# Spontaneous Collaboration via Browsing of Semantic Data on Mobile Devices

Ora Lassila[1], Deepali Khushraj[1], and Ralph R. Swick[2]

[1] Nokia Research Center, Cambridge, MA, USA
[2] Massachusetts Institute of Technology, Cambridge, MA, USA

**Abstract.** In this paper we present a tool for browsing "semantically connected" data that the user has collected (or subscribed to) from multiple sources. By allowing customized views of the data, our tool can be seen as an application platform for data-oriented applications. The view of the data can be customized based on the data itself (e.g., it can depend on specific datatypes); additionally, it can also vary based on the user's current "context" as well as any enforced policies (access, privacy, etc.).

Our tool can be used to form spontaneous collaborations between mobile devices and their users by providing a personalized and localized view of the integrated semantic data from multiple users. Browsing semantic data on a mobile device also adds interesting dimensions to the scenario: A mobile device has a wealth of information about its user, and it also has data available for determining the user's context. The dynamic nature of mobile environments presents certain challenges as other devices can appear or disappear without forewarning. The physical proximity of local services and actors will also offer opportunities for, say, new ways of establishing trust between systems.

We present some usage scenarios that make use of our system and the representation formalism (RDF++, an extension of RDF) that it employs.

## 1 Introduction

RDF [1, 2] is the fundamental building block of the Semantic Web [3]. It can be thought of as a modeling system using *directed, labeled graphs* that can easily be presented as hypertext; at its simplest the presentation of Semantic Web data only has to rely on the RDF metamodel, making *any* RDF data "browsable" without any special knowledge of the schema(ta) involved. More specialized *schema-aware* presentations can then be built on top of the basic solution.

In this paper we present a tool for browsing automatically constructed integrated representations of data acquired from multiple sources (e.g., multiple users). Context-aware organization (e.g., prioritization) and policy-aware filtering can be applied to provide each user with a personalized "browsable" information space. Customization and personalization of how (and what) information is

presented is particularly relevant when the user's access is via a mobile, handheld device where screen space, network bandwidth, etc. may be limited.

Spontaneous collaborations between mobile devices are facilitated in our tool by the use of Semantic Web technologies. This spontaneous collaboration comes about through the unplanned association of data and the use of common identifiers. The multiplicative effect that occurs when people put data in the Semantic Web using the same vocabularies allows our tool to uncover and present paths between resources that may not have been apparent from any subset of the data sources taken by itself. The key to such spontaneous data integration is RDF's use of the Uniform Resource Identifier as the token to label both nodes and arcs in the graph. Two independent data sources that use the same identifiers will of course join their data trivially and the tool's view of a node will reflect this join. The collaborative power of the Semantic Web comes fully into the forefront when two data sources have *not* used the same identifiers for identical nodes or identical concepts. When a third party (a third data source) is discovered that has identified correspondences between the nodes or edge types and when those correspondences are read into the tool, the presentation of the associated regions of the RDF graph will include joins of data that were not previously recognized.

More real-time collaboration between people, between devices, and between people and devices can occur when the system uses context and policy information to identify others who may usefully contribute to the current activity. On mobile devices the user's context such as proximity to other users and the user's current task (driving a vehicle, on the phone, in a meeting) are readily available, making it easy to identify potential collaborators. Additionally, trust relationships are reinforced by physical proximity between users, making mobile devices more conducive to spontaneous collaborations.

Semantic collaborations between devices in a mobile environment are very dynamic in nature as devices might join and leave the collaboration without any forewarning. Our tool provides mechanisms to easily invalidate individual data sources without having to significantly recompute the integrated view. The tool handles this by delaying reasoning until query time.

