The DBin platform: toward a personal tool to experience the Semantic Web

Giovanni Tummarello[1], Christian Morbidoni[1], Francesco Piazza[1], Paolo Puliti[1]

[1]Dipartimento di Elettronica, Intelligenza Artificiale e Telecomunicazioni
Università Politecnica delle Marche,
Via Brecce Bianche – 60131 Ancona (ITALY)
{g.tummarello, c.morbidoni, upf, p.puliti}@deit.univpm.it

Abstract. DBin is a novel kind of personal application which enables users to experience the Semantic Web by participating in P2P "discussion groups" and exchanging metadata and annotations about common topics of interest. The p2p transport layer is provided by the RDF-Growth algorithm which has characteristics of scalability and sustainability even in large real world communities. DBin is fully based on the syntax, semantics and philosophy of the W3C Semantic Web initiative and accommodates both a novel, domain scriptable user interface and a number of experimental modules to deal with specific kinds of metadata and information sources (audio metadata extraction, textual analysis, desktop integration). DBin includes an RDF subgraph digital signature facility enabling personalized trust policies to provide filtering out unwanted information. Maximum extendibility is guaranteed by the use of the Eclipse Rich Client platform and by the Open Source model.

1. Introduction

In this paper we illustrate our efforts toward DBin, possibly the first platform to deliver to end users the "feel" of the Semantic Web in an integrated environment, enabling them to participate in Semantic Web based P2P discussion groups. Using DBin users enjoy a straightforward way to cooperative building and browsing semantic knowledge which is completely encoded and follows the syntax and semantics of the W3C Semantic Web initiative [1] (namely it is based on RDF/RDFS[2] and OWL[3]).
DBin can in a sense be though as a file-sharing application for metadata. Similar to a file-sharing client, in fact, it connects directly to other peers, instead of files, however, it download and shares RDF metadata about resources which the group has defined "of interest". This creates a flow of RDF information which ultimately allows the participants to construct rich personal Semantic Web databases therefore supporting high speed local browsing, searching and personalized filtering and processing of information.

While there is no issue about how to “use” files that have been retrieved over a file sharing system, the case is very different for RDF information. It turns out that in fact each “topic of interest” often requires specialized user interface and domain specific rules in order for users to both browse it fruitfully and contribute with their own annotations.
Such inherent usability issue is in fact well known in the research field of Semantic Web visualization and interaction. In DBin we have therefore chosen to provide the user with a maximally configurable user interface that can be scripted according to the domain of interest of the single discussion group. To access a user community , the user is then suggested to download what we call a Brainlet, that is a package of configuration and a priori knowledge to best interact with the information shared in the group.
DBin main knowledge exchange module works in a P2P model based on the RDFGrowth algorithm, which is inherently scalable with respect to the number of participating peers, while providing an overall uncommitted social model designed to address the 'real world communities' scenario.
To fully support scalability in one such global annotation scenario, trust metrics and certainty of authorship for the information is needed. Based on the RDFContextToolkit [4] module, which provide RDF subgraph digital signature capabilities, DBin support personalized local trust policies and filters based on these.
Once the user collects metadata locally, a number of interesting applications become possible such as an add on module for the desktop integration. These, by indexing local files and extracting as much semantic as possible according to the specific formats, create additional RDF so that the user can seamlessly browse trough both locally and remote resources.
DBin is based on the excellent Eclipse Rich Client [5] platform and is Open Source, so such extension modules are easy to implement and experiment with.

2. The scenario

The scenario which DBin attempt to address is new under many aspects. Many of the P2P approaches based on Semantic Web technologies proposed so far [5][6][7][8], use metadata and ontologies to build a semantically structured definition of the resources a user is searching for and/or is offering to other participants in the network. In general these system are finalized to manage and optimize the retrieval of actual files, like textual documents, audio, video and so on. In such systems a user formulates queries like “All publications about Semantic Web by author X”, knows in advance precisely what to searching for, thus having a certain knowledge of the domain.

