A Framework for Communication Support Agents – Implementation of an RDF-based Personal Repository –

Koji Kamei   Sen Yoshida   Kazuhiro Kuwabara
NTT Communication Science Laboratories, NTT Corporation
e-mail: {kamei, yoshida, kuwabara}@cslab.kecl.ntt.co.jp

Abstract

We propose a framework for developing personal agents that support individuals' communication activities. The framework offers an RDF repository-centered architecture in which a group of applications work collaboratively, supported by the event notification mechanism in the repository. Our RDF-based approach targets the collection of information on interactions between users and information resources, representing those interactions as annotations to original information resources, and employing these annotations to support a user's activity. In this paper, we discuss annotations in personal environments and requirements for personal repositories, then describe the framework's current implementation.

Introduction

In collaborative activity between individuals, sharing and exchanging information among individuals play important roles in communication. We receive a great deal of information from other individuals, interpret it and reflect on it using our knowledge. We then produce new information and forward it to other individuals. We propose a framework for personal agents that aims to support this type of information sharing and exchange among individuals. We primarily target the phases of receiving information and its subsequent use; in other words, our framework aims to support the reflexive phase in an individual's activity. To realize this objective, the personal agent utilizes the user's personal annotations that each individual privately adds to information resources stored at hand.

Here, the annotations include primarily verbal annotations that users' explicitly added to information, and additionally descriptions about interaction between users and information. More specifically, the latter describes how one has handled a resource, such as, "the timestamp when the user read a document," "the folder to which the user moved an e-mail," and "other users to whom the user introduced a Web page." In other words, the annotations that we are concerned with here describe the interaction between a user and items of information, and that between the user and other users. By adding such annotations of interaction, the agent system can organize and reuse information items for its user. Thus, by utilizing the organized knowledge, the agent can support its user's communication activities.

When annotations are added interactively and the organization becomes complicated, it is appropriate to use a schema-less, semi-structured database (Abiteboul *et al.* 1997; Buneman, Fernandez, & Suciu 2000). We have employed the resource description framework (RDF) (Lassila & Swich 1999) to describe this kind of annotations of interaction. In this case, individuals and documents are mapped as *resources* in an RDF statement, and interactions are mapped as *predicates*. The Haystack project (Huynh, Karger, & Quan 2002), for example, shares this type of approach.

It is natural and easy to use RDF statements to describe annotations of interactions. However, ontologies on interaction and information handling have not been adequately discussed to date, due to the difficulty in defining the vocabulary of interaction dissociated from cases of actual use, as opposed to the clearly defined class hierarchies of vocabulary for knowledge representation. It is also addressed in (Stojanovic & Motik 2002) that ontologies must be able to evolve continuously. The current implementation status of our framework described in this paper supplies a means for developing RDF-based annotations of interaction, and contributes to refining the definition of vocabularies through actual implementation of application systems.

In this paper, we describe an agent framework with an RDF-based personal repository. First, we discuss annotations in the personal environment and requirements for the personal repository, then describe current implementation of the framework. Finally, we discuss the future directions of our research, that is, in which application domain we should define the vocabulary of interaction. We describe experimental implementation of two existing communication environments, CommunityOrganizer (Kamei *et al.* 2002), and Gleams of People (Ohguro 2001) as a starting point for this approach.

Personal Repository

In this section, we first discuss the role of annotation in personal environments, and then describe the requirements for a personal repository that enables and utilizes the personal annotations.

Annotations in Personal Environments

In the context of the Semantic Web, annotations change Web pages from human-readable ones to machine-readable ones. The case introduced with the Semantic Web explains that the annotations to a Web page are added by the page's publisher and published worldwide. This model can be classified as a publisher-driven, open-style method of annotation.

On the other hand, information resources acquired by individuals are not limited to public items such as Web pages, but also include non-public items such as private e-mails. Moreover, private information is often more important than public information; therefore it would be natural to suppose that users would also add annotations to items of private information. Furthermore, it is important to pay attention to the receiver-side annotations in addition to publisher-side annotations.

Although both the RDF model and other existing implementations of the annotation server cover this type of receiver-side annotation, the use of personal annotations have not yet been mentioned. For example, the scenario described by the Annotea project (Koivunen & Swick 2001) proposes a case for their use, in which an annotation server is deployed globally to collect and share the annotation statements, and clients then share the server. It also allows multiple servers to exist for different purposes and for different groups of users, but does not address the concept of private annotation servers and collaboration.