## 2 Related Work

Many tools have been constructed for searching, exploring and querying complex information spaces such as semistructured data representations, collections with rich metadata or, indeed, the Semantic Web. These tools often offer a mixture of features such as faceted search, clustering of search results, etc. Examples of such systems include Lore [4, 5], Flamenco [6], mSpace [7], Haystack [8], Magnet [9], DBin [10], semantExplorer [11], Piggy Bank [12], Tabulator [13], and others. Many of them combine semantic data with "classical" Web content or other human-oriented content in order to provide the user comprehensive access to information.

More recently the idea of a "Semantic Desktop" has emerged [14, 15, 16]; these systems largely focus on personal information management and demon-

strate the utility of exposing "legacy" data in Semantic Web formalisms, often involving transformations from both file-based data as well as databases. The large body of work on mapping data between relational databases and Semantic Web formalisms is described in [17].

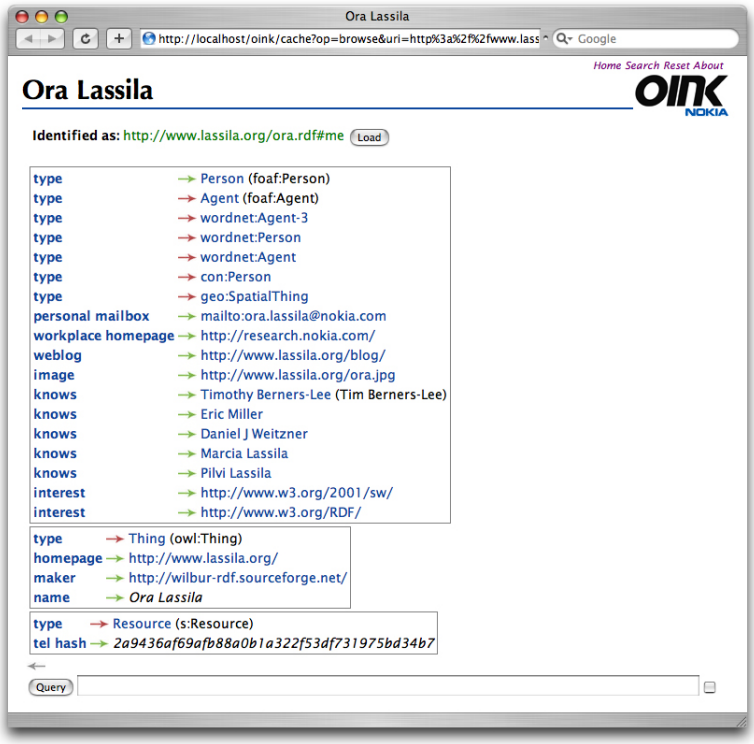


**Fig. 1.** A page from OINK as seen through a typical web browser

## 3 Browsing RDF Data

Since the pervasive mainstream adoption of the World Wide Web, browsing has become a natural user interface paradigm. In order to enable users to browse RDF graphs we have constructed a simple tool called OINK ("Open Integration of Networked Knowledge"). It provides a *node-oriented view* of graphs, visualizing each node as a Web page. These pages show the various ways of identifying the node in question, as well as a list of properties of the node and their values. Figure 1 shows a typical page from OINK, as rendered in a Web browser. OINK allows the simultaneous viewing and exploration of any number of RDF graphs, thus supporting the use of RDF in information integration (technically,

all documents loaded into OINK form a *single* graph; OINK's reasoner supports not only RDF(S) but also *inverse functional properties* as defined by OWL [18] – allowing, say, automatic integration of FoaF [19] profiles).

The data contained within OINK should be considered a cached representations of distributed data, with the feature that the user is free to modify and otherwise manipulate the data as he sees appropriate. For example, the data can be subjected to various reasoners, it can be considered locally closed, and it can be augmented with the user's own annotations.

In addition to visualizing the *outbound* edges, OINK shows the *inbound* edges as well. This allows users to navigate the edges of a graph in the reverse direction (a typical example of this would be when one wants to navigate from an RDF class to any of its subclasses). Since OINK is based on the RDF metamodel – where everything has a representation in RDF itself – all items on an OINK page may support navigation (by clicking) – this includes the definitions of the RDF properties *and* their values. The simple metamodel translates directly to a clear and understandable user interface model.