DBin is intended to address cases such as a user, perhaps new to Semantic Web, how is interested in learning (more) about the research that is going on, He/she is supposed simply to join a 'Semantic Web' topic group, to receive new and unexpected information, for example papers, the conferences on which they have been published, their author's names. Users are not really interested exclusively in “hits” locating remote resources, but rather into learning as much as possible about them so that more uses of this information become possible (e.g. Personalized browsing, joining with local information etc).

3. The RDFGrowth P2P engine: basic concepts

Previous P2P Semantic Web applications, such as [5][8], have explored interactions among groups of trusted and committed peers. In such systems peers rely on each other to forward query requests and collecting and returning results. In contrast, we consider the real world scenario of peers where cooperation is relatively frail. By this we mean that peers are certainly expected to provide some external service, but commitment should be minimal and in a “best effort” fashion.

The RDFGrowth algorithm has been designed to address this requirement of scalability and minimum commitment among peers, and is based on the peculiar philosophy of minimum external burden. Therefore peer are not required to perform any complex or time consuming operation, such as query routing, replication, collecting and merging.

3.1. RDFN: the only query allowed

As a complex graph query possibly simply hog any machine, the only RDF query allowed during metadata exchanges, is a simple and basic one,which not only is fast to execute but also can be cached very effectively. It is defined by the RDFN (RDF Neighbors) operator, which retrieves meaningful peace of information, in the form of RDF metadata, directly connected to a specific resource.

In detail the RDFN (RDF Neighbours) of a URI is a graph composed roughly by the by all the triples that have it as subject or object. In case some of these also refer to blank nodes, then the RDN is also recursively composed by all the triples until just "ground" (URI or Literals) node form the "edge" of the RDFN. Figure 1 shows the RDFN of a sample resource and its composing MSGs.

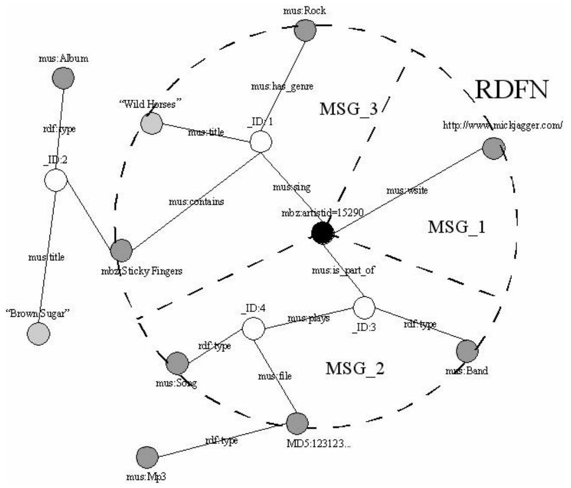


Figure 1 The RDFN of the resource painted in black is delimited by the dotted circle. White nodes represent blank nodes. The RDFN is composed by several slices, each one is an MSG, basically a closure on blank nodes starting from a given triple, and represents the minimum unit of knowledge that is exchanged.

3.2. MSGs

The RDFN(a) can be also considered as the union of all the MSG which involve the resource *a*. An MSG is a subgraph with a well defined structure and is actually the minimum amount of information that can be exchanged in the system. See [9] for details about the RDFN and MSG definitions and theory. Its interesting properties allows to exchange information in a fine granularity, incremental fashion, along with its context. We will see later, in fact, how authorship information can be efficiently attached at MSG level, allowing trust policies and revocations.

3.3. GUED (Group URIs Exposing Definition)

Users connects to a RDFGrowth network by selecting a topic group and joining it. The client then receive an operator, which defines, and is able to retrieve from a graph, the resources which the group is about. A GUED can be implemented as a set of queries. As an example, for a Michael Jackson group, a possible GUED might be "select all the URIs identifying his songs, albums, concerts and interviews", in respect to an agreed ontology.

Once received the GUED operator, the peer execute it once on his DB and the resulting set of URIs are "published" in the p2p network, as an advertisement that they are in fact of interest and will be willing to answer requests from other peers about the "RDF Neighbours" (RDFN, see next section).

3.4. RDFN Hashes and exchange strategy

The algorithm cycles over the set of 'on topic' URIs (selected by the GUED) and for each of them searches for peers who have different surrounding informations (RDFN) than the local ones. This process is performed by looking into a sort of DHT in which hashes of RDFNs (say simple MD5) are exposed by each peer. Once an hash is detected which is different from the one exposed by the local peer, an exchange is initiated.

