SEA: A Lightweight and Extensible Semantic Exchange Architecture

Thomas Franz, Carsten Saathoff, Olaf Görlitz, Christoph Ringelstein, Steffen Staab
ISWeb, University of Koblenz, Germany
http://isweb.uni-koblenz.de
{franz,saathoff,goerlitz,cringel,staab}@uni-koblenz.de

ABSTRACT

With novel centralized information management systems, users benefit from collective data organization. Centralization, however, also imposes privacy and availability constraints. With this work we present SEA, a semantic exchange architecture for distributed information management that employs Peer-to-Peer networking, social information management, and semantic web technologies. SEA addresses privacy and availability by storing data locally while utilizing it globally. The benefits of collective information tagging as offered by prominent centralized services are available through anonymous tag distribution in the Peer-to-Peer network. More fine grained access control allows to share data both publicly with all connected peers and privately within networks of trust. In this paper, we present the initial architecture and early implementation details of the prototype.

Keywords

online collaboration, folksonomy, web 2.0, p2p, semantic desktop

1. INTRODUCTION

In this paper, we introduce SEA, a semantic exchange architecture that enables autonomy of data management and freedom of data organization. Popular systems such as Delicious[1] provide freedom of data organization in the form of tagging. They allow to share information and to exploit the enriched data basis stemming from the collective organization effort. However, they prevent full individual control of the data as they require to upload personal information onto a centralized repository that is controlled by some third-party provider. Theoretically, any provider of centralized information mangement systems can analyze the behavior of users and thus potentially violate ones privacy. Furthermore, handing personal data over to a third-party requires permanent availability of the used services. On the other hand, local storage guarantees full autonomous control over ones data, however, lacks potential advantages gained by collective data management.

Using SEA, data is stored on privately controlled storage devices (e.g. personal computers) to leverage autonomy of data management. Consequently, privacy and availability issues of handing over personal data to a third party are eliminated. SEA introduces tagging to establish freedom in data organization as information tagging enables the representation of different perspectives onto data. Taggings are further utilized to easily define access restrictions, e.g. allowing to state that everything tagged with *car* should be public. Similarly as in prominent collective information management systems, taggings of multiple users are shared to improve information search, e.g. searching for information tagged with *car* will result in suggestions for related tags such as *vehicle*.

Accordingly, SEA combines the advantages of collective information sharing and local data storage without imposing the usual drawbacks of each of these models.

In Section 2, we continue with two different application scenarios that support the demand for functionality as provided by SEA, and that are referenced throughout this paper to ease readability and comprehension. In Section 3, the scenarios ground the discussion of conventional solutions and a requirements analysis, while we utilize extracts from these scenarios in Section 4 to demonstrate the architecture and applicability of SEA. We present implementational details of the currently developed prototype of SEA in Section 5 and contrast it with related work in Section 6. An outlook and a conclusion of the work presented in this paper are given in Section 7 and 8.

2. USE CASES

2.1 Scientific Project Collaboration

Some research institutions, A, B, and C, collaborate on a project called X-Media. In such a scenario, a closed user group, i.e. researchers from institutions participating in the project, needs to securely share, organize and access project related documents. Project members share papers about related work and collectively write project deliverables that document the project status.

Organize. In order to individually organize the collectively managed data, researchers tag the documents they are working on and want to share. Initially, the project team has agreed on a few common tags that are used by all project members, such as *wp4* to denote documents related to workpackage four, and *deliverable* to indicate documents that are deliverables.

Share. For simplicity, access rights for collaborators are also based on tags, e.g. researcher Carsten defines everything tagged with *X-Media* to be accessible for all X-Media

[1] `http://del.icio.us`

WWW2006, May 22–26, 2006, Edinburgh, UK.
.

members. Such members, however, are not allowed to change the access rights of that information and thus Carsten is still in control of it.

Retrieve. If the project coordinator wants to summarize all deliverables, he simply searches for all documents tagged with *deliverable*. Unlike in hierarchical file management where the coordinator would have to check all workpackage directories, the flexibility of tagging ensures that the coordinator automatically retrieves all relevant documents.