## 4 Platform for Browsing and Integration

We originally envisioned OINK as a lightweight tool primarily for "debugging" RDF data, but subsequent use of the system has encouraged its development as a platform for building customized browsing solutions for complex data. The use of the underlying reasoning engine also allows automatic integration of data from multiple sources, further reinforcing the idea that Semantic Web technologies can be used for "spontaneous" end-user tasks.

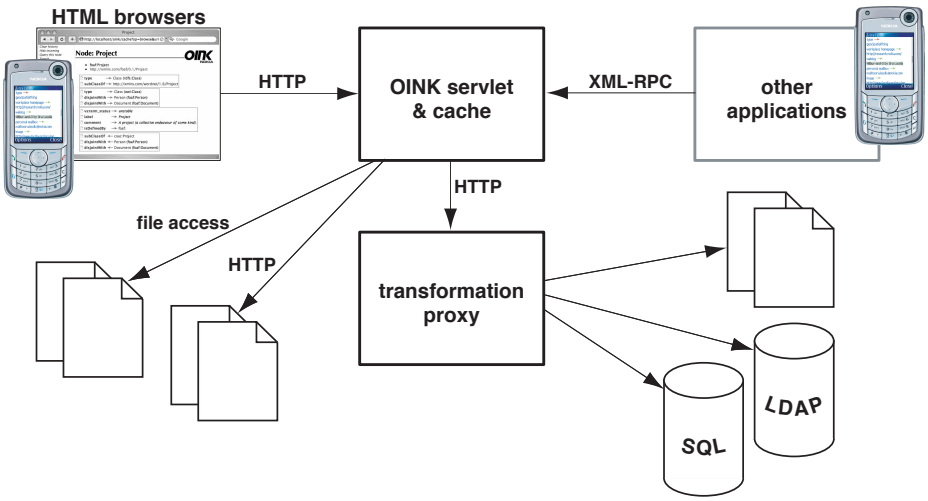


**Fig. 2.** The overall architecture of OINK

The overall architecture of the OINK system is illustrated in Figure 2. In broad strokes, OINK is built around a storage and query engine for RDF graphs. An HTTP *servlet* queries the data in the RDF store, and renders it as XHTML. The current prototype implementation of OINK is written in Common Lisp [20] on top of the WILBUR Semantic Web toolkit [21, 22], and uses the Portable AllegroServe web application server [23, chapter 26].

The main-memory RDF database ("triple store"), acting as a cache, is the central component of OINK.[3] The system operates in terms of *data sources* that are dynamically loaded into the cache. Various facilities are available to the user to maintain the "freshness" of the data in the cache:

**Manual Reloading:** OINK relies on the underlying WILBUR mechanism where "old" data is replaced by freshly parsed "new" data as an atomic transaction on the database. Every RDF statement in OINK is linked to the data source(s) that asserted it. Sources are treated as nodes, and any statements about these sources show up in OINK as well.

**Specification of Refresh Interval:** Data sources can also be *automatically* refreshed either by the user or the document itself designating an interval after which the data in the cache is considered "old".

**HTTP Caching Semantics:** Refreshing can also rely on some of the HTTP headers from the response received when the data was previously loaded.

The OINK cache merely remembers *where* the data comes from, so reinitializing the cache causes all the data sources to be loaded anew from their original locations. The user can easily add more data sources merely by browsing to the node identifying a data source and requesting the particular source to be loaded.[4] Resolving issues related to automatic discovery and acquisition of data by means other than browsing is still work in progress.

The cache can be searched for substring matches on literals or queried using WILBURQL path queries [22]. OINK supports a mechanism where queries are generated automatically from the user's browsing paths through the graph; these queries, in turn, can be used to find similarly related items [25].

