every week during some time period; although our model definitely does not capture all possible patterns of recurrence, it does so for the most common cases in a simple way and all other cases can always be modeled by making one event per recurrence.

For events that just occur once, the time frame is an instance of the class time frame, having the attributes startDate and duration. Recurring time frames are modeled as instances of the classes dailyTimeFrame or weeklyTimeFrame, each being subclasses of recurringTimeFrame with the additional attribute durationOfRecurrence that defines for how long this pattern of recurrence lasts.

Composed Events

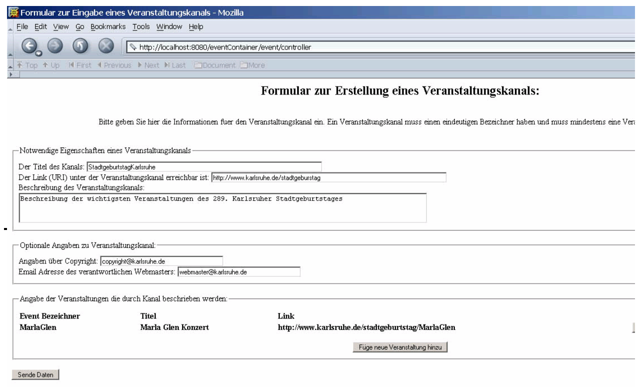
We modeled composed events by allowing that each event may have “childEvent” relations to any number of other events and a “parentEvent” relation to one event. This allows for hierarchies of arbitrary depth and should be enough to model almost all real live cases.

Event Channels

Analog to RSS we defined channel entities to describe who published this information. The attributes of channels are title, description, link, managingEditor and copyright.

The interested reader may also want to read Appendix A, where he finds a complete example of an event description.

6 The Server Side Editor


Formular zur Eingabe eines Veranstaltungskanals - Mozilla

http://localhost:8080/eventContainer/event/controller

Formular zur Erstellung eines Veranstaltungskanals:

Bitte geben Sie hier die Informationen fuer den Veranstaltungskanal ein. Ein Veranstaltungskanal muss einen eindeutigen Bezeichner haben und muss mindestens eine Vera

Notwendige Eigenschaften eines Veranstaltungskanals

Der Titel des Kanals: StadtgeburtstagKarlsruhe

Der Link (URI) unter der Veranstaltungskanal erreichbar ist: http://www.karlsruhe.de/stadtgeburstag

Beschreibung des Veranstaltungskanals:

Beschreibung der wichtigsten Veranstaltungen des 289. Karlsruher Stadtgeburtstages

Optionale Angaben zu Veranstaltungskanal:

Angaben über Copyright: copyright@karlsruhe.de

Email Adresse des verantwortlichen Webmasters: webmaster@karlsruhe.de

Angabe der Veranstaltungen die durch Kanal beschrieben werden:

Event Bezeichner	Titel	Link
MarlaGlen	Marla Glen Konzert	http://www.karlsruhe.de/stadtgeburtstag/MarlaGlen

File Edit View Go Bookmarks Tools Window Help
Top Up First Previous Next Last Document More
Füge neue Veranstaltung hinzu
Sende Daten

A server side editor allows the user to upload a description file, to change it as she pleases and to then download the new file (that she would than usually place on her website).

Our demonstrator of this technology was created with Java 1.4, Tomcat 4.x and the Jena RDF API[Jena, 2000], it uses RDQL[RDQL,2004] to access the RDF files.

It proved that it needs only very little resources and a free library to build one of the technological cornerstones of this Semantic Weblet.

7 Future Work

We truly believe that the Semantic Weblet described in this paper could work and that its wide spread adoption could make the Internet more useful. We are aware though, that it’s unlikely that this will ever happen, since the adoption of such a technology always depends on much more than just usefulness.

Speaking about the technology the protocol itself still needs some tweaking, adding (better) support for images, time zones, different currencies and extensions of the category ontology. Also a migration to OWL light could be evaluated, but we fear that there are still not enough OWL APIs to make it worthwhile.

A bit more complicated are questions surrounding malicious event announcement files – SPAM files, that would inevitable emerge. Currently nothing stops a file on a server in Nigeria from announcing that its Viagra sale is a part of the Tour de France; but quite simple rules are imaginable that would stop this kind of abuse.

Appendix A – Event Example

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF
 xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#" #
 xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
 xmlns:event="http://fzi.de/2003/09/event.rdf# "
 xmlns:eventChannel="http://fzi.de/2003/09/eventChannel.rdf#"
 xmlns:category="http://fzi.de/2003/09/category.rdf#"
 xmlns:venue=http://fzi.de/2003/09/venue.rdf#>
<eventChannel:eventChannel
  rdf:about="http://www.getgo.de/event.rdf#channel">
  <eventChannel:title>getgo.de</eventChannel:title>
  <eventChannel:description>
     Alle Arten von Veranstaltungen im Netz.
  </eventChannel:description>
  <eventChannel:link>
     www.geto.de
   </eventChannel:link>
  <eventChannel:copyright>
     webmaster@getgo.de
   </eventChannel:copyright>
   <eventChannel:managingEditor>
     webmaster@getgo.de
   </eventChannel:managingEditor>

