The DBin platform: A complete environment for Semantic Web Communities

Giovanni Tummarello\textsuperscript{a}, Christian Morbidoni\textsuperscript{b,∗}

\textsuperscript{a} DERI Galway, National University of Ireland, Galway, Ireland
\textsuperscript{b} SeMedia Group, Università Politecnica delle Marche, Ancona, Italy

Keywords:
Peer-to-peer
Semantic Web Communities
RDF
Collaboration
Cooperative knowledge management

\textbf{Abstract}

DBin is a Semantic Web application that enables groups of users with a common interest to cooperatively create semantically structured knowledge bases. Such Semantic Web Communities are made possible by creating customized user environments called Brainlets. Brainlets provide user interfaces and domain specific tools (e.g., querying, viewing and editing facilities) that enable community participants to interact with the data of interest. Brainlets are directly created by domain experts using an XML description language. DBin clients communicate and exchange annotations using a P2P infrastructure. Access control and digital signatures, put by DBin inside the authored RDF, enable trust and information filtering.

\section{1. Introduction}

The W3C Semantic Web initiative has been active for a consistent number of years, and Semantic Web programming tools and libraries have reached a certain maturity. It has been widely noticed, however, that very few, if any, applications are available today for the end user to clearly experience at least some of the promises of the Semantic Web vision. In this article we describe the DBin project,\textsuperscript{1} which aims at creating a Semantic Personal Knowledge Manager (S-PKM) with the following main features:

- Being based on Semantic Web languages and usable in different domains by applying specific ontologies and settings, but yet enabling users to have a merged view of all the knowledge pertaining to different real world domains.
- Making use of ontology-based reasoning, whenever possible, for assisting the user in visualizing, editing and browsing semantic data.
- Working as a personal information manager and being integrated with the local desktop environment.
- Being interconnected with other S-PKM installations and with external data sources, thus enabling collaborative semantic knowledge authoring.
- Being powerfully adaptable to different domains and communities without the need for programming. Domain experts, rather than programmers, should be able to create domain specific applications on top of the platform and deliver them to end users in a simple, integrated, intuitive way. These domain specific applications should possibly co-exist in the same SPKM installation, interact among each other and share data.

In our opinion, seeking the realization of such an integrated tool is important. It serves both to validate the individual Semantic Web components as useful in large use cases, and to possibly discover the need for new components and infrastructures.

In developing DBin it became evident that, on top of the existing tools, there was the need both for a number of novel infrastructures and for pragmatic decisions, that in some cases would limit the excessive freedom, inherent in the Semantic Web vision, in favor of actual usability. Some of the topics which required novel solutions in terms of infrastructural components has been:

- Transport layer; this deals with the problem of publishing, importing and discovering semantic web knowledge.
- Authorship and trust; how the semantically structured information are digitally signed and how this enables personalized filtering.
- Semantically structured information visualization and domain specific, highly customizable user interfaces.
- Agreement on domain ontologies and entity names.

\textsuperscript{∗} This work has been partially supported by the European FP6 project inContext (IST-034718), by Science Foundation Ireland under the Lion project (SFI/02/CE1/1131), and by the European project DISCOVERY (ECP-2005-CULT-038206).

\textsuperscript{1} Corresponding author.
E-mail addresses: giovanni.tummarello@deri.org, g.tummarello@gmail.com (G. Tummarello); christian@deit.univpm.it (C. Morbidoni).

\textsuperscript{1} DBin web site: http://dbin.org.
The first two issues are dealt in DBin with new methodologies and subsystems. In Section 3 we will overview the RDFGrowth P2P algorithm, that enables topic based sharing of data, and the way such semantic data can be published and retrieved from the Web. Furthermore, DBin offers a digital signature methodology enabling information authorship verification and local trust based filtering policies (Section 4).

DBin, on the other hand, pragmatically approaches the remaining issues with its Brainlet plug-in model. Brainlets are introduced in Section 2, where the overall scenario is described, and then discussed in more detailed in Section 5. Finally, in Section 6, we present the results of a user survey to verify reaction to the novel scenario and application model.

DBin has been completely implemented in Java, on top of the Eclipse RCP\(^2\) framework, and therefore is platform independent and has a plug-in structure which supports agile extensibility. Current releases can be downloaded from the project’s web site.

2. The Semantic Web Communities model: high level system architecture and user experience

In our system we distinguish between two different user’s behaviours: they might simply want to existing Semantic Web Communities, thus being able to cooperatively build the community semantic knowledge (which we call end users), or might be interested in starting up and/or maintaining communities (power users). The power user starts up a new community by first creating a customized user environment, called Brainlet, for the editing and exploitation of semantically structured annotations.