Based on the WILBUR triple store, the OINK cache supports RDF(S) entailments [26] via on-demand generation of the *deductive closure* of the graph stored in the cache. Reasoning is based on a technique where access queries to the triple store are rewritten so that they reflect the "virtual" deductive closure [22]. Effectively, the reasoner supports an extended version of RDF(S) augmented with integration-related features such as owl:sameAs and *inverse functional properties* [18] – we call this representation vocabulary "RDF++". Since the reasoner does not modify the contents of the cache, inconsistencies in data can be handled gracefully by merely retracting the "offending" triples.

The OINK servlet merely responds to HTTP requests by querying the cache and producing pages – one per each node in the graph, on demand – that reflect

[3] So as not to confuse the reader, it should be pointed out that this is a lower-level caching mechanism than the "Semantic Cache" we have reported on earlier [24].

[4] Data sources are identified using URIs; since nodes in the graph may also be identified using URIs, each data source appears as a node stored in the cache.

the graph structure. By default, the generation of a page for an RDF resource (node) is divided into three components:

**Identity:** The node is identified in various ways known to the system. If present, the value of rdfs:label property (or some subproperty thereof) is used. A link is also provided to the Web resource itself: this is useful, for example, if the node described is a real Web page or some other document that can be rendered in the Web browser.

**Properties:** The properties of the node are arranged into clusters corresponding to the most specific non-overlapping classes of the node, using associated rdfs:domain specifications. Any "leftover" properties are assumed to be associated with rdfs:Resource and are visualized last. The classes that are considered include both asserted and inferred classes of the node. Finally, inbound arcs are visualized separately. Each cluster of properties can be associated with a special markup generator that can emit class-specific visualizations (this aspect of the system is elaborated in Section 5).

**Navigation History:** OINK tracks the path the user has taken to navigate to any specific node; this history is shown and allows the user to invoke queries generated automatically from the navigational path [25].

OINK will be able to generate structurally simplified pages when it detects that the requesting client is a limited, small device, such as a mobile phone.[5] Browsing semantic data on a low-bandwidth, small-screen device is appealing, since the information is in a more compact and possibly more succinct form than natural language HTML.

In order to be able to load various "legacy" data into OINK, we use a transformation engine that allows data format transformations to be written in XSLT or as CGIs. In a loose sense the engine takes the form of an HTTP *proxy*. OINK makes HTTP requests to the proxy, designating not only the original data source but also the desired transformation. The proxy loads the data, applies the transformation, and hands the resulting RDF to OINK.[6] The purpose of the transformation engine, from the architectural viewpoint, is to separate data format conversions from the rest of the system. OINK is a "pure" RDF application in the sense that it does not know anything about other data formats.

## 5 Customized Views of Integrated Data

The use of OINK as a means of accessing information integrated from multiple sources can benefit from mechanisms for customizing the view of the data for each user. Generally, the following mechanisms are used:

**Context-aware Prioritization:** The user's current *usage context* – effectively, the state of the world as defined in [28, 29] – can generally be used as a means of narrowing the possible space of interactions (e.g., what the user might be

[5] The current prototype only knows about Nokia S60 phones.
[6] We are in the process of implementing GRDDL support [27].

looking for). More specifically, as we have demonstrated in [24], knowledge of context can be used in identifying "objects of interest" that might serve as default values for various user interactions as well as to eliminate impossible choices; the context can also serve to prioritize collections of objects (such as search results). With limited screen space of mobile devices this can be of great benefit.

**Policy-aware Filtering:** Any system that supports the sharing of information among a group of users has to support some forms of access control. We believe that the conditions of allowing (or denying) access to a particular information object could often be complex and might defy being expressed in terms of traditional group-based access control. *Ad hoc* situations involving information sharing could not only involve *intensional* descriptions of access rights but these policies might even have to be adapted in a context-aware manner [30]. In our current implementation we plan to implement policy-awareness as a means of filtering information (at fine granularity), based on sophisticated policy representation.