   <eventChannel:event
     rdf:resource=
     "http://fzi.de/2003/09/channel_test#simplyRed"/>
   </eventChannel:eventChannel>
   <event:oneTimeEvent rdf:about=
     "http://fzi.de/2003/09/channel_test#simplyRed">
     <event:title>Simply Red</event:title>
```

```
      <event:description>
        Simply Red, oder besser gesagt der Sänger Mick Hucknall
        mit seinem Soul-Projekt Simply Red, gehen nach vielen
        Jahren Pause endlich wieder auf Tournee. Natürlich bleibt
        Mick Hucknall auch nach dieser langen Auszeit seinem alten
        erfolgreichen Sound treu, denn mit seinem souligen Pop
        konte Simply Red weltweit schon mehr als 45 Millionen
        Schallplatten verkaufen.
      </event:description>
      <event:link>http://www.getgo.de/simplyRed.html</event:link>
      <event:organizer>SWR3</event:organizer>
      <event:propPurchaseDetails>
        <event:purchaseDetails>
          <event:boolPublic>true</event:boolPublic>
          <event:entranceFee>55.85</event:entranceFee>
          <event:ticketLink>
            www.ticketcorner.de/simplyRed/Mannheim
          </event:ticketLink>
        </event:purchaseDetails>
      </event:propPurchaseDetails>

      <event:propCategory rdf:resource=
        "http://fzi.de/2003/09/category.rdf#instanceRockPop"/>

      <event:propTimeFrame>
       <event:timeFrame>
         <event:startDate>2003-12-08T20:00:00</event:startDate>
         <event:duration>P0Y0M0DT3H0M0S</event:duration>
       </event:timeFrame>
      </event:propTimeFrame>

      <event:propVenue>
        <venue:location
          rdf:about="http://fzi.de/2003/09/channel_test#rosen">
          <venue:description>
            Rosengarten Mozartsaal
          </venue:description>
          <venue:countryName>
             Deutschland
          </venue:countryName>
          <venue:countryID>de</venue:countryID>
          <venue:cityName>Mannheim</venue:cityName>
          <venue:zipCode>68161</venue:zipCode>
          <venue:street>Am Rosengartenplatz</venue:street>
          <venue:streetNumber>2</venue:streetNumber>
        </venue:location>
      </event:propVenue>
  </event:oneTimeEvent>
</rdf:RDF>
```

References

[Atom,2004] AtomEnabled: http://www.atomenabled.org/, 2004

[BlogChalking,2004] BlogChalking - Collaboratively mapping weblogs for smarter blogsearching. http://blogchalking.tk/, 2004

[BlogLines,2004] Blog Lines: http://www.bloglines.com/, 2004

[BLOGG.de,2004] BLOGG.de: htpp://www.blogg.de/

[CC,2004] Creative Commons Metadata http://creativecommons.org/technology/metadata/, 2004

[DublinCore,2004] Dublin Core Metadata initiative: http://dublincore.org/,2004

[FOAF, 2004] The Friend of a Friend project: http://www.foaf-project.org/,2004

[GeoUrl, 2004] GeoUrl: http://geourl.org/,2004

[GoogleNews,2004] Google News: http://news.google.com, 2004

[HP,1999] HP Invent: Semantic Web Vision, http://www.hpl.hp.com/semweb/sw-vision.htm, 1999

[Jena, 2000] HP Invent: Jena 2 – A Semantic Web Framework, http://jena.sourceforge.net/index.html, 2000

[Kaon, 2002] FZI & AIFB: KAON – Karlsruhe Ontology and Semantic Web Tool Suite, http://kaon.semanticweb.org, 2002

[MovableType,2004] MovableType: http://www.movabletype.org/,2004

[Pyrple,2004] pyrple (python rdf api) http://infomesh.net/pyrple/

[RAP,2004] RAP - RDF API for PHP V0.8, http://www.wiwiss.fu-berlin.de/suhl/bizer/rdfapi/, 2004

[RootBlog,2004] RootBlog: http://www.rootblog.com/

[RDQL, 2004] RDQL – RDF Data Query Language, RDQL - RDF Data Query Language, 2004

[Sambuca, 2004] Sambuca A Graph-Oriented RDF Toolkit,http://www.dannyayers.com/2004/03/sambuca/, 2004

[Sibler, 2003] RDF Austauschformat for Events, http://www.sibler.info/uni/studienarbeit/studienarbeit.html, 2003

[SharpReader,2004] Sharp Reader: http://www.sharpreader.net/,2004

[Straw,2004] Straw: http://www.nongnu.org/straw/,2004

[Syndic8,2004] Syndic8: http://www.syndic8.com/stats.php,2004

[Tomcat, 2004] Apache Tomcat, http://jakarta.apache.org/tomcat/, 2004

[XH,2004] Xhtml Friends Network: http://gmpg.org/xfn/,2004