An Agent Framework for Inter-personal Information Sharing with an RDF-Based Repository

Koji Kamei, Sen Yoshida*, Kazuhiro Kuwabara**, Jun-ichi Akahani, and Tetsuji Satoh

NTT Communication Science Laboratories, NTT Corporation
2-4 Hikaridai, Seika-cho, Soraku-gun, Kyoto 619-0237 Japan
kamei@cslab.kecl.ntt.co.jp

Abstract. In this paper, we propose a framework for personal agents that supports inter-personal information exchange and sharing. The framework offers a repository-centered architecture in which a user can store information resources and annotations. On this framework, applications work collaboratively with the help of the event notification mechanism.

The personal agent especially targets the collecting and utilizing of information that describes how the user handled information resources. The descriptions of interactions are obtained by applications on the agent framework and stored into the RDF repository as annotations to the original information resources. A personal agent consists of applications that utilize these "annotations about interaction" for infortion sharing.

We first discuss "annotation about interaction" in personal environments, then describe the current implementation of our framework that offers an RDF-based repository. The experimental implementation of both communication-oriented and information-sharing-oriented applications on this framework is introduced as a starting point for our approach.

1 Introduction

In collaborative activity between individuals, sharing and exchanging information play important roles in communication. We receive a great deal of information from other individuals, interpret it and reflect on it using our own knowledge, after which we 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. 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 primarily include verbal annotations that users explicitly added to information resources, and in addition to them, descriptions of interaction between users and information. More specifically, the latter describes how one has handled a

* Presently with Research and Development Center, Nippon Telegraph and Telephone West Corporation.

** Presently with ATR Intelligent Robotics and Communication Laboratories.

D. Fensel et al. (Eds.): ISWC 2003, LNCS 2870, pp. 438–452, 2003.

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 both 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 organization becomes complicated, it is appropriate to use a schema-less, semi-structured database [1,2]. We have employed the resource description framework (RDF) [3] to describe these types of annotations about interaction. In this case, individuals and documents are mapped as *resources* in an RDF statement, and interactions are mapped as *predicates*.

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 [4] 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. Defining vocabularies about interaction is not only a challenge in ontology engineering, but also a challenge in information design [5]. In our approach, designing applications that help users to handle information resources is inseparable from defining vocabularies about information handling. In other words, defining ontologies about information handling and interaction will lead information designers to invent new applications.

In this paper, we describe an agent framework with an RDF-based personal repository. First, we discuss requirements for the personal repository and annotations in the personal environment, then describe the framework’s current implementation status. We also describe the experimental implementation of two types of existing applications on our framework as a starting point for this approach: CommunityOrganizer [6] and Gleams of People [7] for communication-oriented applications, Nakif [8,9] and KVP2P [10] for information-sharing-oriented ones.

2 Personal Agent for Supporting Interaction

2.1 Personalized Information Repository

The aim of the personal agent is to support information exchange and sharing between individuals. This type of agent is classified as an information agent according to the agent technology roadmap [11]. To achieve this, it is necessary that an individual can easily utilize information resources that he or she has handled. We focus on information storage that is deployed beside the user, storing any type of information resources the user has handled.

A personal repository is essentially a storage system that privately accumulates the information resources a user has handled. The repository stores not only information

resources such as documents, but also metadata as annotation to information resources. Here, metadata includes information about how the user handled information resources, since we are focusing on interaction between the user and information resources. The repository employs the resource description framework (RDF) to describe these semi-structured annotations.

The Haystack project is a well-known project that also employs semi-structured storage for accumulating and modeling personal information [12]. Haystack and our personal repository can share both the approach to back-end storage systems and the whole objective that the system supports personal information exchange. Both projects have recently focused on user interface matter; however, current interests seems to be different. Haystack is focusing on making use of RDF as a component architecture for reconfigurable user interfaces; that is, how to develop user interfaces [13]. Their prototype interface, named Ozone, is dynamically constructed at runtime, and the programming language, named Adenine, supports the definition of the user interface in RDF. On the other hand, our personal repository focuses on what type of metadata should be stored and watched while a user interacts with the user interfaces; that is, how to design user interface and user interaction. The system architecture needs to support interaction between the repository and user interface modules, and ontologies that can describe the metadata about interaction should be defined. There exist differences between current interests, although these approaches do not conflict with each other.