Track Changes. To track changes on the deliverable Carsten is working on with Olaf and Thomas, he subscribes to the corresponding tag *D.4.8*. If Olaf or Thomas have changed the document, Carsten will be notified about changes of documents tagged with *D.4.8*.

2.2 Sharing Images

As an additional use case, we consider the scenario of image sharing. Users may want to share some pictures only with their friends while other pictures should be publicly available.

Olaf has been on a hiking trip with Christoph and he wants to share beautiful landscape pictures publicly, while he wants to share those depicting him and Christoph only with his friends to ensure privacy.

Multi-Perspective Image Management. While Olaf does photography as a hobby, he regularly takes pictures and handles large amounts of pictures. He utilizes tags in order to organize his pictures towards different perspectives, e.g. he classifies pictures by the persons depicted on them but also by the event and the time they have been taken.

The different perspectives he uses for the classification of images allows Olaf to easily find all pictures where Christoph is depicted, but also to find all pictures taken at hiking trips in the alps during last summer.

Usability. The employment of a P2P system decreases Olaf's effort of sharing his photographs as the effort of uploading large numbers of pictures to a central server to share them is ommited. Furthermore, if Olaf modifies some pictures, the changes he made are immediately available to others without needing to upload them.

Olaf organizes his pictures once for both local usage and global sharing, whether he is online or offline.

Simple Access Management. Olaf can easily share pictures with friends by, for instance, defining that all pictures tagged with *alps*, *hiking*, *2005* are visible to Christoph. To make pictures publicly available, he tags those as *public*.

3. REQUIREMENTS

In the following, we check conventional solutions with respect to the previous use cases and derive requirements for support of such scenarios.

Information Security. Due to confidentiality of most of the project work, partners need to securely share their project contributions. The usage of centralized storage services offered by third parties is often prohibited in project contracts as it imposes additional security risks. A P2P system enables each user to take full control of the shared data, in particular no third party needs to be involved.

Cost Reduction. Even if one of the project partners would setup a dedicated document management system to solve the security constraint, this would result in additional costs for both setup and system administration. In particular in non-professional scenarios like the previous image sharing use case, the setup and maintenance of a dedicated server is not feasible. Instead, the feasibility of installing a P2P client has been proven by very large numbers of client installations of P2P file sharing applications such as Emule[2], et cetera.

Desktop Integration. Sharing documents with a central repository requires additional user effort in the form of explicit uploads and downloads. In general, additional user effort does not encourage users to employ the system except its use is enforced so that not all benefits of collective data management can be exploited. For instance, while knowledge about related work papers would benefit a project, such papers are hardly shared if the effort of uploading such papers is too high. P2P systems can be well integrated into the user's desktop enabling users to effortlessly share their data.

Simplified Change Management. Another drawback of centralized systems is the redundant storage of the data on both the central document repository and local computers. While redundant data storage is generally not a problem of storage costs or harddisk space anymore, it results in error-prone and difficult change management. Given Carsten has worked on some document but not updated the centrally stored initial version, this change cannot be tracked by his colleagues. Only when he explicitely uploads the changed document, his changes become effective. Redundant data storage also imposes redundant organization, e.g. if shared data needs to be utilized offline it has to be organized both locally and on the central sharing platform.

Sophisticated and Easy-to-Use Resource Management. Hierarchical data organization lacks the ability to represent the same data in different contexts as indicated by the image sharing use case. More freedom in the organization of ones data is needed to cope with large amounts of information. To minimize user effort, seamless integration of access control is another requirement. In the previous use cases, we indicate the simplicity gained by using taggings for organizing information as well as defining access control as users only need to learn one paradigm in order to do two different tasks.