During the exchange peers synchronizes their knowledge about the resource. In addition to simple hashes more advanced heuristics cab be applied to identify new information present in the network and choose the peer with who it is more profitable to ask to.

3.5. Considerations

A key point in this approach to metadata sharing is that the algorithm grows a local triple store at every peer, this not only enables fast browsing and complex query execution (performed using local computational power on the local DB, no external commitment), but also make possible for metadata to naturally cross the borders across communities. As an example, suppose that in a “movie community” someone posts a picture of an actor and in a “rap music” community the same actor has been mentioned as performer. Then a user participating at same time to both communities would, by the logic of DBin and the RDFGrowth algorithm, make so that the picture is also “posted” in the movie group.

While the approach fits well the target scenario, there are some shortcoming that might be mentioned:

- As the database continuously expands a large HD space might be required to store all the metadata from numerous or big communities. Given the cost/capacity of modern drives, we think that the HD space problem is very secondary. Actual con-

tent caching policy e.g. Images, can be changed at will, see the section about the URIBridge.
- Although the time it takes to get a new piece of information is bounded, the approach is not suitable for real time structured information (e.g. Current weather in RDF)
- It takes some startup time for a new user to get enough information for a meaningful navigation and querying.

Such a “growth only” scenario matches the monotonic nature of the RDF semantics. To obtain more information about a resource can’t in fact “hurt” since, by definition, previously inferred statements will still hold when new data becomes available. It is of course possible, in the real world applications, to rely on “context” information to later apply non monotonic rules on the local database, but it should remain a local peer decision to do so with no consequences on the shared knowledge. Digital signatures are an example of such context information which support several fundamental higher level non monotonic behaviours of the overall system.

4. Dealing with the actual data

Relaying on RDFGrowth, DBin users only exchanges pieces of RDF graphs describing the resources of interest, which might be real world concepts (such as a person) or digital content (e.g. mp3 files, pictures, documents) actually retrievable on the Internet. It is clear that in many cases the user would like to be able not only to explore or add annotations, but also to reach the actual data. As an example if someone, say Bob, joins the ”Semantic Web Research” research group and founds out that there is a new paper focusing on “Full-text search over RDF graphs”, he/she will be probably interested in knowing at which conference it was presented or the author names. Then, after having discovered that the conference was an important one and that the authors already published some good papers on this field, he/she will probably want to read the actual article.

Given this, it is clear that some facility is needed in DBin to provide the user with the digital content referred by the RDF metadata. But also an 'upload' mechanism is needed to let the users be able to share his digital data (e.g. Images, text etc). Referring to the previous example, consider that Bob already knows the paper and he wants to attach a comment to it and a picture of the author having a talk on it at a conference. I this case Bob wants the other participants to be able not only to see the metadata he added but also to take a look at the picture itself, and possibly leave a comment about it. In DBin this facilities are provided by the URIBridge module.

While the download mechanism is straightforward once a URL is available for a specific resource, as it can be retrieved, for example, over standard HTTP protocol, the uploading part requires some further considerations. The URIBridge is based on upload servers where users can store files they want to share (e.g. pictures, text, mp3s). After having uploaded a resource, the user is provided with a URL which can be used to create annotations about the data as well as to retrieve that data in oder to visualize it.

So every DBin client can be configured to work with one or more upload servers, much like an E-Mail client requires a SMTP server. While the default installation of

DBin comes with a simple upload server, this limits the users to small files. For power users, installing a personal upload server is however trivial, just the deploy of a simple PHP script.

5. Identities and authorship of annotations

5.1. Authorship rather than provenance

In such a system, which deals with potentially large and unregulated communities, it is important to have information about who said what, in particular which user is the author of a particular annotation received from the network.

As the system is based on high replication of metadata over the P2P group, to look at the provenance of the metadata (which peer sent it to me), is useless in order to obtain authorship information. In fact, if Bob's client sends a comment about a paper to someone else, it does not mean that Bob himself wrote the comment, as it might come from an other user, who again might obtained it by someone else.