Our proposed agent framework is designed to support communication activities on this RDF-based personal repository. The framework offers an event notification mechanism to help implement services and applications that constitute a personal agent. Since multiple application modules run collaboratively on a personal agent framework and share information items and RDF statements in the repository, they need to be notified when the data that they are referring to change.

2.2 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 world-wide. 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 [14] 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. It does not, however, address the concept of private annotation servers and collaboration.

Nejdl *et al.* [15] discussed collaborative annotation as a part of EDUTELLA, which is a framework that allows 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 to be directed toward coherency rather than individuality.

2.3 Design Problem in Personal Agents

Annotations are usually added by hand. To add annotations about information handling, however, it is desirable that annotations are added automatically by applications according to operations the user executed on those applications.

Two problems arise when dealing with such applications. One is to define vocabularies for describing interaction between individuals and information resources. This relates to the ontology definition problem in the Semantic Web. The other is to design applications that lead users to handle information resources along the defined vocabularies. The latter is covered by researches into both information design and interaction design.

Those two problems are mutually dependent and should be solved simultaneously; that is, information designers need to consider the defining ontology of interaction. To aid this design process, we propose a personal agent framework with an RDF-based personal repository that accumulates information resources and annotations, especially with vocabulary about interaction.

2.4 Information Sharing between Personal Environments

In the environment described above, information resources are assumed to be stored locally beside the user, and gradually shared through peer-to-peer information exchange.

Information 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, for example, *forwarded* and *introduced* can be included in information-sharing vocabularies.

Exchanging and sharing information is obviously an interaction between individuals, but at the same time, it is also an interaction between an individual and information resources. Keeping track of this kind of interaction is necessary to design applications for smooth information sharing and exchange.

3 Personal Repository

In this section, we describe the architecture and implementation of a personal repository that enables and utilizes the personal annotations.

A personal agent exists beside each user to support his or her activities, and a personal repository occupies the center of our personal agent framework; that is, a personal

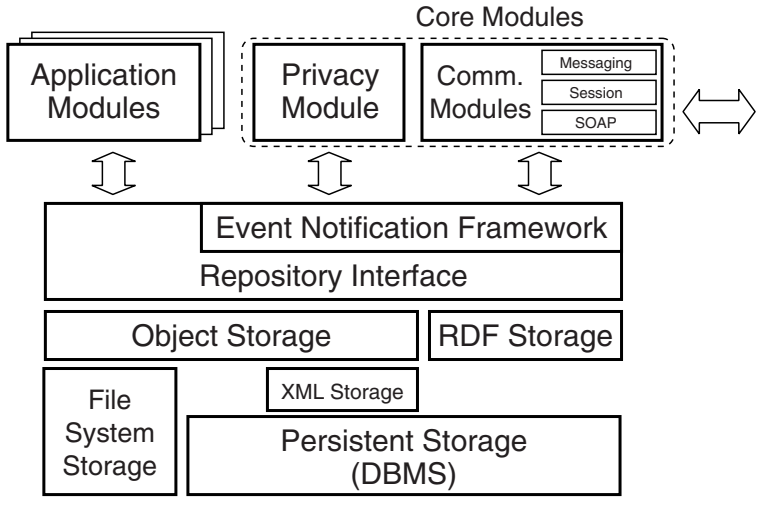

Fig. 1. Architecture of the personal repository

repository exists for each user and accumulates information and annotations that its owner has handled. A personal agent consists of groups of various programs that store and make use of information organized upon the personal repository.