Brainlets are plug-ins in the DBin platform and can be thought of as configuration packages preparing the client application to operate in a specific domain (e.g., wine lovers, Italian opera fans, etc.). From the user perspective, the relationship between Brainlets and the DBin platform is similar to that between HTML and a Web Browser. Much like HTML web sites are developed in an XML-based language and then parsed by a web browser resulting in a presentation of data, Brainlets are created in XML and RDF and then rendered by the DBin platform as rich Semantic Web, end-user applications. Although this process does not require programming, it needs a basic knowledge of Semantic Web query languages (in particular SeRQL\(^[7]\)) and understanding of RDF(S). Brainlets contain configurations which customize different functionalities and components of the DBin platform, including:

- The ontologies to be used for supporting knowledge creation and presentation of data.
- GUI layout and coordination. Each Brainlet defines a specific GUI, which is composed by several widgets. Widgets are first instantiated from a rich set of predefined ones and then configured for the domain of interest, e.g., an “ontology navigator” widget might be configured to show certain classes or instances and to hide others. Such widgets are then interlinked among each other. This means that chains of reactions to actions, such as a selection change, can be defined. For example, the selection of a class in an “ontology navigator” widget might cause and other widget (e.g., an instance navigator) to show all the instances of the selected class.
- Templates for domain specific annotations (e.g., a Movie Brainlet might have a Review template with associated slots that users can fill).
- 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 cinemas where a movie of genre X is being shown tonight).
- A trust model and information filtering rules for the domain (e.g., basing on public keys of well known founding members or authorities).
- Scripts for guiding the user in creating new identifiers (URIs) for domain resources (e.g., adding a new “paper” to the knowledge base).
- Scripts connected to Brainlet specific menus or buttons that implement domain specific functions.
- Support material, customized icons, help files, etc.
- Optionally, Brainlets might contain Java code and libraries for add-on capabilities beyond those provided by the standard Brainlet widgets.

To the end user, most of the above aspects are simply hidden behind the integrated Brainlets UI which presents itself, for example, as shown in Fig. 1 (ESWC Budva Brainlet). It is important to notice that the Brainlet UI is not simply a mash up of visualizers. As the components are coordinated among each other, the result is that a Brainlet guides the user into a meaningful and domain specific workflow interaction with the structured data. At any time, the domain ontologies are used as much as possible for assisting users in editing and browsing knowledge, for example suggesting which kind of annotations are possible for a given resource.

Multiple Brainlets can be present at the same time into a DBin installation. When this happens they share the same RDF dataset, thus providing different views on the same knowledge base.

In Fig. 2 we show our scenario and its main actors. Once Brainlets have been created by power users and made available on a Web site, they are downloaded and installed by the end users into their local DBin client, which renders them as domain specific GUIs. Brainlets also have roles in how a user can connect to the others and exchange domain specific data. This is done by including pointers to P2P topic channels. A P2P topic channel is a sort of virtual room, which the user can decide to join in. Once a user has joined a P2P topic channel, he/she automatically acquires knowledge, pertaining a specific topic of interest, from the other participants.

A specific P2P topic channel can be created by configuring an RDFGrowth server, which acts as a meeting point for the DBin clients (but do not carry themselves metadata or binary attachments). While RDF metadata is exchanged among users by RDF-Growth (Section 3), binary attachments are automatically stored by DBin in a web accessible space. Moreover, users can create and publish RSS feeds and RDF dumps derived from the internal knowledge.

Let us consider a possible scenario where Bob is a researcher interested in knowing more about a topic he recently discovered: the Semantic Web. He has heard of the DBin platform so he downloads and install it.

At start-up he uses a graphical wizard for creating his own identity, choosing a nickname and automatically generating a public/private key pair. Then a list of “World P2P servers” is shown, resembling familiar file sharing applications. By selecting one of the servers, Bob accesses a list of P2P topic channels which the server provides access to. He spots one, hosted at the address of a well-known semantic web research group and decides to join the community. As Bob is new to such community, DBin suggests him to install the community’s default Brainlet, called “SW Research Brainlet”: he could continue without the Brainlet (still being able to acquire community’s knowledge), or go trough the Brainlet installation (thus enabling assisted knowledge visualization and editing facilities). Bob chooses the second option and DBin downloads and installs the Brainlet.

Bob can now browse the specific page which presents collections of Semantic Web subtopics, people, conferences, papers,
mailing lists, etc. The environment enables browsing, searching and interaction. While Bob is connected to the P2P channel, he sees new information arriving and incrementally populating the graphic interface. Bob leaves DBin running and connected to the channel and, from time to time, he learns new information about his papers of interest (e.g., comments, attachments, and semantically structured data and links among resources). He can then contribute by adding more annotations which, in a short time, will reach others in the group.