4. ARCHITECTURE

Along the use cases of Section 2 and the previously derived requirements, we present SEA, the Semantic Exchange Architecture. As illustrated in Figure 1, SEA constitutes a network of decentralized repositories in which information is collectively organized using non-hierarchical classification. A repository runs on a local desktop and provides a portion of the local information as well as organizational meta data about such information in the form of taggings. Taggings

[2]`http://www.emule-project.net`

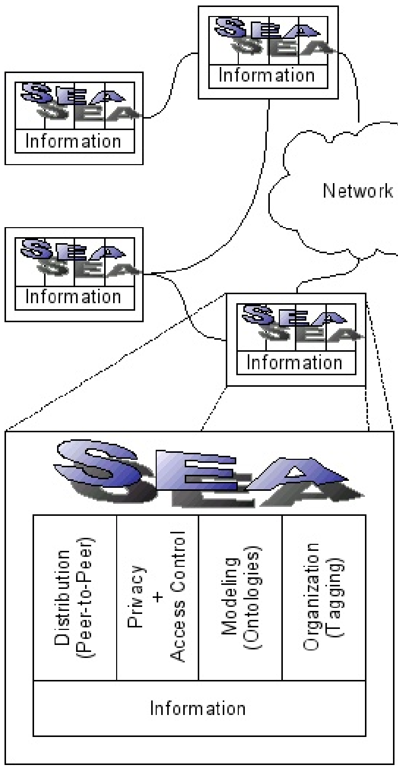


Figure 1: Features of SEA

are used to enable local as well as networked access and exchange of the information distributed over multiple peers.

4.1 Data Organization

Conventional hierarchical data organization is unable to represent different perspectives onto some data. SEA employs tagging as a mechanism to allow more flexible data management and retrieval. Taggings are user defined catchwords that can be associated with information objects. They provide a high degree of freedom to organize information since no predefined relations between tags exist as for instance in taxonomies. An information object can be associated (tagged) with multiple tags to represent different perspectives onto the data.

4.2 Data Model

SEA employs ontologies as meta models (micro models[3]) for the managed data. We argue that building novel systems on ontologies from the beginning leverages integration of knowledge, reasoning and further improvements later on. Figure 2 illustrates how we combined different ontologies that are further explained in the following. Interoperability and extensibility, however, are the main and intial reasons for using ontologies in SEA.

Tagging Ontology. The Tagging Ontology is based on Richard Newmans proposal [7] for an ontological micromodel that represents the tagging of ressources. The class *Tagging* models the association of an *InformationObject* from our information resource ontology to a *Tag*. Furthermore, validity and the creator of the tagging are modeled as well as properties used to relate tags.

[3]`http://esw.w3.org/topic/MicroModels`

Information Resource Ontology. This ontology contains the class *InformationObject* that is accompanied by several common properties for such resources to represent their physical location, name, and size.

Access Control Ontology. Access control is modeled along the definition of the following section. An access right represents the user, access mode tuple consisting of a reference to the user and the access mode.

4.3 Access Control

Tags are utilized to organize information but also to define access rights for the shared information. This approach is easy-to-use by users as only the system of tagging needs to be learned in order to both flexibly organize information and maintain access control. The implementation of access control in SEA is based on querying and modifying given access data that follows the ontological model illustrated in Figure 2. The implementation is characterized by a few rules that we explain in the following.

4.3.1 Access Modes

Access modes (cf. Figure 2), correspond to possible actions on information objects, such as reading, or modifying.

Rule 1. Access modes are positiv.

Rule 1 determines that access modes state permissions rather than prohibitions. Prohibitions are implicit since by default, SEA prohibits any access unless an appropriate (positive) access right is specified. Initially, we consider one access mode, m_{read}, which defines read access to an information object.