The personal repository is currently implemented in a Java2 SDK 1.4 environment, and employs various open-source products relating to Web and XML technologies. The whole system architecture is shown in Figure 1. We first describe three core modules of the personal repository, then address its programming interface and a visual interface.

3.1 Back-End Storage

The personal repository needs to support the features of an RDF database, since relationships between resources can be described as RDF statements. Additionally, an auxiliary information storage 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.

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

The RDF storage is implemented on a Jena Semantic Web toolkit [1]. For the back-end storage of Jena, PostgreSQL [2] is used for persistency. Objects are stored in several sub-storages. Messages that encoded into XML format are stored in an XML storage that

[1] http://www.hpl.hp.com/semweb/jena-top.html
[2] http://www.postgresql.org/

utilizes eXist[3], which implements XML:DB API [4] and shares the back-end PostgreSQL system with the RDF storage. Images and HTML files are stored into a file-system-based storage for ease of browsing from Web browsers. Other objects are currently stored in the PostgreSQL as text data.

3.2 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 have changed. An event notification framework enables application modules to collaborate with each other. That is, personal agent consist of multiple application modules that are activated by the event notification framework on the repository. The repository contains a group of core modules that are started when the repository is invoked. In addition, 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 *write* 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 framework is built upon both types of storage described in the previous section. This feature is based on the concept of the active database [16]. Referring to the ECA model, this framework monitors *write* operations to the storage as an *event*. It then 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. Each module registers a template pattern for each *condition* from which it desires to be notified of the change. The current implementation of the personal repository provides Java level API for event notification framework, therefore, the template pattern (condition) should be represented as a Java object. The syntax of the template pattern for the XML repository is defined as an XML element and represented as an instance of `org.w3c.dom.Element` class. For the RDF repository, the template is described as an RDF statement with wildcards. The `Template` class resembles to the `Statement` interface in Jena, but allows to set null value for each element (subject, predicate and object) to represent the wildcard value.

When the *condition* is satisfied, the *action* clause is processed. Currently the action clause should be written in Java as a callback function. Overall behavior of the agent and applications is realized as results of the interaction among those modules invoked by changes of data in the repository. The coordination between user interface modules and the repository is also implemented in this callback method.

The repository is considered as a type of a shared blackboard in the framework since a group of programs works collaboratively for the owner, although it does not offer any layered structure, unlike the (distributed) blackboard model [17]. The layers or structures will be defined as data by RDF statements when some program requires such kinds of structures.

[3] http://exist-db.org/
[4] http://www.xmldb.org/xapi/index.html

The event notification framework can also be compared with tuple-space-based systems such as Linda [18], especially while watching RDF statements. Our framework offers both a blocking and a non-blocking *read* operation. After a *read* operation has been processed, the matched element remains in the repository since it is not a discrete tuple but a part of organized structure of information. If it is necessary to remove some elements, users need to give instructions to remove them in the action clause of the rule.

3.3 Communication between Personal Agents

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 other known agents, and in actual cases of group communication, messages may also be exchanged in a server-oriented manner.

The repository keeps acquintances' information, that is, mappings between the acquintance's name and his or her agent's address (a URL for inter-agent communication). The system does not actually define any hard limit to the number of entries, although the address list might usually contain handleds of users since this framework supports information exchange between acquintances. It does not mean that the system only supports communication in a closed community, because users can forward information from one acquintance to another, just like in our real-world experience. Information exchange between more than a thousand of repositories will be realized by multi-hop communication.

The address list is currently maintained in two ways. A user can maintain the list entries by hand through applications' user interfaces such as Gleams of People (described below). Another is the tentative use of peer discovery in a LAN, since acquintances with whom a user closely shares information exist in the same LAN. The current implementation of this function is implemented as a module upon the repository with Rendezvous technology[5]. It acquires mappings between an acquintance's name and its URL from Rendezvous, then updates information in the repository. Peer discovery from outside a LAN has not been implemented yet; however, it can be implemented as a similar module that refers global white/yellow pages and subsequently updates the repository.