As is evident that tracking the provenance is not thinkable in our system, we have to provide a methodology for 'marking' every peace of metadata added to the system with verifiable statements about the authorship. Moreover, this authorship information has to be attached to the metadata in a way that make it possible to hold them during the replication and exchange process.

5.2. User identities and digital signatures in DBin

The MSG definition and properties highlighted in the previous section, when combined with a canonicalized serialization as suggested in [10], enable signing MSGs themselves in a efficient way: attaching the signature information only to a single triple composing the MSG. This methodology, described in detail in [4], also assures that the context (in this case the authorship) will remain within the metadata when they will be exchanged over the network, as well as enables multiple signature to be attached to the same MSG, also at different times.

When started up for the first time, DBin clients require the users to provide a valid URI which will act as an identifier for the user itself (for example a mailto URL or a web page). Then a public and a secret keys are generated; the private key is stored locally, while the public one is uploaded by means of the URIBridge, just as it happens for files. Every time a user will add an annotation to the system, that is a certain number of MSGs, it will contain the user's identifier as well as the URL of the public key, and will be signed using the user's private key.

In this way, after having received a peace of metadata from the P2P group, clients are able to retrieve the public key and to identify the author of the annotation, without caring about the provenance of the metadata itself.

5.3. Information filtering and revision

The mechanism described in the previous section provide a framework for authorship assessment, trust based filtering and information revision.

Once the authorship of a MSG can be verified, a variety of filtering rules can be applied at will. These, in DBin, are always non-destructive; information that doesn't match certain trust criteria can be hidden away but does not get deleted. Is it straightforward, for example, to implement a local 'black list' policy, allowing users to add authors to that list and to filter the local knowledge in order to hide all the information signed by the same user's identity.

Moreover, the hash of a signed MSG, can be used as an Inverse Functional Property (IFP, see), that is, as a unique way to name to the MSG itself. This in turn can be used in a subsequent MSG to indicate the one that it substitutes. Given that the paternity of this subsequent MSG can be verified to be identical, the client can safely perform the information update, no matter where it received the update patch from.

DBin also includes a preliminary support for a centralized certification authority, dubbed RDFTrust. By obtaining a signed certificate (which involves a time consuming identification procedure), users can enjoy instantly an higher degree of trust. If trust was to be abused (e.g. By spamming or malicious "ontology violations") the certificate would be revoked thus causing all the previously inserted information to "sink" in each DBin installation. The certification authority web site is currently being implemented as a web application with a matching application API.

6. User interface: "Brainlets"

There has been a lot of work recently on Semantic Web visualization and a number of user interface have been proposed [11], [12], [13], [14], [15].
While pro and cons can be argued for each specific approach, it is clear that user interface issues are a complex issue with no clear single solution.

Rather than a single answer to this issue, DBin provides a general set of "application oriented" generic GUI tools by which 'power users' can build applications specifically targeted to the domain of interest.

DBin *domain specific applications*, are called "Brainlets", and can in a sense be directly related to the concept of "GUED", the operator which decides how to select the knowledge to be exchanged in the p2p group, from the RDFGrowth algorithm.