Nejdl *et al.* discussed collaborative annotation (Nejdl *et al.* 2002) as a part of *EDUTELLA*, which is a framework to allow communication between different RDF repositories. They proposed a modification exchange language designed to keep decentralized repositories coherent. Although the language is useful for exchanging personal annotations, their interests seem heading for rather coherency than individuality.

Requirement for Personal Repository

data model and back-end storage The personal repository needs to support the features of an RDF database, since relationships between individuals and items of information can be described as RDF statements. Additionally, an auxiliary information storage item is required to save these items and issue a URI to each one. Object-oriented database systems or XML database systems with Object-XML mapping features are suitable for this requirement.

event notification framework Since multiple applications run simultaneously on a personal agent framework and share RDF statements and information items in the repository, they need to be notified when the data that they are referring to changes. An event notification framework enables application modules to collaborate with each other.

inter-repository communication The communication layer between personal agents is based on the concept of peer-to-peer communication; that is, each agent can send messages directly to any known agents. In actual cases of group communication, messages may also be exchanged in a server-oriented manner.

Implementation of Personal Repository

The personal repository is currently implemented in a Java2 SDK 1.3.1 environment, and employs various open source products relating to XML technologies. The whole system architecture is shown in Figure 1.

Back-end Storage

Data in personal repositories are stored in both XML and RDF storage. The former assigns a URI to each XML document so that the items can be addressed using URIs from RDF statements. The latter holds RDF statements that describe relationships between information items: items in XML storage and those defined outside of the repository. Both XML and RDF storage possess specialized retrieval functions.

The RDF storage is implemented on a Jena Semantic Web toolkit [1]. For the back-end storage of Jena, MySQL [2] is used for persistency. The XML storage utilizes eXist[3], which implements XML:DB API [4] and shares the back-end MySQL system with the RDF storage.

The query interface to the repository is currently defined at the Java API level, though it is obviously necessary to define a query language independent from low-level APIs. We are currently on the way to designing a repository query language that supports not only the query/update function, but also our event notification framework (described below). We alos plan to implement a script engine for the language as a service on the event notification layer.

Event Notification Framework

The event notification framework is built upon both types of storage described in the previous section. This framework monitors *write* operations to the storage, and notifies application modules of modifications such as registration of an RDF statement related to particular nodes and/or predicates, or registration of an XML document with a specific element and/or value. This feature is based on the concept of the active database.

Personal agent applications consist of multiple modules that are activated by the event notification framework on the repository. Each module registers a template pattern for each condition from which it desires to be notified of the change.

The repository contains a group of core modules that are started when the repository is invoked. In addition,

[1]http://www.hpl.hp.com/semweb/jena-top.html
[2]http://www.mysql.com/
[3]http://eist-db.org/
[4]http://www.xmldb.org/xapi/index.html

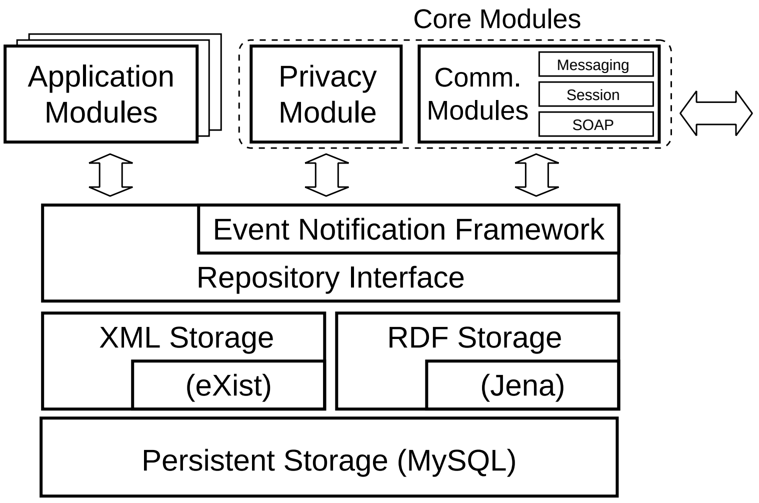


Figure 1: The architecture of the personal repository.

the repository can contain various modules that are registered by applications invoked by the user. Since multiple applications can listen to a repository simultaneously, a store operation (of an RDF statement or XML document) performed by a certain application module may trigger another application. As a result, coordination among applications can be realized.