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 [6], 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. Although 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 [7] SOAP library and deployed to Tomcat [8] servlet containers. Since inter-repository communication is performed in a

[5] http://developer.apple.com/darwin/projects/rendezvous/
[6] http://www.w3.org/TR/soap12-part0/
[7] http://xml.apache.org/axis/
[8] http://jakarta.apache.org/tomcat/

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 request other agents to notify changes in their repositories by remotely registering notification conditions described in the previous section. When a listener registration request arrives from another agent on the communication layer, a listener proxy is created and registered to the repository. When the proxy listener is fired from the repository, the proxy listener composes a message that describes the event, and sends it to the remote listener. A privacy management module is deployed to protect privacy. This module monitors the registration of condition templates from other repositories and decides whether to acccept the registration of the listener.

3.4 Programming Interface

The programming interface to the repository is currently defined at the Java API level. Since it is a type of storage system, modification and query are primitive APIs. It is obviously necessary to define a query language independent from low-level APIs. We are currently in the process of designing a repository query language that supports not only the query/update function, but also our event notification framework. We also plan to implement a script engine for the language as a service on the event notification layer.

3.5 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 forms: graphic representation in a two-dimensional plane, and list-like representation.

The graphical view in Figure 2a shows the resource node to which a user is paying attention located at the center of the view, while 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. A list of all subject nodes is also provided to support direct jumps to other nodes.

Although its current representation is limited, this tool is helpful in developing our application because it allows us to inspect and modify the items in the repository while in operation. Since the tool also monitors the repository's event notification framework, the the changes in the repository are immediately reflected in the view.

4 Existing Applications of Repository

In this section, we introduce several applications being implemented on the personal agent framework. Two types of applications are described: first, two communication-