**Data-driven Customization:** It should also be possible to customize the view of data based on the data itself, e.g., based on the data types involved. OINK offers a class-based framework for providing customized methods for generating data visualization. Effectively this allows each RDF class to be associated with its own method of visualization, yet making the system flexible in the sense that "unknown" data can also be visualized via the default rendering method(s). At the time of writing, we are also working on supporting the *Fresnel* presentation customization vocabulary [31].

Given that OINK can acquire data from multiple data sources, including legacy sources, and with the hypothesis that the notion of *context* can capture any information we might have about the user, the above mechanisms, in concert, allow us to build highly customized and personalized data browsing solutions. Such solutions can be thought of as applications in their own right; conceptually (and perhaps architecturally) they are not unlike modern Web browser "plug-ins". We expect most OINK customizations to be loaded as yet more RDF data.

## 6 Approaches to Spontaneous Collaboration

Spontaneous collaborations between mobile devices can be enabled by browsing an integrated view of the semantic data that resides on the individual devices involved in the collaboration. In order to create an integrated view of the data, devices should be willing to either:

- provide results of running a remotely invoked query, or
- export some subset of their data.

In the first case, each device would be required to run a local instance of OINK and the XML-RPC interface of OINK could be used to query data from remote devices. The user would still be able to frame queries by browsing the local data

and schemas. The query formulated during browsing is sent to selected devices in the collaboration and the individual results are merged together and reasoned upon before they are presented to the user. Selection of preferred devices could be made easy by recommending relevant devices based on the user's current context, such as devices that are in physical proximity, devices of participants in a currently active meeting etc. In this approach, access control and policy enforcement on the data would be done at query time by the remote device's OINK instance.

In the latter approach, *a priori* knowledge of what kind of data is useful or relevant for the collaboration will be required before the data is exported. Extra bandwidth would be utilized in transferring the data, and the device requesting data would be required to have sufficient memory and computation power to run a localized view of the integrated semantic data. However, there are some advantages to this approach: the response time for queries is likely to be fast, better reasoning support can be offered since all the data is in one place and some of the less resourced devices could share data without running a local instance of OINK.

In practice, we expect that a combination of both approaches will be used. Repeatedly needed data with loose access privileges could be shared *a priori*, while less frequently used data could be shared at query time. In our recent experiments with OINK we have mostly focused on the subset data export approach, in the future we would like to focus on a combination of both approaches.

Another kind of integration occurs through unplanned associations of data via the use of common identifiers. While any type of URI is appropriate for use with RDF, some types of URIs have particular features that make them more likely to facilitate such spontaneous data integration. As shown in Figure 2, OINK makes HTTP requests to the Web to load its cache. The HTTP protocol provides a simple mechanism for a responding server to direct the requesting client to other resources in the Web in place of returning the requested content. The most widely deployed of these HTTP responses has the semantics "the name of the resource you requested has changed and is now ...".

A far more powerful HTTP response has the very general semantics "this server is not able to directly provide the requested content but this other resource may be relevant to your request". The distinguishing characteristic of the use of this HTTP response in Semantic Web applications is that the "other" resource can describe (e.g., in RDF) precisely *how* it is related to the resource first requested in addition to providing any other RDF data it wishes. That is, the related resource can *always* be a document that contains data whereas the resource identified by the original request URI might be an abstract concept or a physical object, neither of which have content that can usefully be transported via current Internet protocols.

If we augment OINK with user-specified policies about when to follow (dereference) URIs to such related resources, OINK can be made to automatically load into its triple store information from data sources of whose existence the user was not explicitly aware. We expect that by using the personal context information

available to the mobile device, OINK can augment these policies and requests to create new opportunities for spontaneous collaboration.

## 7 Application Scenarios

In this section we present some scenarios using applications we have built based on OINK. The applications are entirely data-driven and there were no customizations made to OINK to make the specific scenarios work as described – that is, any semantics of the applications are encapsulated within the data they use.