From the main group, Bob gets to know about the ISWC conference. He looks up the URI across the servers and finds out that there is one group which mentions it: the ISWC group. Bob decides to join
this group, possibly installing the ad hoc “ISWC Brainlet”. By keep-
ing the connection to the ISWC active, Bob imports a lot of infor-
mation about the ISWC conference series (e.g., locations of the past edi-
tions, tracks, best papers, organizers, delegates, etc.). If the two Brainlets are compatible, e.g., they use the very same ontology for modeling the same concepts, the knowledge from the two groups can be browsed as one by either Brainlet. For example, if the two Brainlets use the FOAF ontology for describing persons, the ISWC papers authors could be visible from the SW Research Brainlet.

This model allows users to incrementally build an heteroge-
neous knowledge base, with information coming from different com-
munities, having at disposal different domain specific appli-
cations (the Brainlets) to meaningfully explore it.

3. The metadata transport layer

3.1. P2P and Semantic Web: related works and RDFGrowth

The P2P model as a transport medium for RDF has been inves-
tigated in several previous works. Edutella, described in [1], allows distributed queries within a federation of peers and has been later extended [3] to improve its scalability by introducing schema-
based routing and clustering. RDFPeers, discussed in [2], is an other interesting approach to build scalable distributed RDF repositories. A P2P publish/subscribe system, as an alternative to explicit query based approaches, is described in [4].

Such systems, where peers rely on each other to forward query requests and collect the results, perfectly fits specific scenarios where all peers are trusted sources, interested in providing a valu-
able service to the others. In contrast, RDFGrowth addresses a scenario of open and unregulated communities, where peers are ex-
pected to provide some external service, but with minimal com-
mitment and in a best effort fashion. In RDFGrowth Peers are not required to perform any complex or time consuming operation, such as query routing, collecting and merging.

In general, previous approaches provide support for searching across a federation of repositories which usually agree to execute distributed queries. In RDFGrowth, on the other hand, peers agree (by joining a community) on a definition of which resources are of interest and then exchange all the knowledge they have about those resources with other peers in the same topic channel. The result is that both new resources of interest and unexpected inform-
ation about them are discovered and imported without formulating explicit queries. The knowledge collected from the P2P network is then stored locally at each client, enabling efficient querying without generating external traffic or computational load. More than that, it becomes possible for the user to browse such information and discover resources by free exploration. Having information locally, as opposed to querying distributed remote systems, enables global scalability and maximum personalization of the way infor-
mation can be used.

In the next section we give an overview of the RDFGrowth algo-
rithm, while a detailed discussion can be found in [5].

3.2. Overview of the RDFGrowth algorithm

In general, RDFGrowth provides synchronization among dis-
tributed RDF datasets. However, such a synchronization is not performed in full, but along aspects of knowledge, and is restricted to that information that is considered of interest by a given com-
munity.

3.2.1. The GUED

Each P2P community has an interest banner, that we call Group URIs Exposing Definition (GUED), which defines what is “of inter-
est” for the community and is basically used to filter the RDF data exchanged among the community participants. A GUED is usually implemented as a set of queries which results are lists of URIs. A GUED for a Semantic Web Research community could be composed, for example, by the following queries (here formulated in natural language): “Select all resources of type Papers which have topic Semantic Web”, “Select all resources of type Person which have Semantic Web as a research topic” and “Select all resources of type Conference which have Semantic Web among the topics”.

It has to be noticed that such queries will eventually refer to terms (classes and properties) of a specific ontology. In DBin a P2P group is usually created in conjunction with a specific Brainlet (pos-
sibly by the very same person) and the GUED queries refer to terms defined in one (or more) of the ontologies used by the Brainlet. Upon joining a community, a peer runs the GUED queries on the local RDF dataset, thus selecting the local set of resources which are of interest within the group. Then each peer is allowed to give out only those bits of information which describe one of these resources.

3.2.2. The RDF Neighbours

In RDFGrowth, such bits of information are called RDF Neigh-
bours (RDFN). The RDFN of a URI A, with respect to a graph G, is a graph composed by all the triples in G that have A as subject or object. In case some of these contain blank nodes, all the state-
ments in the graph which involve such blank nodes are included until just “ground” (URI or Literals) nodes form the edges of the RDFN. Intuitively the RDFN of a resource is all the information a peer has that directly involves the resource itself. In RDFGrowth, the RDFN of a resource is the only query that peers are required to execute. This fosters scalability as RDFN allows effective caching and fast execution.

3.2.3. The RDFN exchange strategy

Once having joined a group and run the GUED on the local RDF graph, each peer publishes all the selected URIs and, for each of them, an hash value made from its RDFN (e.g., a simple MD5 of the RDFN in a canonical form). The result is a structure that is concep-
tually similar to a Distributed Hash Table (DHT), being acceptable by all the peers in the same group, with the URIs as key and the RDFN hashes as value. By looking into such a data structure, each peer can discover new URIs of interest and new information about already known URIs. This is done by confronting the local RDF hash with the remote one: if they mismatch the RDFN exchange is executed.