4.3.2 Ownership

Access control in SEA is furthermore based on ownership as defined by the following rules:

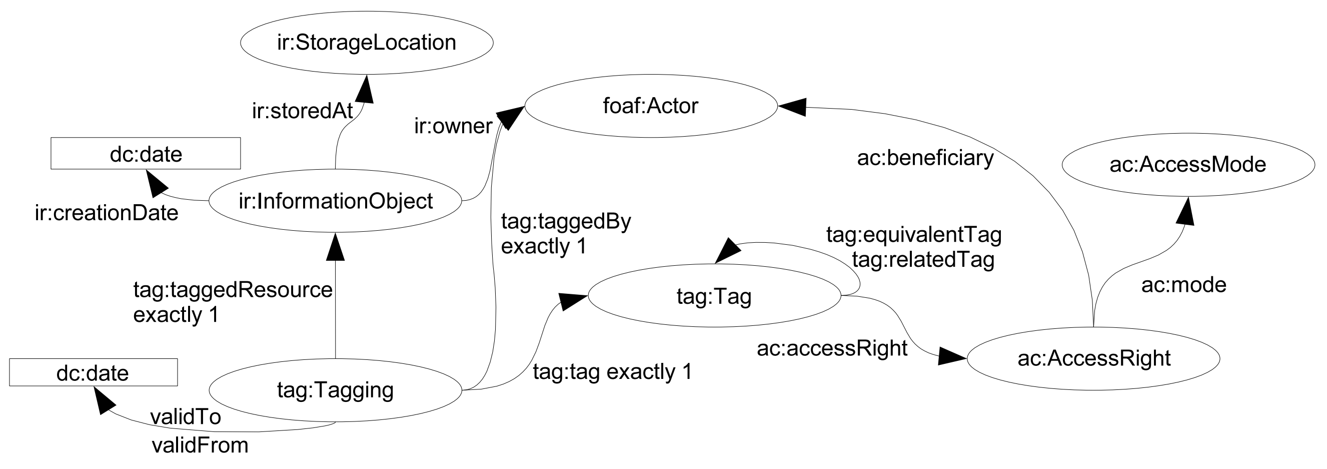
Rule 2. Access rights associated to tags by some user do only apply to information objects owned by that user.

Rule 3. Only access rights defined by the owner of an information object are effective.

The following example of access right propagation illustrates the application of rule 2 and 3: Some user X is the owner of information object o_1 and o_2. X has tagged o_1 with tag_1 and associates access rights to tag_1. Accordingly, the access rights associated with tag_1 hold for all information objects owned by X and tagged with tag_1. If now another user Y tags o_2 with tag_1, the access rights associated with tag_1 do not apply since Y is not the owner of o_2. Now if Y tags an information object o_3 he ownes with tag_1, then the access rights defined by user X for tag_1 also do not apply to o_3.

4.3.3 Access Right Summation

Information objects can be tagged with multiple tags, which could potentially define different access rights, or access rights for different users. As access modes are positive (rule 1), no conflicting access rights can occur so that checking whether some user can execute a certain operation on some object means to find an appropriate access right for that user. Rule 4 expresses this behavior.



Namespaces:
dc=http://purl.org/dc/elements/1.1/ foaf=http://xmlns.com/foaf/0.1/
ir=http://isweb.uni-koblenz.de/ontology/IR ac=http://isweb.uni-koblenz.de/ontology/AC tag=http://isweb.uni-koblenz.de/ontology/TAG

Figure 2: Standard Ontologies in SEA

Rule 4. Effective access rights of an information object are based on the summation of the access rights defined by each tag associated with the information object.

The following example explains how rule 4 is applied: Given an information object o that is tagged by its owner with tag_1 and tag_2. The owner also associated access right (X, m_{read}) with tag_1 stating that user X can read objects tagged with tag_1. Similarly, the owner associated tag_2 with (Y, m_{read}). Applying rule 4 yields to the result that X and Y can read o. Note that this only applies if rule 3 is valid, i.e. the user that associated the access rights to tag_1 and tag_2 is the owner of o.

4.4 Data Distribution

To explain which information needs to be distributed to fulfill the control as well as usability requirements, we start by enumerating the following common retrieval scenarios:

1. Find all information objects tagged with tag k.
2. Get all tags associated with information object o.
3. Get all tags co-occurring with tag k.