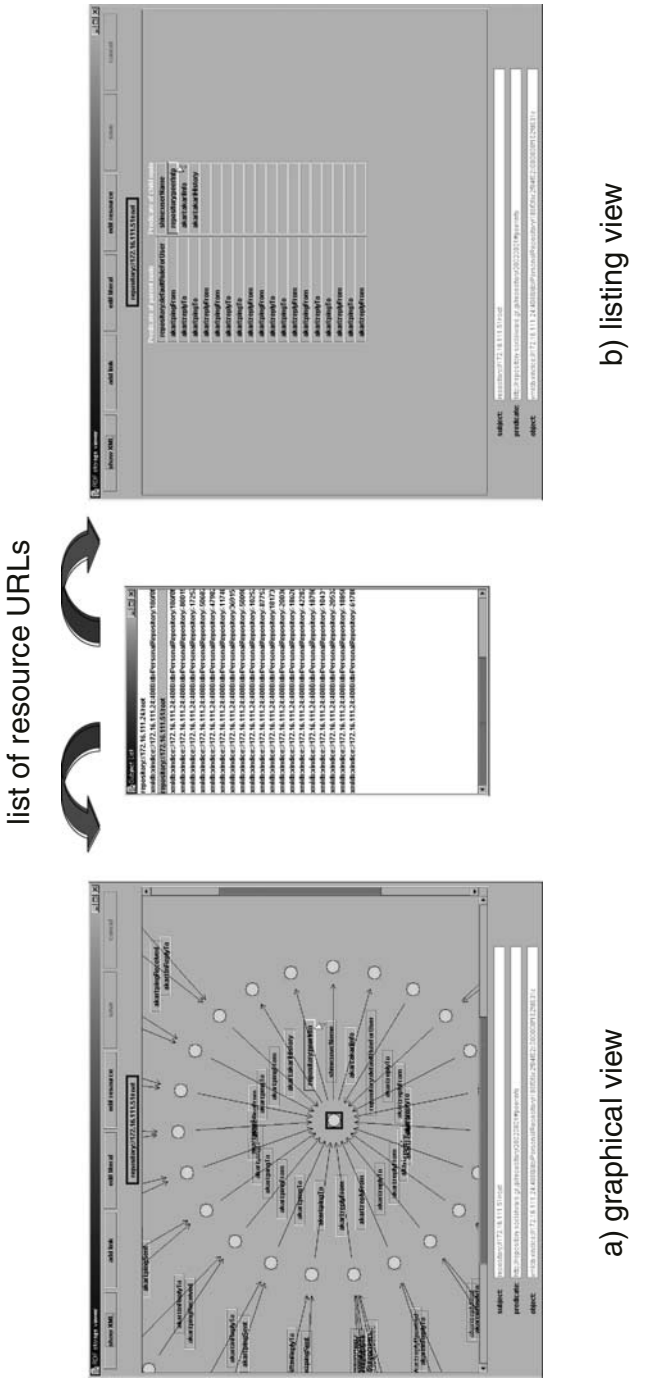

Fig. 2. The screen image of the repository viewer

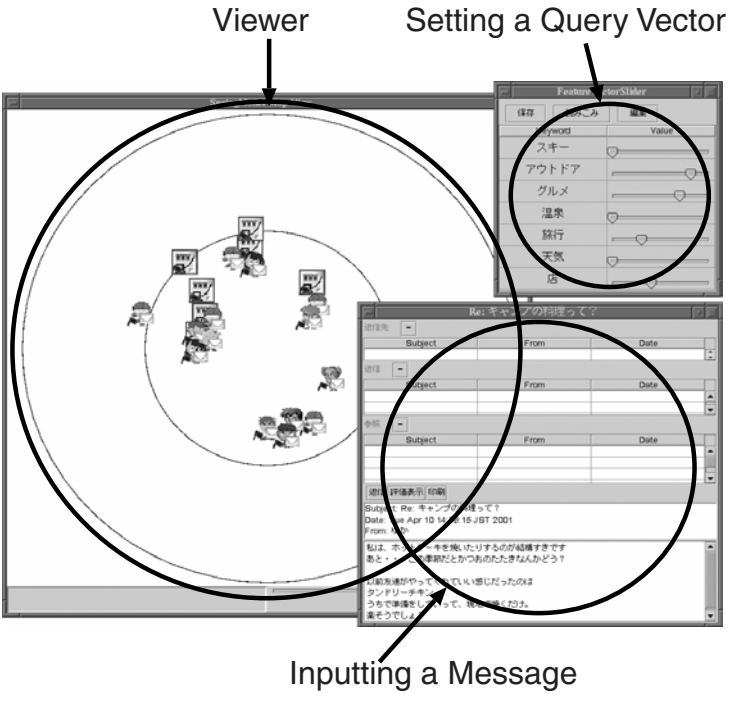

Fig. 3. CommunityOrganizer (screen image)

oriented applications, CommunityOrganizer and Gleams of People, then two information-sharing-oriented applications, Nakif and KVP2P.

4.1 CommunityOrganizer

CommunityOrganizer provides a shared space where public short messages and Web page introductions are exchanged, and in which people are made aware of possible communities of interests. 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. Users can also introduce resources on the Web to their communities. In this case, the URI of the resource is described instead of verbal messages.

Figure 3 shows the primary component of CommunityOrganizer's user interface, on which 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, and the view is updated when new information items are added, enabling users to become aware of the dynamic formation of information clusters.