The first scenario demonstrates data merging to identify participants in a teleconference. The following data sources were used:

- *Zakim* [32], a Semantic Web agent, helps facilitate W3C meetings using IRC and an audio teleconference bridge. When a phone call is made into the conference bridge, Zakim registers the participant's presence and identifies the participant by the caller's telephone number. If the telephone number was previously known to Zakim, it identifies the participant using the previously given name. Zakim makes available as RDF the data it collects about active teleconferences and participants. This data includes the name of each participant and a hash (for privacy reasons) of the caller's telephone number.
- *OinkIt*, an application that creates FoaF [19] profiles from the contact book on a Nokia S60 phone. OinkIt uses the foaf:knows relationship for the entries in the contact book and uses the same hash function as Zakim to include the contacts' telephone numbers. For privacy reasons, OinkIt only considers contacts that the user has placed in a specified contact group. OinkIt exports the resulting RDF from the phone to OINK via the XML-RPC interface. OINK's XML-RPC interface returns a URI of a page in OINK from which the user can start browsing the loaded data (see Figure 3).


**Fig. 3.** Exporting a FoaF profile from the phone

**Identifying Teleconference Participants:** OINK's integrated view of data from Zakim and the FoaF profiles (from some of the participants) helps in identifying participants who were not previously known to Zakim. The RDF++

reasoning support in OINK makes this possible; the semantics of the hashed telephone number property are declared to be OWL inverse functional; that is, two contacts who share the same hashed telephone number are declared to be the same individual.[7] OINK can therefore merge data about individuals from the phone contact books with data about individuals in Zakim's teleconference participant list.

**Establishing Social Connections:** Once the identity of the participants is resolved using OINK, the browsing features of OINK can be used to discover social links between conference participants. By viewing the inbound links for a participant in OINK, it becomes easy to determine all the people who know the participant. This could be useful in making introductions.


**Fig. 4.** Restaurant recommendation based on social collaboration

In the second scenario, a constraint solver, the Alloy Analyzer [33] uses OINK's integrated view of the data to find a restaurant that would satisfy the preferences of a group of people. The preferences of multiple users are specified as constraints and gathered from group members' phones using OinkIt. OinkIt exports any preferred restaurants of a user along with his cuisine preferences into OINK (see Figure 4). Group members identify their preferred restaurants by putting the corresponding contact entries into a "restaurants" group in their contact lists. The identity of preferred restaurants from the users' contact lists are established by merging the users' data with data about restaurants from Yahoo's database and applying inverse functional reasoning on the phone number property of restaurants. OINK does this without any special knowledge about restaurants or phone numbers. Establishing the identity of preferred restaurants enables us to reuse data from the users' contact lists. Finally, Alloy uses the XML-RPC interface of OINK to query for user preferences and solves the constraints to recommend a good restaurant that is in proximity to the group's current location.

[7] This works well enough in practice today and is getting more accurate as more and more people rely exclusively on their mobile phones for voice communications.

## 8 Conclusions

We have presented a generic customizable RDF data browser and integrator, which facilitates spontaneous collaborations. Our experiments with several data sources indicate that the tool is useful as an application development platform.

The incorporation of two features from OWL (owl:sameAs and inverse functional properties) into the base RDF(S) semantics (referred to as RDF++) gives us a language that is good at resolving identities; our aim was a minimal extension that would provide better data integration support. Our *on-the-fly* reasoning technique is practical and can facilitate spontaneous and serendipitous integration. We conclude that rewriting queries to generate on-demand virtual deductive closures is an effective technique for dealing with inconsistencies in merged data.

In the application scenarios section we demonstrated that mobile devices have a wealth of information that is strongly related to their owners' context and that this data can be used to facilitate collaboration. Context-sensitive view customization is particularly appealing for mobile devices where both screen real estate and users' attention are at a premium. In our concrete implementation, this is still work in progress.