In order to ensure privacy of users, only object identifiers and tags (including their relations) but no ownership and location information are distributed in the network. Such information allows to identify the information as illustrated by the three previous cases. However, it is not possible to retrieve an information object as location information is missing. We address the contradicting demands of privacy and retrievability by distinguishing between two different cases that each have different privacy demands and implement two different options for the retrieval of actual information objects for each such case.

Publicly Shared Data If a user decides to publicly share an information object, its location information is distributed so that the information provider can be directly contacted.

Protected Data In order to share protected (not publicly shared) data, SEA uses a different retrieval mechanism that is based on the consultation of a finite list of peers for which the corresponding user is known, similar to a buddy list in instant messaging software. We argue that this solution is sufficient to find the information object as the owner of the information is expected to know those users for which he grants access and vice versa. If one wants to grant access to users one does not know, public access can be granted.

4.4.1 Distribution Mechanism

SEA utilizes a distributed hashtable (DHT) approach to distribute information in the network. In short, DHTs partition the distributed data among a set of peers and keep information about which peer stores which portion. This approach guarantees that all such distributed information can be found and further, that it can be found within log_n hops. As indicated by the name, DHTs are interfaced like common hashtables and store key-value pairs, where a key is required to access associated values.

We already argued which information needs to be distributed and now explain how it is distributed using the DHTs. Four hashtables are employed to efficiently represent the needed information. Table tab_o contains for each information object id the set of all tags associated with that object and thus allows to retrieve all tags associated to an information object. Another table tab_k allows for querying in the opposite direction, i.e. retrieval of all object ids for a tag. While the computation of tag correlations would be possible by using only tab_o and tab_k, it would require multiple request and thus increase network load. We argue that memory and space costs are lower than those for network bandwith and model tag correlations by an additional DHT tab_{co} that maps each tag k to the set of tags that occur together with k. As we distribute location information for those information objects that are public, that information is contained by the table tab_l that maintains for each public information object a set of locations where it is available.

5. IMPLEMENTATION

In the following we present the implementation of SEA and its main components, i.e. the data repository, SEA core, peer manager and the DHT module as depicted in figure 3. Applications using SEA are not part of the architecture. They use the platform independent REST interface to request and modify resources. Internally these operations are propagated to the repository, the peers in the trusted network or the whole network based on the specific action.

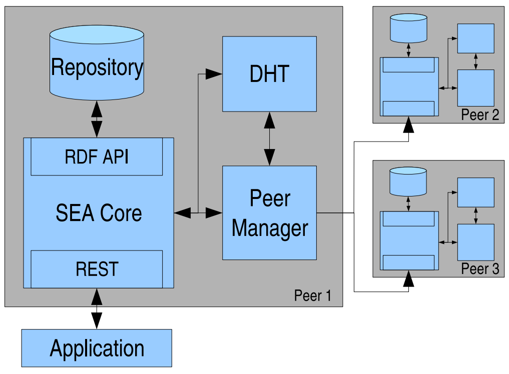


Figure 3: Implementation of SEA

5.1 Data Repository

All metadata available in our system will be stored in a RDF repository. RDF [5] is the proposed standard for representation of metadata by the W3C and forms also the basis for OWL, the Web Ontology Language [6]. RDF offers great felixibility, employing a graph based and schema less model, which ensures easy integration of various kinds of metadata later on. Since used micromodels are defined by the ontology representation language OWL, we can also integrate them easily in the repository.

Data retrieval in SEA is based on the standardized query language for RDF graphs, SPARQL [9]. For modifications of RDF graphs, no such standard does exists so that modifications are implemented through proprietary APIs that are usually provided by RDF stores. In order to implement the SEA Core module independent of proprietary RDF APIs, a wrapper is implemented that provides a concise interface between SEA Core and any RDF store. The wrapper provides a SPARQL interface for queries and a generic, triple-based interface for repository modifications.