3.2.4. Direct URI Lookup

An RDFGrowth server can host multiple topic channels and keeps track of the URIs of interest to each of them. This enables a peer to explicitly look up a URI, resulting in all the groups which mention the URI itself, thus providing a simple way for users to discover new communities to be joined.

3.3. Interaction among Semantic Web Communities

Fig. 3 shows a possible use case where users participate in one or more communities. Alice (marked A) participates in a single community where knowledge about web development tools, technologies, program-
Fig. 3. DBin and its relationship with different actors in the "Semantic Web Communities".

...munities''. Fig. 3. DBin and its relationship with different actors in the "Semantic Web Communities".

If communities share identifiers (e.g., stable URLs to identify web applications, web technologies, etc.) then an annotation (e.g., a URI that is of interest to both the communities (i.e., the URI belongs to the GUED of both groups). For example, if a new content relevant to the users participating in the different communities, it is important to track the authorship of exchanged information owned at a peer which adhere to the GUED of a specific topic group.

In both cases one can decide to publish his/her own view on the data, that is filtering the overall knowledge based on local trust policies and revocations (discussed in the next section). When an RDF Dump is being published periodically by a community leader (power user), it turns out to be a way of providing an official web accessible representation of the knowledge that is being incrementally created within the P2P community. Each RDFGrowth server can be queried via HTTP to retrieve the list of the available communities and for each of them a URL can be specified where the official dump is stored.

4. Information filtering, revision and publication

4.1. MSGs

Before describing DBin's support for information filtering and revision, let us introduce the concept of Minimum Self-contained Graph (MSG). The RDFN of a resource (introduced in Section 3.2) can be decomposed in smaller pieces, called MSGs, which represents the minimum amount of RDF knowledge that can be exchanged in RDFGrowth. A formal definition of MSGs and their theoretical properties is given in [6]; intuitively, these are fragments of RDF graphs composed by a starting triple and, in the case a blank node is involved, all the triples that involve such a blank node. The procedure is executed iteratively on each triple collected. In this way the MSG border is never a blank node, but always a URI or a literal. From the MSG definition, it can be formally derived [6] that a generic RDF graph can be unequivocally decomposed into a set of MSGs and that, given an RDF graph and one of its statements, the statement is included in one and only one MSG. Furthermore MSGs can be expressed in the form of a canonical string, for example using the algorithm described in [8]. By hashing such a string we obtain a unique, content based, identifier for the MSG.

4.2. Information authorship

As DBin deals with potentially large and unregulated communities, it is important to track the authorship of exchanged information. As the system is based on high replication of metadata, we cannot base on provenance tracking as each user shares RDF data from other sources (e.g., local RDF files); (b) to publish a topic based filtered sub-set of the overall local dataset, that is all the information owned at a peer which adhere to the GUED of a specific topic group.

In both cases one can decide to publish his/her own view on the data, that is filtering the overall knowledge based on local trust policies and revocations (discussed in the next section). When an RDF Dump is being published periodically by a community leader (power user), it turns out to be a way of providing an official web accessible representation of the knowledge that is being incrementally created within the P2P community. Each RDFGrowth server can be queried via HTTP to retrieve the list of the available communities and for each of them a URL can be specified where the official dump is stored.

4. Information filtering, revision and publication

4.1. MSGs

Before describing DBin's support for information filtering and revision, let us introduce the concept of Minimum Self-contained Graph (MSG). The RDFN of a resource (introduced in Section 3.2) can be decomposed in smaller pieces, called MSGs, which represents the minimum amount of RDF knowledge that can be exchanged in RDFGrowth. A formal definition of MSGs and their theoretical properties is given in [6]; intuitively, these are fragments of RDF graphs composed by a starting triple and, in the case a blank node is involved, all the triples that involve such a blank node. The procedure is executed iteratively on each triple collected. In this way the MSG border is never a blank node, but always a URI or a literal. From the MSG definition, it can be formally derived [6] that a generic RDF graph can be unequivocally decomposed into a set of MSGs and that, given an RDF graph and one of its statements, the statement is included in one and only one MSG. Furthermore MSGs can be expressed in the form of a canonical string, for example using the algorithm described in [8]. By hashing such a string we obtain a unique, content based, identifier for the MSG.

4.2. Information authorship

As DBin deals with potentially large and unregulated communities, it is important to track the authorship of exchanged information. As the system is based on high replication of metadata, we cannot base on provenance tracking as each user shares RDF data from other sources (e.g., local RDF files); (b) to publish a topic based filtered sub-set of the overall local dataset, that is all the information owned at a peer which adhere to the GUED of a specific topic group.