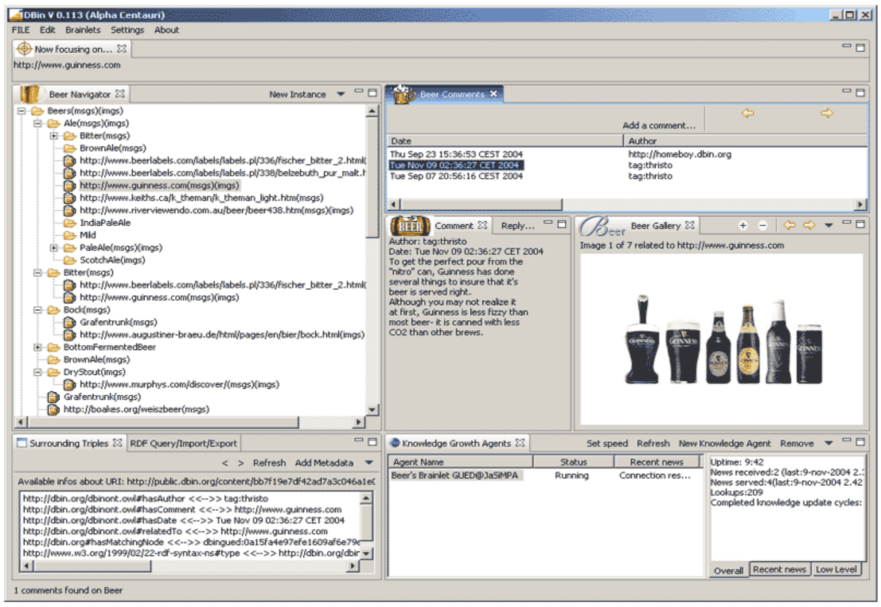
Brainlets can be though as "configuration packages" preparing DBin to operate on a specific domain (e.g. Wine lovers, Italian Opera fans etc..). Given that Brainlets include customized user interface, the user might perceive Brainlets as full "domain applications" which are run by DBin. which come with regarded as "integrated packages" which describe and implement "domain applications".

In short Brainlets are composed of:

- The setup information for the RDFGrowth algorithm and the transport layer to connect with others using the same Brainlet (namely, at this point, the name of the randevouz servers, the channels and the GUED)
- The ontologies to be used for annotations in the domain (e.g. The beer ontology).
- A general GUI layout;. which components to visualize (e.g. A message board, an ontology browser, a "detail" view) and how they are cascaded in terms of selection/reaction
- Templates for domain specific "annotations", e.g., a "Movie" Brainlet might have a "review" template that users fill. This allow a "reviews" view to have useful or-

derings based on the known fields in the review. The GUI for the templates is generated automatically

- Templates for readily available, “pre-cooked” domain queries, which are structurally complex domain queries with only a few simple free parameters, e.g. “give me the name of the cinema where the best movie of genre X is being shown tonight”.
- A suggested trust model and information filtering rules for the domain. e.g. Public keys of well known “founding members” or authorities, preset “browsing levels”.-Support material, customized icons, help files etc..
- A basic RDF, GUED conforming, knowledge package



http://www.guinness.com

```
Beers(msgs)(imgs)
  Ale(msgs)(imgs)
    Bitter(msgs)
    BrownAle(msgs)
    http://www.beerlabels.com/labels/labels.pl/336/fischer_bitter_2.html(
    http://www.beerlabels.com/labels/labels.pl/338/belzebuth_pur_malt.h
    http://www.guinness.com(msgs)(imgs)
    http://www.keiths.ca/k_theman/k_theman_light.htm(msgs)
    http://www.riverviewendo.com.au/beer/beer438.htm(msgs)(imgs)
    IndiaPaleAle
    Mild
    PaleAle(msgs)(imgs)
    ScotchAle(imgs)
  Bitter(msgs)
    http://www.beerlabels.com/labels/labels.pl/336/fischer_bitter_2.html(
    http://www.guinness.com(msgs)(imgs)
  Bock(msgs)
    Grafentrunk(msgs)
    http://www.augustiner-braeu.de/html/pages/en/bier/bock.html(imgs)
  BottomFermentedBeer
  BrownAle(msgs)
  DryStout(imgs)
    http://www.murphys.com/discover/(msgs)(imgs)
  Grafentrunk(msgs)
  http://boakes.org/weiszbeer(msgs)
```

Date	Author
Thu Sep 23 15:36:53 CEST 2004	http://homeboy.dbin.org
Tue Nov 09 02:36:27 CET 2004	tag:thristo
Tue Sep 07 20:56:16 CEST 2004	tag:thristo

Author: tag:thristo
Date: Tue Nov 09 02:36:27 CET 2004
To get the perfect pour from the "nitro" can, Guinness has done several things to insure that it's beer is served right.
Although you may not realize it at first, Guinness is less fizzy than most beer- it is canned with less CO2 than other brews.