The tool also keeps track of the user's browsing path to automatically construct queries for finding similarly related objects; this is a form of *query-by-example* technique, which works very well for non-expert users.

OINK is a light-weight tool, written in very few lines of code – this makes it easy to port it to other platforms. We have currently reimplemented WilburQL in Python [34] and are working towards translating the rest of OINK to Python as well. Our future goal is to have OINK working on mobile devices that can run PyS60, the Python interpreter for the Nokia S60 platform. We also plan to incorporate speech-based interactions into OINK.

## References


[1] Lassila, O., Swick, R.R.: Resource Description Framework (RDF) Model and Syntax Specification. W3C Recommendation, World Wide Web Consortium (1999)

[2] Brickley, D., Guha, R.: RDF Vocabulary Description Language 1.0: RDF Schema. W3C Working Draft, World Wide Web Consortium (2003)

[3] Berners-Lee, T., Hendler, J., Lassila, O.: The Semantic Web. Scientific American **284**(5) (2001) 34–43

[4] Goldman, R., Widom, J.: DataGuides: Enabling Query Formulation and Optimization in Semistructured Databases. In: VLDB '97: Proceedings of the 23rd International Conference on Very Large Data Bases, San Francisco, CA, USA, Morgan Kaufmann Publishers Inc. (1997) 436–445

[5] Goldman, R., Widom, J.: Interactive Query and Search in Semistructured Databases. In: WebDB '98: Selected papers from the International Workshop on The World Wide Web and Databases, London, UK, Springer-Verlag (1999) 52–62

[6] Yee, K.P., Swearingen, K., Li, K., Hearst, M.: Faceted Metadata for Image Search and Browsing. In: CHI '03: Proceedings of the SIGCHI conference on Human factors in computing systems, New York, NY, USA, ACM Press (2003) 401–408
[7] monica schraefel, Karam, M., Zhao, S.: mSpace: interaction design for user-determined, adaptable domain exploration in hypermedia. In: AH2003: Workshop on Adaptive Hypermedia and Adaptive Web-Based Systems. (2003)
[8] Quan, D., Karger, D.: How to Make a Semantic Web Browser. In: Proceedings of the 13th International Conference on the World Wide Web, ACM Press (2004) 255–265
[9] Sinha, V., Karger, D.R.: Magnet: Supporting Navigation in Semistructured Data Environments. In: SIGMOD '05: Proceedings of the 2005 ACM SIGMOD international conference on Management of data, New York, NY, USA, ACM Press (2005) 97–106
[10] Tummarello, G., Morbidoni, C., Puliti, P., Piazza, F.: The DBin Semantic Web Platform: An Overview. In: Workshop on the Semantic Computing Initiative (SeC 2005), Chiba, Japan (2005)
[11] Scerri, S., Abela, C., Montebello, M.: semantExplorer: A Semantic Web Browser. In Isaías, P., Nunes, M.B., eds.: IADIS International Conference WWW/Internet 2005. (2005) 35–42
[12] Hyunh, D., Mazzocchi, S., Karger, D.: Piggy bank: Experience the Semantic Web Inside Your Web Browser. In Gil, Y., Motta, E., Benjamins, V.R., Musen, M.A., eds.: The Semantic Web – ISWC 2005, 4th International Semantic Web Conference, Springer-Verlag (2005)
[13] (http://dig.csail.mit.edu/2005/ajar/ajaw/Help)
[14] Decker, S., Frank, M.: The Social Semantic Desktop. Technical Report DERI-TR-2004-05-02, DERI (2004)
[15] Sauermann, L.: The Gnowsis Semantic Desktop for Information Integration. In: 1st Workshop on Intelligent Office Appliances. (2005)
[16] Sauermann, L., Schwartz, S.: Gnowsis Adapter Framework: Treating Structured Data as Virtual RDF Graphs. In Gil, Y., Motta, E., Benjamins, V.R., Musen, M.A., eds.: The Semantic Web – ISWC 2005, 4th International Semantic Web Conference. Number 3729 in Lecture Notes in Computer Science, Springer-Verlag (2005) 1016–1028
[17] Beckett, D., Grant, J.: Mapping Semantic Web Data with RDBMSes. SWAD-Europe Deliverable 10.2, World Wide Web Consortium (2003)
[18] Patel-Schneider, P.F., Hayes, P.J., Horrocks, I.: OWL Web Ontology Language Semantics and Abstract Syntax. W3C Recommendation, World Wide Web Consortium, Cambridge, MA (2004)
[19] Brickley, D., Miller, L.: FOAF Vocabulary Specification. http://xmlns.com/foaf/0.1/ (2004)
[20] Steele, G.L.: Common Lisp the Language, 2nd edition. Digital Press (1990)
[21] Lassila, O.: Enabling Semantic Web Programming by Integrating RDF and Common Lisp. In: Proceedings of the First Semantic Web Working Symposium, Stanford University (2001)
[22] Lassila, O.: Taking the RDF Model Theory Out for a Spin. In Horrocks, I., Hendler, J., eds.: The Semantic Web - ISWC 2002, 1st International Semantic Web Conference. Volume 2342 of Lecture Notes in Computer Science., Springer-Verlag (2002) 307–317
[23] Seibel, P.: Practical Common Lisp. APress, Berkeley, CA (2005)