In both cases one can decide to publish his/her own view on the data, that is filtering the overall knowledge based on local trust policies and revocations (discussed in the next section). When an RDF Dump is being published periodically by a community leader (power user), it turns out to be a way of providing an official web accessible representation of the knowledge that is being incrementally created within the P2P community. Each RDFGrowth server can be queried via HTTP to retrieve the list of the available communities and for each of them a URL can be specified where the official dump is stored.

4. Information filtering, revision and publication

4.1. MSGs

Before describing DBin's support for information filtering and revision, let us introduce the concept of Minimum Self-contained Graph (MSG). The RDFN of a resource (introduced in Section 3.2) can be decomposed in smaller pieces, called MSGs, which represents the minimum amount of RDF knowledge that can be exchanged in RDFGrowth. A formal definition of MSGs and their theoretical properties is given in [6]; intuitively, these are fragments of RDF graphs composed by a starting triple and, in the case a blank node is involved, all the triples that involve such a blank node. The procedure is executed iteratively on each triple collected. In this way the MSG border is never a blank node, but always a URI or a literal. From the MSG definition, it can be formally derived [6] that a generic RDF graph can be unequivocally decomposed into a set of MSGs and that, given an RDF graph and one of its statements, the statement is included in one and only one MSG. Furthermore MSGs can be expressed in the form of a canonical string, for example using the algorithm described in [8]. By hashing such a string we obtain a unique, content based, identifier for the MSG.

4.2. Information authorship

As DBin deals with potentially large and unregulated communities, it is important to track the authorship of exchanged information. As the system is based on high replication of metadata, we cannot base on provenance tracking as each user shares RDF data from other sources (e.g., local RDF files); (b) to publish a topic based filtered sub-set of the overall local dataset, that is all the information owned at a peer which adhere to the GUED of a specific topic group.

In both cases one can decide to publish his/her own view on the data, that is filtering the overall knowledge based on local trust policies and revocations (discussed in the next section). When an RDF Dump is being published periodically by a community leader (power user), it turns out to be a way of providing an official web accessible representation of the knowledge that is being incrementally created within the P2P community. Each RDFGrowth server can be queried via HTTP to retrieve the list of the available communities and for each of them a URL can be specified where the official dump is stored.

4. Information filtering, revision and publication

4.1. MSGs

Before describing DBin's support for information filtering and revision, let us introduce the concept of Minimum Self-contained Graph (MSG). The RDFN of a resource (introduced in Section 3.2) can be decomposed in smaller pieces, called MSGs, which represents the minimum amount of RDF knowledge that can be exchanged in RDFGrowth. A formal definition of MSGs and their theoretical properties is given in [6]; intuitively, these are fragments of RDF graphs composed by a starting triple and, in the case a blank node is involved, all the triples that involve such a blank node. The procedure is executed iteratively on each triple collected. In this way the MSG border is never a blank node, but always a URI or a literal. From the MSG definition, it can be formally derived [6] that a generic RDF graph can be unequivocally decomposed into a set of MSGs and that, given an RDF graph and one of its statements, the statement is included in one and only one MSG. Furthermore MSGs can be expressed in the form of a canonical string, for example using the algorithm described in [8]. By hashing such a string we obtain a unique, content based, identifier for the MSG.

4.2. Information authorship

As DBin deals with potentially large and unregulated communities, it is important to track the authorship of exchanged information. As the system is based on high replication of metadata, we cannot base on provenance tracking as each user shares RDF data from other sources (e.g., local RDF files); (b) to publish a topic based filtered sub-set of the overall local dataset, that is all the information owned at a peer which adhere to the GUED of a specific topic group.

In both cases one can decide to publish his/her own view on the data, that is filtering the overall knowledge based on local trust policies and revocations (discussed in the next section). When an RDF Dump is being published periodically by a community leader (power user), it turns out to be a way of providing an official web accessible representation of the knowledge that is being incrementally created within the P2P community. Each RDFGrowth server can be queried via HTTP to retrieve the list of the available communities and for each of them a URL can be specified where the official dump is stored.

4. Information filtering, revision and publication

4.1. MSGs

Before describing DBin's support for information filtering and revision, let us introduce the concept of Minimum Self-contained Graph (MSG). The RDFN of a resource (introduced in Section 3.2) can be decomposed in smaller pieces, called MSGs, which represents the minimum amount of RDF knowledge that can be exchanged in RDFGrowth. A formal definition of MSGs and their theoretical properties is given in [6]; intuitively, these are fragments of RDF graphs composed by a starting triple and, in the case a blank node is involved, all the triples that involve such a blank node. The procedure is executed iteratively on each triple collected. In this way the MSG border is never a blank node, but always a URI or a literal. From the MSG definition, it can be formally derived [6] that a generic RDF graph can be unequivocally decomposed into a set of MSGs and that, given an RDF graph and one of its statements, the statement is included in one and only one MSG. Furthermore MSGs can be expressed in the form of a canonical string, for example using the algorithm described in [8]. By hashing such a string we obtain a unique, content based, identifier for the MSG.