Image 1 of 7 related to http://www.guinness.com

Available infos about URI: http://public.dbin.org/content/bb7f19e7df42ad7a3c046a1e(

```
http://dbin.org/dbinont.owl#hasAuthor <<-->> tag:thristo
http://dbin.org/dbinont.owl#hasComment <<-->> http://www.guinness.com
http://dbin.org/dbinont.owl#hasDate <<-->> Tue Nov 09 02:36:27 CET 2004
http://dbin.org/dbinont.owl#relatedTo <<-->> http://www.guinness.com
http://dbin.org#hasMatchingNode <<-->> dbingued:0a15fa4e97efe1609af6e79
http://www.w3.org/1999/02/22-rdf-syntax-ns#type <<-->> http://dbin.org/dbir
```

Agent Name	Status	Recent news
Beer's Brainlet GUED@JaSiMPA	Running	Connection res...

```
Uptime: 9:42
News received:2 (last:9-nov-2004 2.:
News served:4(last:9-nov-2004 2.42
Lookups:209
Completed knowledge update cycles:
```

1 comments found on Beer

Figure 2 A screen shot of the Beer2Beer Brainlet running. The principal “views” are: an ontology (and instances) browsing Navigator, the Knowledge Agents view, showing statistics about the currently running knowledge agents, and a set of “Annotation” views. Among these a comment view, a picture gallery and an “annotation listing” view.

Most importantly, Brainlets can be created as much as possible with no programming skills, that is, just by editing XML configuration files; more advanced Brainlets can however be made including custom Eclipse plug-ins as needed. Most of the previously mentioned features have been implemented as shown in figure 2, a screen shot of “Beer2Beer”, our first XML based Brainlet.

6.1. Ontology issue and social model

Brainlets are therefore preloaded by power users with domain specific user interaction facilities, as well as with domain ontologies suggested by the Brainlet creator. This seems to induce an interesting social model, mostly based on consensus upon Brainlets choice, which can help some of the well known issues in distributed metadata environments, a central one being the ontology mismatch problem. Brainlets, by providing an aggregation medium for ontologies, users, data representation structures, are therefore good catalyst of the overall semantic interoperability process.

In fact, as users gather around popular Brainlets for their topic of choice, the respective suggested ontologies and data representation practice will form an increasingly important reality. If someone decided to create a new Brainlet or Semantic Web application in general which could target the same user group as the said popular Brainlet, it is clear that there would be incentive in using identical or somehow compatible data structures and ontologies.

7. Desktop integration module

The effort in using RDF and other Semantic Web languages and tools to help management of desktop resources and applications is known as the Semantic Desktop and has become a very active research field in last years. In Gnowsis [16], has been presented which uses different plug-ins to represent local files in a RDF graph. Users are allowed to add annotations and browse local resources taking advantage of the semantic connections established. A similar tool is described in [17], where also the usefulness of metadata sharing capability is highlighted.

One of the most interesting aspects of DBin is that metadata coming from local resources can be merged with those coming from external ones (e.g. the RDFGrowth P2P groups). A few modules are dedicated to the extraction of metadata from local resources. A local file system processor creates a “semantic” representation of the available files, that is a RDF graph where nodes points to local files by means of ”file:” URIs, then this graph is imported inside the triple store. Different techniques are applied to extract metadata according to the file type. These include full text analysis and indexing, file type specific operation (e.g. ID3 tags extraction) and file name and path heuristics.

Once such an “enhanced” RDF graph has been created, different techniques can be applied to calculate a semantic distance between nodes. Current implementation uses a spread activation algorithm, where edges are weighted according to the ontologies properties they represent. This 'distances' are used both to build an index for full text, which not only takes care of the content of files but also of the metadata describing them, and to provide browsing capabilities. The idea is that the user might be able to find interesting connections between remote resources he/she is browsing and local files which are related to them. A complete discussion of the approach and the method used here is out of the scope of this paper.

8. Software Engineering/License

DBin is programmed in Java and based on the Eclipse Rich Client platform. As such, DBin is naturally multi-platform, features an OS native look and feel and is highly extensible trough the well known Eclipse plug-ins and extension points technology. Both the framework and modules presented here are open source. Licensing terms have not been settled yet, but they're expected to be either LGPL, BSD or CPL.