The event notification between user interface modules and the repository, and information exchange between other personal agents are also implemented using this event notification framework.

Overall behavior of the agent and applications is realized by the interaction of those modules through data (and changes in data) in the repository.

Communication between Personal Agents

The protocol for communication between personal repositories supports two features. One is connection management, which determines where the peer exists, and the other is a simple messaging feature that delivers an XML document to the peer. These protocols are implemented as messaging services on SOAP [5], which is a lightweight protocol intended for exchanging structured information in a decentralized, distributed environment. We defined the protocol using SOAP to keep the protocol itself independent of Java language in order to secure interoperability. While SOAP provides a range of features, such as remote procedure calls, message forwarding, etc., we employ its message exchange feature on the HTTP protocol to exchange XML elements. Currently, the services are implemented with Apache Axis [6] SOAP library and deployed to Tomcat [7] servlet containers. Since inter-repository communication is performed in a peer-to-peer manner, each repository needs to invoke its own servlet container to receive requests from other repositories.

At this communication level, each agent can register templates, which describe the notification condition, to the other agents' repositories to request that it be notified of the changes in the target repository. A privacy management module is deployed to monitor the registration of templates from other repositories and control this remote notification feature.

Repository Viewer

While developing modules for a personal repository, we also implemented a repository viewer as one of the development support tools. The repository viewer is a tool for browsing and/or authoring RDF statements and XML documents stored in a repository. When a user specifies the URI of a node in the repository, it retrieves RDF statements that describe the specified node as its subject or object. The result is represented in two ways: graphic representation in a two-dimensional plane, and list-like representation.

In the graphical view (Figure 2a), the resource node to which a user is paying attention is located at the center of the view. Other nodes located around the view are resources that are arranged in two hops according to their relationship with the center node. By pointing to the edge with a mouse, the RDF triple of the edge is displayed at the bottom of the screen. By clicking a node on the view, the view is updated to place the clicked node at the screen's center. In the listing view (Figure 2b), the *predicate* of incoming edges is listed on the left of the screen, and outgoing edges on the right. To support direct jumps to other nodes, a list of all subject nodes is also provided.

Although its current representation is limited, this tool is helpful in developing our application because it

[5]http://www.w3.org/TR/soap12-part0/
[6]http://xml.apache.org/axis/
[7]http://jakarta.apache.org/tomcat/

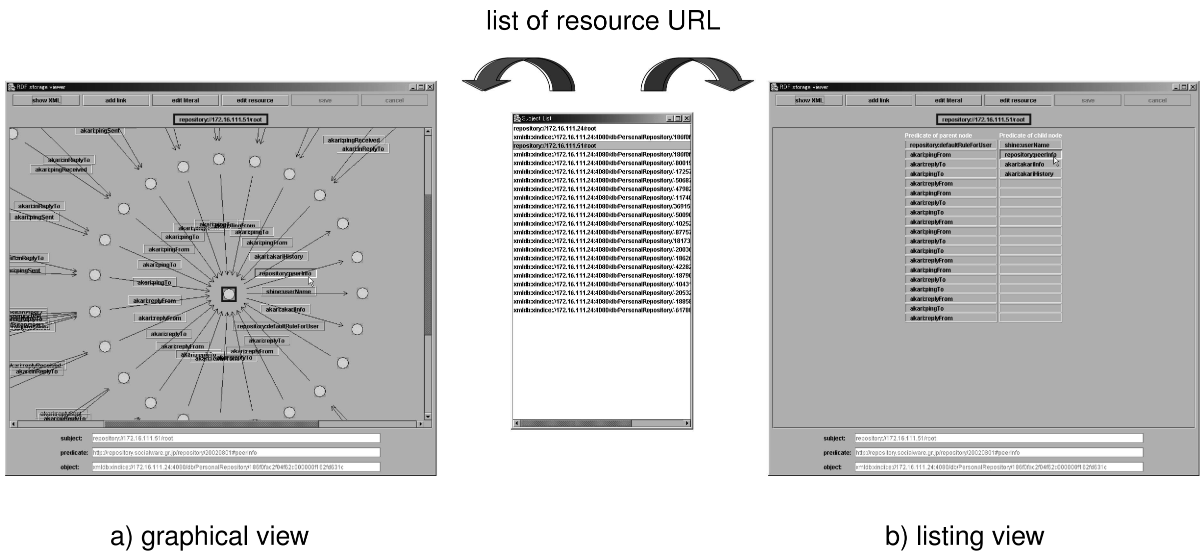


Figure 2: The screen image of the repository viewer.

allows us to inspect and modify the items in the repository while in operation. The tool also monitors the repository's event notification framework, so that the changes in the repository are immediately reflected in the view.