4.2. Information authorship

As DBin deals with potentially large and unregulated communities, it is important to track the authorship of exchanged information. As the system is based on high replication of metadata, we cannot base on provenance tracking as each user shares RDF data from other sources (e.g., local RDF files); (b) to publish a topic based filtered sub-set of the overall local dataset, that is all the information owned at a peer which adhere to the GUED of a specific topic group.
user is required to set up an identity, made by a user identifier (e.g., a mailto URL), and a pair of private and public PGP keys is automatically generated. The private key is securely stored locally while the public one is uploaded and made available in a web space. Each time the user adds information, that is a number of MSGs, to the local RDF graph, the private key is used to digitally sign each one of these MSGs (expressed in canonical form). This methodology, described in detail in [6], consists in attaching to the MSG some additional data in RDF format, including the signature digest and the URL of the author’s public key. This data are then exchanged in the P2P network along with the original MSGs and can be used by other peers to track and verify the authorship of each piece of metadata they import. Multiple signature can be attached to the same MSG, also at different times.

4.3. Revocations

The knowledge created within Semantic Web Communities, as described so far, have an inherent monotonic nature: information can only be added, while removing is not possible. While user are allowed to remove information from their local RDF graphs, such information would remain in the network, if previously shared, and would still be available for community participants to download, further annotate and share. For this reason, in DBin, revoking an MSG results in adding a new piece of RDF that expresses the will to remove the MSG. The revocation is attached to the MSG itself as it happens for digital signatures, so it can be exchanged within communities and locally processed in order to hide or mark revoked data. Furthermore, as the revocation is an information itself, it will be digitally signed, so to be able to find out who revoked what and to decide, at a local level, if revocations have to be applied or rather ignored.

4.4. Local trust policies

Once the authorship of MSGs can be verified, a variety of local filtering rules can be applied at will. Such rules are always non-destructive, as information that does not match certain trust criteria can be hidden away but does not get deleted. DBin offers a set of predefined policies. By default all the data are shown except for what has been revoked by someone. It is possible, however, to see the overall knowledge, including the revoked one, or to apply only certain revocations. For example those made by the author of revoked data or by users included in a friends list. An other simple policy is to hide information authored by someone included in a local black list that can be edited while browsing data.

5. The user interface level

5.1. Related works

RDF data visualization has a central role in Semantic Web research and many approaches have been proposed so far. Some of them use graphically represent RDF in the form of a graph, like RDF Gravity5 and IsaViz [9]. In general this approach suffers from difficulties in understanding and browsing data where the dataset is very large and connected even if sub-graphs browsing facilities partially solves the problem. The graph paradigm is also used in Welkin,6 that allows to view the graph as clusters of similar resources. An interesting browsing paradigm, called “faceted browsing” is implemented by Longwell7 and SWED.8 It consists in iteratively restricting the selection based on properties values. It is often useful to have templates for deciding, for each kind of resources, which properties have to be shown, in which order and so on. This is done in the RDF Template Language9 and in [10], where they are expressed in RDF themselves by using an ad-hoc ontology. Many applications, among which IsaViz, Longwell and Arago [11], adopted such an ontology (Fresnel). However, Fresnel does not face issues as browsing, discovering and editing RDF. With Brainlets we want to go a step forward: to configure a complete Semantic Web application with custom knowledge discovery, navigation and editing capabilities. The choice of RDF as a GUI configuration language allows such configurations to be exchanged and reused across applications. However, in the Brainlet framework, we decided to use an XML-based language, as it is easier to be understood and written. Other approaches to RDF data presentation are discussed in [12], each one having its pro and cons, demonstrating that user interface issues have, in general, no single solution. DBin addresses this by the Brainlet framework, which enables mashing up of different visualization paradigms.

5.2. Brainlets configuration

To create a Brainlet one needs first to select appropriate ontologies to represent the domain of interest, these are usually included and shipped in the Brainlet itself, although they could be placed on the Web. In the case of the ESWC Brainlet the ESWC2006 Conference Ontology10 and the FOAF11 ontology has been used.

5.2.1. View parts

A Brainlet is composed by a set of view parts, which can be invoked (choosing from a set of predefined ones), properly configured and positioned at will.

Usually, each view part takes a resource as a main “focus” and shows distinct aspects of the same RDF knowledge around this resource. For example, the Properties view in Fig. 4 (on the right) shows the outgoing (orange arrows) and incoming (green arrows) RDF statements surrounding the selected RDF node while the Gallery view displays pictures related to a resource.

Selection flows are also scripted at this point; it is possible to establish the precise cause effect chain by which selecting an icon on a view will cause other views to change. This is done by creating custom aggregation points which act as selection event routers among the views. Each view is identified by a unique (within the Brainlet) ID, as multiple views of the same type can be declared within the same Brainlet.