9. Conclusions

DBin is novel semantic web application aimed at regular users and providing them with relatively simple means to create and enjoy “the Semantic Web”. DBin is meant to demonstrate the usefulness and scalability of a model where a large, local, semantic web database is grown and rich user interfaces and filtering is then applied a posteriori.

Under a usage point of view, while DBin's differ fundamentally from the way the current Web is used, it is not so much different from popular P2P file sharing applications. In the same way as many users have gotten used to wait to obtain data by running a classic P2P file sharing, DBin users will “peacefully” discover new information about topics in which they express interest in. Content and annotations produced by the user, on the other hand, can reach precisely those who had expressed interest in it and naturally cross the boundary of the P2P group they were posted originally to. Given RDFGrowth design in fact, relevant annotations are intrinsically and automatically bridged by the peers that visit multiple groups or return at later times.

Further research and implementation work will be directed to enhance integrating external metadata sources. In particular we will further investigate the interaction with the semantic desktop, in order to provide the user with novel and interesting way to connect the local data and the knowledge shared in P2P groups.

All the work presented here has been implemented in Java as Free Software and is currently available at the respective websites. Version 0.3 is about to be released and this will be the first public release since the start of the project. While the project holds great promises, actual use and adoption by real user communities will have to be monitored and fostered. Given the novelty of the application, we in fact expect that a very large amount of work will have to be done to polish the usability and to provide with the needed social/group model interactions facilities (e.g. some form of e-bay style trust system might be interesting).

References

[1]Semantic Web Activity, http://www.w3.org/2001/sw/

[2]RDF Primer, http://www.w3.org/TR/rdf-primer/

[3]OWL Web Ontology Language Overview, W3C Recommendation 10 February 2004, http://www.w3.org/2001/sw/WebOnt/

[4] G. Tummarello, C. Morbidoni, P. Puliti, F. Piazza , "Signing individual fragments of an RDF graph" 2005 WWW2005, poster track

[5]Eclipse Rich Client Platform, http://www.eclipse.org/rcp/

[6] Wolfgang Nejdl, Boris Wolf , "EDUTELLA: A P2P Networking Infrastructure Based on RDF" 2002 WWW2002, Honolulu

[7] Min Cai, Martin Frank , "RDFPeers: A Scalable Distributed RDF Repository based on A Structured Peer-to-Peer Network" 2004 13th International World Wide Web Conference WWW2004, New York

[8] Wolfgang Nejdl, Wolf Siberski, Martin Wolpers, Alexander L”ser, Ingo Bruckhorst , "SuperPeer Based Routing and Clustering Strategies for RDF Based Peer-To-Peer Networks" 2003 Twelfth International WWW03 Conference, Budapest

[9] Paul - Alexandru Chirita, Stratos Idreos, Manolis Koubarakis, and Wolfgang Nejdl , "Publish/Subscribe for RDF-based P2P Networks" 2004 ESWS, Heraklian, Greece

[10] Giovanni Tummarello, Christian Morbidoni, Joackin Petersson, Paolo Puliti, Francesco Piazza , "RDFGrowth, a P2P annotation exchange algorithm for scalable Semantic Web applications" 2004 First P2PKM Workshop, Boston

[11] Jeremy Carroll , "Signing RDF Graphs" 2003 ISWC2003

[12] R. Albertoni, A. Bertone, M. De Martino , "Semantic Web and Information Visualization" 2004 Proceedings of the First Italian Workshop on Se-

mantic Web Applications and Perspectives, Ancona (ITALY)
[13] "RDF Gravity - RDF Graph Visualization Tool" Technical Report: HPL-2004-57
[14] E Pietriga , "Isaviz: a visual environment for browsing and authoring rdf models " 2002 11th International World Wide Web Conference
[15] , "RDFX" Technical Report: HPL-2004-57
[16]Welkin, a graph-based RDF visualizer, , 2004, http://simile.mit.edu/welkin/
[17] Leo Sauermann , "Gnowsis Semantic Desktop ISWC2004 Demo" 2004 ISWC 2004 - Demo Track
[18] Robert MacGregor, Sameer Maggon, Baoshi Yan , "MetaDesk: A Semantic Web Desktop Manager" 2004 International Workshop on Knowledge Markup and Semantic Annotation (SemAnnot 2004), Hiroshima