5.2 SEA Core

The SEA Core module constitutes application specific functionality. For applications built on SEA, the core module provides a REST API [3] that offers a number of central services which it implements by utilizing attached SEA components such as the RDF store. Currently we consider the following operations:

- Requesting resources with a specific tag, tags of a resource with a specific identifier and tags that are related to another tag
- modifying the tags of resources
- authentication of trusted users

Following the REST architectural style, the services are offered via a dedicated URL and requests are sent using HTTP for communication. XML is employed to represent the results, which are returned as part of the HTTP response. This API style offers great flexibility, simple deployment of HTML and browser based applications and also platform independence. A number of frameworks can be used to implement the REST Api, e.g. lightweight solutions, such as TurboGears[4] for Python or more complex set-ups using e.g. Apache Tomcat[5].

5.2.1 Requests

If resources with a specific tag are requested, the SEA Core distributes the request to the local repository, the Peer Manager, and the DHT implementation. The Peer Manager then sends the request to all trusted peers and the DHT implementation also queries the whole network. The core then aggregates the different results and returns one result set back to the application.

A request for the tags of a specific resource is just distributed via the DHT implementation to the whole network. Since within the DHT's this information is directly available, neither the local repository nor the trusted peers have to be queried additionaly. In case of requesting related tags, again the query is just propagated to the DHT implementation, and again the results are returned with just one query.

5.2.2 Modification of Taggings

Tags can currently only be modified locally. In this case only the local repository has to be updated accordingly and no propagation to any other peer is involved. However, this might change in a later version of the framework.

5.2.3 User Authentication

In order to authenticate a trusted peer, we will employ a signature based approach, using mutual authentication. The authentication process is handled by the peer manager and respective request are just forwarded by the SEA Core. Although such authentication is important in our scenario, respective implementations are available and can be plugged in later on. Therefore, we just include a dummy authentication in the first prototype.

5.3 Peer Manager

The peer manager module is responsible for all operations involving the communication with other peers, i.e. rendevous, authentication and request forwarding. Additionally, it provides peer information for the DHT implementation. Communication with the other peers is done via the common interface exposed by every peer. Results of forwarded requests are handed back to the SEA Core for further processing.

5.4 DHT Implementation

The current prototype of SEA employs the Bamboo[6] distributed hashtable implementation to efficiently store tagging information so that it is available for all peers in the network. According to Section 4.4 each peer is responsible for storing a portion of the four hashtables required to allow a comprehensive evaluation of tag relations.

[4]`http://www.turbogears.org/`
[5]`http://tomcat.apache.org/`
[6]`http://bamboo-dht.org`

5.5 Current Implementation Status

A DHT component has been implemented as well as a component for tagging files on the file system. The latter represents one portion of the SEA Core component, is based on the ontologies mentioned in Section 4.2, and utilizes the Sesame2[7] RDF repository. Both components have been connected so that taggings can be distributed and retrieved between peers.

Current efforts are concentrating on completing the SEA Core implementation and the development of the Peer Manager. We refer the reader to the SEA website[8] for updated information.

6. RELATED WORK

A part of the features implemented by SEA have been coined in [2] as the Networked Semantic Desktop. SEA deals with issues of the following three main research areas:

Semantic Desktops. The Gnowsis [10] Semantic Desktop is a system based on an RDF repository for storing meta data about arbitrary desktop data such as files, emails, or contacts. A plugin mechanism allows to connect arbitrary desktop applications as a data provider for Gnowsis.

Aduna's *Metadata Server*[9] is a server component that crawls different information sources such as the local file system and email servers to extract meta data and stores it in a triple store. Enhanced search tools (Autofocus and the web-based Spectacle) are based on the server component and allow to query within the data and visualize the query results.

While SEA also allows to organize and interlink arbitrary desktop information, it enhances semantic desktops by decentralized information sharing.

P2P Systems. The Bibster system [4] is a P2P client for sharing bibliographic data that is represented as RDF. Bibster also allows meta data editing and ensures that locally supplemented meta data benefits participants of the network. Semantic query routing algorithms based on the information maintained by single peers has also been developed by contributors to Bibster [11] in order to efficiently distribute queries within the P2P network.

RDFPeers [1] establishes a distributed RDF repository. Based on hashed RDF data (subjects, objects, and predicates), queries for RDF content are efficiently routed and guaranteed to return complete results.