5.2.2. Resources navigation

The “Navigator” provides resource browsing based on a flexible and dynamic tree structure. Such an approach can be seen to scale very well with respect to the number of resources. In Fig. 4 (left part) we show a possible alternative configuration for the ESWC Brainlet Navigator view (the one illustrated in Fig. 1 is more rich and complex), which deals with navigation of conference events. In such a navigator, generated by the following XML configuration,

---

two distinct branches coexist: the first one groups events by type, the second by the room they take place in.

```xml
<view type="navigstor" id="nav1" title="ESWC Navigator" relative="tool">
  <Branch name="Events by type" root="http://www.eswc2006.org/ontology#Event">
    <Child query="SELECT X FROM {X} serql:directSubClassOf {$parent}">
    </Child>
  </Branch>
  <Branch name="Events by location" root="http://www.eswc2006.org/ontology#Place">
    <Child query="SELECT X FROM {X} serql:directType {$parent} />
    </Child>
  </Branch>
</view>
```

5.2.3. Precooked queries

Within a specific domain there are often some queries that are frequently used to fulfill relevant use cases. The precooked queries facility gives Brainlet creators the ability to provide such “fill in the blanks” queries to end users. In the ESWC Brainlet, for example, user can search all papers written by someone whose name is like X, where X is a string parameter to be filled in.

```xml
<Child query="SELECT X FROM {X} eswc:hasLocation {$parent} />
```

5.2.4. URI wizards

As in our system all the users are entitled to add new concepts to the shared knowledge base, they should be prevented from choosing different URIs for the very same object. For this purpose we introduced the URI Wizard concept. URI Wizards define procedures for guiding users in assigning identifiers to newly created instances. Different procedures can be associated to different types of resources. A procedure for adding a new paper could be, for example, to point the DBLP search engine to find the article and to use the link to the document as an identifier, or even to let the user type the title and to create a URI hashing it.

The Brainlet framework also includes other modules, which we do not discuss here for lack of space, that can be configured to provide interesting capabilities, such as the Web browser and the GoogleMaps (that is shown in the screenshot of Fig. 1). Please refer to [13] for a more complete discussion.
5.3. Web service and web interface module

DBin provides both web service and web application functionalities. Web services provided by an ever running installation of DBin, connected to an ad hoc P2P group, have been successfully used to power a Semantic Web driven web site. The Web service provides both low level querying functionalities and higher level functions, typically those that are interactively accessibly either via the UI or via the DBin console.

This setting is particularly interesting as it spares the need to construct a web-based authoring environment; DBin receives the information from the P2P network and this enables multiple parties to contribute and update the information driving the web site. Such updates happen in a decentralized way, but the digital signature infrastructure enables control over what is ultimately accepted and served by the DBin installation which provides the web services.

6. User survey and validation

To address the validation of the DBin application and scenario, we conducted a user survey. The user group was composed by 20 people. Most of the users reported at least some previous contact with Semantic Web ideas or tools. Some participants were experts with clear knowledge of the field, while others were professional figures, however selected for having familiarity with advanced software (e.g., Engineers, Web designers, etc.). The recruitment happened via email announcements on public mailing lists, on related IRC channels such as SWIG and via direct mailing to selected individuals.

The survey included instructions to set up a DBin client and start the experience by browsing existing P2P channels. Moreover we suggested users to take some action that we consider important to exploit peculiar features such as for example adding new entities and annotations, establish relations and find information by browsing and using pre cooked queries. A web survey was then proposed to the users about their perceived usefulness of one such tools and of the Semantic Web Communities paradigm. We report that approximately 70% of our users answered that they perceived novel possibilities enabled by the software as opposed to 10% who said same or less possibilities and 20% who did not know. Table 1 shows the results obtained from a question about the Rich personal Semantic Web client paradigm with respect to web tools like wikis, blogs and web forums. We provided six features to be scored with six possible values: not at all, in general not, uncertain, sensibly so, very much and ‘I do not know’.

The same choice has been used for answering about the professional figures (we proposed seven categories) which the user felt could take advantage in using applications like DBin in their daily activity (Table 2). We then asked the users if they would find DBin potentially useful in applications to their job and hobbies, obtaining a 72% and 83% of clear yes values.

Table 1
Answers about the comparison between applications like DBin and collaborative web tools like wikis and blogs.