Toward Defining Vocabularies of Interaction

To define vocabularies for describing various annotations, we believe that a trial-and-error method is advantageous; therefore, we first implemented a repository along with its interactive RDF viewer using that method. We are concurrently analyzing interactions in several specific application domains as the next logical step.

With regard to ontology definition, there are various standard vocabularies suitable for knowledge representation. However, it is difficult to find ontologies on human activities and information handling. *FOAF* (Brickley, Miller, & rdfweb-dev listmembers 2002), for example, is one well-known ontology published recently that defines vocabularies for describing individuals and relationships between them. Although the FOAF ontology proposes *"knows"* relationships, and covers vocabularies suitable for introducing individuals to others, it cannot describe activities of the individuals.

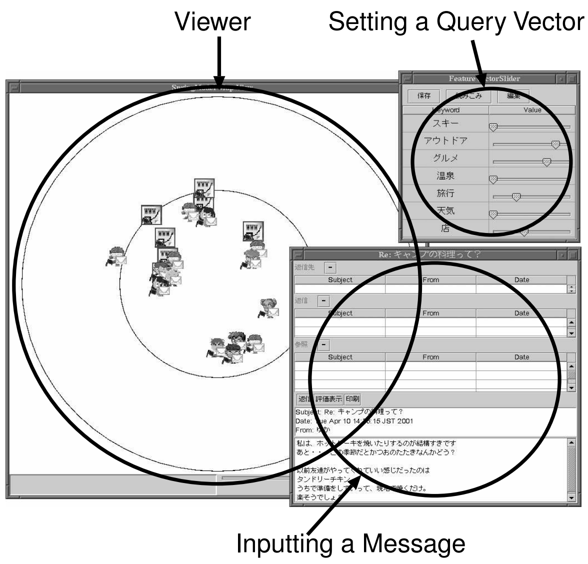
In the multi-agent framework *Shine* (Yoshida *et al.* to appear), which we have developed to support network communities, we proposed the concept of a "person database." Applications on Shine can store any kind of information relating to a "person." An application on Shine, *CommunityOrganizer*, has been developed to support the initial phase of informal communities on the network by assisting users to become aware of information resources and other people with common interests. Since it covers several types of interaction from direct communication by exchanging verbal messages to indirect communication such as collaborative recommendation, we plan to focus on defining vocabularies for this application in the next step.

In the following section, we describe the internal design of two applications, named CommunityOrganizer and Gleams Of People, which were implemented on Shine. Those applications are currently being (re-)implemented on top of the RDF-based repository. An example vocabulary usage in those applications are illustrated in Figure 5.

CommunityOrganizer

CommunityOrganizer provides a shared space where public short messages and Web page introductions are exchanged, and in which people are made to be aware of possible communities of interests. Figure 3 shows the primary component of its user interface. Information items relevant to the user's interests are shown as icons on the two-dimensional display. Icons are arranged so that relevant icons are located closer and form clusters. The view is updated when new information items are added; therefore, users can become aware of the dynamic formation of information clusters. In other words, when several messages concerning a topic are posted in a short period, the messages (and relevant resources) move to create an animated cluster.

Since the exchanged messages are similar to e-mail, they contain header information such as *sender*, *subject*, *timestamp of creation time*, etc. — fields that are represented as annotations to a message in RDF storage. To calculate the relevance, a feature vector is added for each message to represent the content. The vector is described as an XML document consisting of a set of



Re: キャンプの料理って？

Subject: Re: キャンプの料理って？
Date: Tue Apr 10 14:[illegible]:15 JST 2001
From: [illegible]

私は、ホット[illegible]ーキを焼いたりするのが結構すきです
あと・・・この季節だとかつおのたたきなんかどう？

以前友達がやってくれていい感じだったのは
タンドリーチキン
うちで準備をしていって、現地で焼くだけ。
楽そうでしょ？

Figure 3: CommunityOrganizer (screen image)

keyword-value pairs, and the document also becomes an annotation (*content feature*) to the original message.