SEA adds access and privacy control to P2P networking and enables sophisticated collective data organization independent of the application domain.

Centralized Sharing and Editing. Confoto[10][8] allows to share fotos similarly as with prominent sites as mentioned in the introduction. Additionally, Confoto allows to collectively annotate the shared information using domain specific properties based on predefined domain schemata. As a centralized system, it is accessible via a web frontend and provides import functionalities to add arbitrary RDF data to the repository.

With SEA, we address security concerns about centralized information sharing. SEA provides the same functionality as in collective information sharing systems while guaranteeing full control over ones data.

7. OUTLOOK

In the follwing we present some open issues and desirable features that we plan to work on after the prototype is finished. First of all, the performance of the current DHT approach is likely to degrade with an increasing management overhead imposed by a growing number of taggings stored in the system. Therefore we will research alternative approaches for distributing information in peer-to-peer networks, specifically taking our requirements into account.

Our current implementation only considers read access to resources. The first use case, however, illustrated that also collaborative work on resources is of great interest. Apparently this creates additional requirements for the framework which have to be carefully checked to gurantee the consistency between different versions of the same resource. Such a colleborative work on resources will also require novel and sophisticated user interfaces so that changes made at other peers are reflected accordingly.

8. CONCLUSION

In this paper, we started with the identification of requirements and shortcomings of today's prominent information sharing platforms to present a generic framework (SEA) that is applicable to various item domains and contains multiple novel features that are important in the setting of collective, distributed, information organization. SEA's open architecture offers easy adoption, extension, and development as it is based on acknowledged standards that are well supported by programming libraries and development tools.

9. ACKNOWLEDGMENTS

This work has been supported by the european projects Knowledge Sharing and Reuse across Media (X-Media), and Adaptive Services Grid (ASG), both funded by the Information Society Technologies (IST) 6th Framework Programme.

10. REFERENCES

[1] M. Cai and M. R. Frank. Rdfpeers: a scalable distributed rdf repository based on a structured peer-to-peer network. In *WWW*, pages 650–657, 2004.

[2] S. Decker and M. R. Frank. The Networked Semantic Desktop. In *WWW Workshop On Application Design, Development and Implementation Issues in the Semantic Web*, 2004.

[3] R. T. Fielding. *Architectural Styles and the Design of Network-based Software Architectures.* PhD thesis, UNIVERSITY OF CALIFORNIA, IRVINE, 2000.

[4] P. Haase, B. Schnizler, J. Broekstra, M. Ehrig, F. van Harmelen, M. Menken, P. Mika, M. Plechawski, P. Pyszlak, R. Siebes, S. Staab, and C. Tempich. Bibster - a semantics-based bibliographic Peer-to-Peer system. *J. Web Sem.*, 2(1):99–103, 2004.

[5] F. Manola and E. Miller. Rdf primer. Web, February 2004. http://www.w3.org/TR/rdf-primer/.

[7] http://openrdf.org

[8] http://isweb.uni-koblenz.de/Research/sea

[9] http://www.aduna.biz

[10] http://www.confoto.org

[6] D. L. McGuinness and F. van Harmelen. Owl - web ontology language, 2004. http://www.w3.org/2004/OWL/.

[7] R. Newman. Tag ontology design. http://www.holygoat.co.uk/projects/tags/.

[8] B. Nowack. CONFOTO: A Semantic Browsing and Annotation Service for Conference Photos. In *International Semantic Web Conference*, pages 1067–1070, 2005.

[9] E. Prud'hommeaux and A. Seaborne. Sparql query language for rdf, 2005. http://www.w3.org/TR/rdf-sparql-query/.

[10] L. Sauermann. The Gnowsis Semantic Desktop For Information Integration. In *WM 2005: Professional Knowledge Management*, pages 39–42, 2005.

[11] C. Tempich, S. Staab, and A. Wranik. Remindin': semantic query routing in peer-to-peer networks based on social metaphors. In *WWW*, pages 640–649, 2004.