<table>
<thead>
<tr>
<th>Feature</th>
<th>N</th>
<th>I</th>
<th>U</th>
<th>S</th>
<th>V</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flexibility</td>
<td>6</td>
<td>0</td>
<td>12</td>
<td>59</td>
<td>24</td>
<td>0</td>
</tr>
<tr>
<td>Responsiveness</td>
<td>0</td>
<td>18</td>
<td>18</td>
<td>29</td>
<td>18</td>
<td>18</td>
</tr>
<tr>
<td>Interactivity</td>
<td>0</td>
<td>6</td>
<td>19</td>
<td>25</td>
<td>44</td>
<td>6</td>
</tr>
<tr>
<td>Ability to be personalized</td>
<td>0</td>
<td>12</td>
<td>41</td>
<td>6</td>
<td>24</td>
<td>18</td>
</tr>
<tr>
<td>Graphics</td>
<td>0</td>
<td>29</td>
<td>24</td>
<td>24</td>
<td>24</td>
<td>0</td>
</tr>
<tr>
<td>Clarity</td>
<td>0</td>
<td>29</td>
<td>24</td>
<td>41</td>
<td>6</td>
<td>0</td>
</tr>
</tbody>
</table>

N, not at all; I, in general not; U, uncertain; S, sensibly so; V, very much; D, I do not know. Answers in percent values.

Finally we collected future features or improvements suggestions. Results are shown in Table 3. The highest rated improvement is usability, demonstrating that the interface is still too complex and rich. We plan to perform more studies focused on usability issues. An improved web accessibility to the content is also considered very important as well as support for information trust. The results have been generally very positive, confirming that DBin is perceived as likely to provide benefit in support of technical groups and professionals while being less suited to web users with less specific interests.

It has been recently shown [14] that a group of relatively technically inexperienced users can grasp the use of a specialized Semantic Web-based tool to annotate entities of common interest (in the case of the specific study, fantasy novels and other works of art). The work observes that given the appropriate tools, there are social and practical situations in which annotating becomes a rewarding experience thus leading to a great result in the collaborative efforts. The results of our survey seem in line with such finding: users considered the DBin scenario potentially interesting to several different communities. Some of these communities already exists, yet limited to small groups: DBin is being used in cooperatively editing data powering a web portal, to cooperatively annotate bookmarked resources thanks to the del.icio.us plugin [15], and as an experimental cooperative semantic GIS tool [16].

7. Related works to the DBin project

In this paper we have been so far presenting related works in a number of previous occasions. This has been functional to the explanations of the individual infrastructures, e.g., GUI, P2P. In this section we instead discuss the previous works which can be related to the DBin project in a more general sense.

During the last years many attempts have been done in creating applications that could show the users the Semantic Web in action. Piggy Bank [17], is an MIT project which consists in a Firefox extension giving the browser the ability to retain across sessions RDF data, which is collected as the user browsers the Web. RDF information is either stored when explicitly available on the site or scraped trough the use of site-specific adapters which create structured content out of the HTML. Piggy bank integration with the common web browsing paradigm is certainly something DBin,
will have to look into as a feature for future releases. On the other hand, Semantic Web Communities HTML presence pages as highlighted in Section 3.4 can themselves become interesting locations to visit for Piggy Bank users, as these provide polished RDF sources for them to import and browse.

Among the applications that focuses on information aggregation and discovery on the Semantic Web, Disco13 is a novel and simple web application based on the linked data paradigm that enables surfing across published interlinked RDF graphs. Haystack [18] is an advanced configurable desktop data manager based on Semantic Web technologies and supports a python-like language (Adenine [19]) for manipulating RDF and script GUI level functionalities. Unlike Haystack, DBin fosters the creation of domain specific applications by simple XML configuration and focuses specifically on enabling collaborative knowledge creation. Finally, this paper considerably extends and revises the previous overview works [20,21] by providing for the first time accounts of the communities web presence tools and scenario validation.

8. Conclusions

In this paper we presented a comprehensive overview and a first evaluation of the Semantic Web Community scenario enabled by the DBin platform.

DBin uses P2P to exchange the knowledge collectively authored by the communities of users via topic specific user interfaces named Brainlets. Brainlets and P2P channels can be configured, deployed by domain experts and then easily discovered, installed and used by people. The knowledge collaboratively created within communities can then be made available on the Web, enabling external reuse of data. Content and annotations produced by the users can flow across groups to reach people that have interest in that specific entity (URI), no matter where it was originally posted.

While there are multiple solutions in DBin that are interesting per se, we believe the most important aspect is the holistic integration of all the components under a single “scenario philosophy”, in other words, the ability to enable real Internet users, for the first time, to “take a peek” from the shoulders of the Semantic Web tower.

To enable this, we propose pragmatic solutions suggested by the scenario itself. For example, we leverage the existence of groups and their “maintainer” enabling them to provide Brainlets. These, in turn, transparently provide the users with pragmatic answers to notable Semantic Web open research issues such as trust, ontology management, user interface, etc. While such solutions can hardly be thought of as satisfying in respect to all scenarios and possible user needs, we believe that they might “go a long way” in a large number of use cases. Such ideas seem furthermore validated by the survey we conducted and by the other forms of validation and feedback we had so far.

References