Shine As mentioned before, CommunityOrganizer is developed using Shine framework. Shine is a peer-to-peer agent framework that provides basic vocabularies about agents, users, communication and communities. The vocabularies defined in Shine provides a start point for defining vocabularies for personal agent frameworks.

In the Shine, the primary elements of the system are an agent and its user, so each agent has *agentID* and *username* attributes. From the perspective of the repository's owner, other users (in the repository) are represented as *acquaintances*.

Nakif In CommunityOrganizer, a hybrid filtering technique called *Nakif* is integrated. Nakif incorporates a content-based method into a collaborative filtering system, thus it provides high quality evaluations of communities of interests (Funakoshi *et al.* 2001).

When a user evaluates a document (assigns a *positive* or *negative* score) in Nakif, the evaluation is shared among members in the communities of interest. When a user receives an evaluation, Nakif appends the *evaluation* as an annotation to the target resource, and updates the evaluator's *user profile* vector based on the *evaluation* and the *content feature* of the resource. The learning module in Nakif gathers those *user profile* vectors and treats the collection as a *user profile matrix.*

Vocabularies for Information Sharing In the applications described above, information resources are assumed to be stored locally beside the user, and to become shared gradually through peer-to-peer information exchange.

Communication items exchanged between people are not limited to messages they have written, but also include introductions to information items that another person has created. In other words, a message created by a user can be received by another person through peer-to-peer intentional message forwarding. In this sense, *forwarded* and *introduced* can be included in information-sharing vocabularies.

In CommunityOrganizer, introduced Web page resources include an *introduced by* annotation. In Nakif, the exchange of *evaluation* carries this function because it can indicate the existance of the evaluated document.

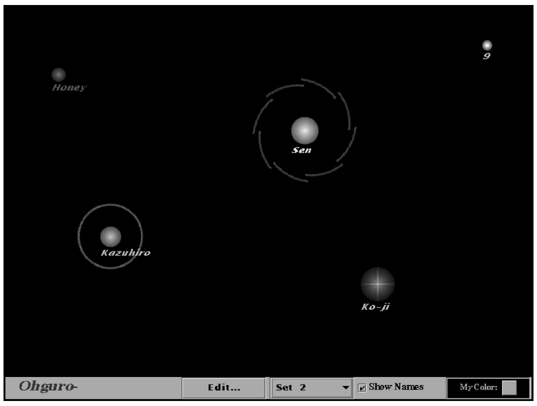


Figure 4: Gleams of People (screen image)

Gleams of People

Gleams of People is a very lightweight communication medium that is designed to convey a simple greeting (such as "how are you?") or a message just intended for keeping in touch.

Figure 4 shows a screen image of Gleams of People. A message can be sent by touching the sphere corresponding to the intended receiver. The content of the message is just a 'color,' which represents the mood of the sender. When a message is received at the receiver's side, a sphere gleams with the color sent, then the receiver's agent automatically sends back its owner's mood. In this way, it conveys *things that are not so important to talk about, but that are worth expressing.*

The spheres indicate presence and status information of corresponding users at a grance, and also indicate history of communication activities unobtrusively; that is, the interval for which the sphere gleams reflects the number of messages received from the corresponding user because it is important to present the communication history to the user in order to foster social relationships.

With a personal repostitory, the system can record the *akari status* that describes how the gleam is expressed, and the history of exchanged messages with respect to its user.

Conclusion

In this paper, we proposed a framework for developing personal agents that support communication ac-