[24] Khushraj, D., Lassila, O.: Ontological Approach to Generating Personalized User Interfaces for Web Services. In Gil, Y., Motta, E., Benjamins, V.R., Musen, M.A., eds.: The Semantic Web – ISWC 2005, 4th International Semantic Web Conference. Number 3729 in Lecture Notes in Computer Science, Galway, Ireland, Springer-Verlag (2005) 916–927
[25] Lassila, O.: Generating Rewrite Rules by Browsing RDF Data. Accepted to the RuleML-2006 conference (2006)
[26] Hayes, P.: RDF Semantics. W3C Recommendation, World Wide Web Consortium (2004)
[27] Hazaël-Massieux, D., Connolly, D.: Gleaning Resource Descriptions from Dialects of Languages (GRDDL). W3C Team Submission, World Wide Web Consortium, Cambridge, MA (2005)
[28] Khushraj, D., Lassila, O.: CALI: Context-Awareness via Logical Inference. In: ISWC 2004 workshop on Semantic Web Technology for Mobile and Ubiquitous Applications. (2004)
[29] Lassila, O., Khushraj, D.: Contextualizing Applications via Semantic Middleware. In: The Second Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services (MobiQuitous), IEEE Computer Society (2005)
[30] Toninelli, A., Montanari, R., Kagal, L., Lassila, O.: A Semantic Context-Aware Access Control Framework for Securing Collaborations in Pervasive Computing Environments. Accepted to the ISWC-2006 conference (2006)
[31] Bizer, C., Lee, R., Pietriga, E.: Fresnel – A Browser-Independent Presentation Vocabulary for RDF. In: End User Semantic Web Interaction Workshop at the 4th International Semantic Web Conference, Galway, Ireland (2005)
[32] (http://www.w3.org/2001/12/zakim-irc-bot.html)
[33] Jackson, D.: Alloy: A Lightweight Object Modelling Notation. ACM Transactions on Software Engineering and Methodology **11**(2) (2002) 256–290
[34] van Rossum, G.: The Python Language Reference Manual. Network Theory Ltd. (2003)

## Acknowledgements

Work described in this paper was funded in part by Nokia Technology Platforms. The authors would like to thank the following people for their help and support: Sadhna Ahuja, Jan Bosch, Felix Chang, Barbara Heikkinen, Jamey Hicks, Daniel Jackson, Marcia Lassila, Heikki Saikkonen, and Marko Suoknuuti.