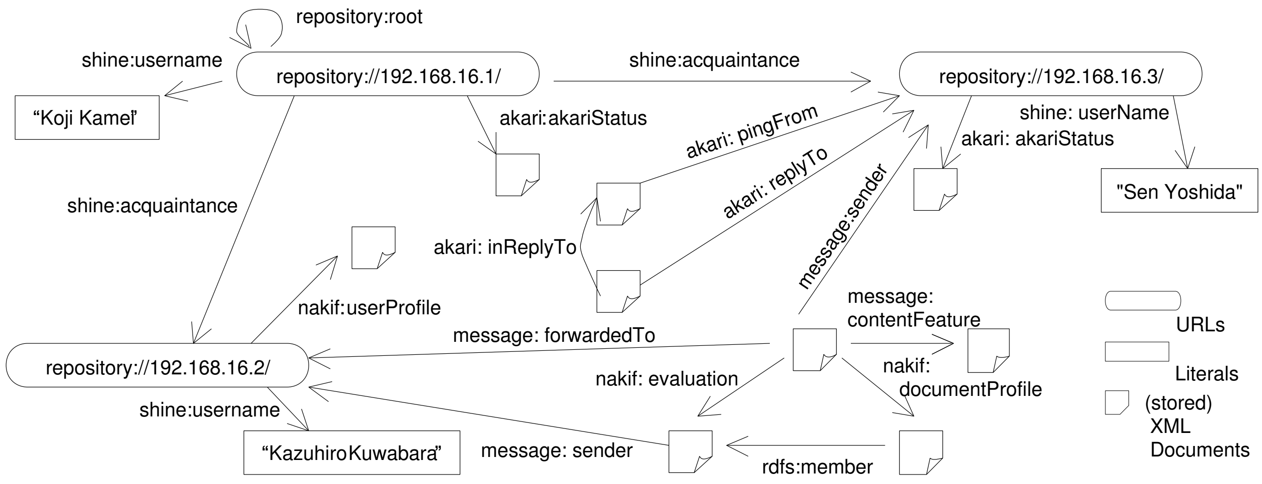


Figure 5: Vocabularies used in existing applications

tivities of individuals. The framework offers an RDF repository-centered architecture on which a group of applications and services work collaboratively, supported by the repository's event notification mechanism. The framework and support tools described above have not been fixed, but they should be improved through experience in implementing actual interaction-oriented applications. As a starting point for defining vocabularies for interaction, we described experiments that implement existing applications on top of the personal agent framework.

Acknowledgments

The concept of a personal repository arouse through continuous discussions with Mr. Kaname Funakoshi and Mr. Takeshi Ohguro. They also gave us useful advice for incorporating their works Nakif and Gleams of People. In addition, we wish to thank to Mr. Takashi Kawachi for the implementation of the personal repository and its framework,

References

Abiteboul, S.; Quass, D.; McHugh, J.; Widom, J.; and Wiener, J. L. 1997. The Lorel query language for semistructured data. *International Journal on Digital Libraries* 1(1):68–88.

Brickley, D.; Miller, L.; and rdfweb-dev listmembers. 2002. FOAF: the 'friend of a friend' vocabulary. Technical report, xmlns, http://xmlns.com/foaf/0.1/.

Buneman, P.; Fernandez, M.; and Suciu, D. 2000. UnQL: A query language and algebra for semistructured data based on structural recursion. *VLDB Journal* 9(1):76–110.

Funakoshi, K.; Kamei, K.; Yoshida, S.; and Kuwabara, K. 2001. Incorporating content-based collaborative filtering in a community support system. In *Intelligent Agents: Specification, Modeling, and Applications (PRIMA2001 Proceedings)*, number 2132 in LNAI, 198–209.

Huynh, D.; Karger, D.; and Quan, D. 2002. Haystack: A platform for creating, organizing and visualizing information using RDF. In *Semantic Web Workshop.*

Kamei, K.; Fujita, K.; Jettmar, E.; Yoshida, S.; and Kuwabara, K. 2002. Effectiveness of spatial representation in the formation of network communities: Experimental study on community organizer. *Interacting with Computer* 14(6):739–759.

Koivunen, M.-R., and Swick, R. 2001. Metadata based annotation infrastructure offers flexibility and extensibility for collaborative applications and beyond. In *KCAP 2001 workshop on knowledge markup and semantic annotation.*

Lassila, O., and Swich, R. R. 1999. Resource description framework (RDF) model and syntax specification. Technical report, W3C Recommendation, http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/.

Nejdl, W.; Siberski, W.; Simon, B.; and Tane, J. 2002. Towards a modification exchange language for distributed RDF repositories. In *ISWC2002*, number 2342 in LNCS, 236–249. Springer-Verlag.

Ohguro, T. 2001. Towards agents which are suggestive of "Awareness of Connectedness". *Trans. IEICE* E84-D(8):957–967.

Stojanovic, L., and Motik, B. 2002. Ontology evolution within ontology editors. In *EON2002 Evaluation of Ontology-based Tools*, 53–62.

Yoshida, S.; Kamei, K.; Ohguro, T.; and Kuwabara, K. (to appear). Shine: A peer-to-peer based framework of network community support systems. *Computer Communications.*