The event notification framework of the personal repository notifies the application module of messages' arrival to update the visualization. In other words, when several

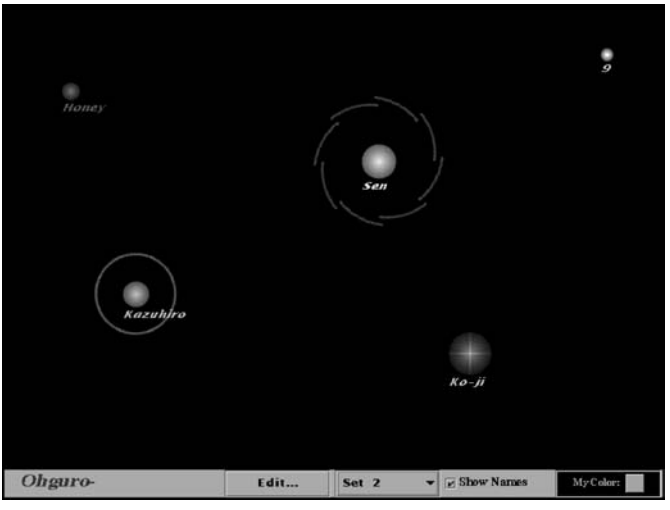

Fig. 4. Gleams of People (screen image)

messages concerning a topic are posted in a short period, the messages (and relevant resources) move to create an animated cluster.

To calculate the relevance between the view and resources, a feature vector is added for each resource to represent the content. The vector is described as an XML document consisting of a set of keyword-value pairs, and stored as an annotation (linked with the *content feature* predicate) to the original message.

4.2 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 simply intended for keeping in touch [7]. 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 end, 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 still worth expressing*.

The spheres indicate presence and status information of corresponding users at a glance, and also unobtrusively indicate the history of communication activities; 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 repository, 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.

4.3 Nakif

Nakif is a hybrid filtering technique that incorporates a content-based method into a collaborative filtering system, thus it provides high-quality evaluations of communities of interests [8].

When a user evaluates a document in Nakif, that is, assigns a *positive* or *negative* score to the document, 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 evaluated 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*.

When an information resource or an evaluation of a resource arrives from another user, Nakif will be invoked by the repository. It then locally evaluates the resource (arrived or evaluated) and stores the calculated score as an annotation to the resource. The change will influence other applications, such as CommunityOrganizer.

One alternative application of Nakif is that a personal agent can automatically forward a newly received (or evaluated) information resource to other agents. With the user profile matrix described above, Nakif can estimate users who will evaluate the document positively. This inversed utilization of original Nakif enables information sharing with two-way information filtering.

4.4 KVP2P

KVP2P [10] appends two features to the personal agent. One is a personalized information retrieval method with a personalized concept-base, while the other is a method for collaborative information retrieval between personal repositories. A personal repository accumulates each user's information and its structure; therefore, it reflects its owner's interests and communication activities. Because such a system requires an IR mechanism as one of its basic functionalities, we employed a concept-based IR system as a search engine for personal repositories.

The "concept base" is an extended vector space model for information retrieval [19] that is grounded in co-occurrence to solve the above problem. This method first counts the word co-occurrences that occur in proximity in the documents, then constructs a word co-occurrence matrix. Second, it reduces the rank of the matrix using singular value decomposition (SVD) to yield the keywords' vector space. The vector for a document is represented as the sum of the keyword vectors generated from it, and in this method, documents having similar content have strong relevance even if the documents do not use the same expressions. Since the personal repository contains information resources that accurately describe its owner's interests, we have proposed a method for constructing a personalized co-occurrence-based concept-base on the personal repository [20].

The personalized concept-base facilitates information retrieval in a personal repository. However, a new problem arises when querying to other users' personal repositories: since the documents in personal repositories differ from each other, the derived concept-base for each repository also differs. That is, simply transferring query requests does not provide desirable results. To solve this problem, KVP2P offers an IR function that

searches for documents in equalized vector spaces. A scheme for vector space equalization, the automated relevance feedback scheme, is proposed and implemented as a service on the repository-based personal agent framework. The idea is derived from the technique known as relevance feedback. Conventional relevance feedback is a human-to-computer feedback technique in which a human judges whether documents retrieved by a computer are desirable. On the other hand, in the proposed scheme, the caller's personal agent can judge the adequacy of the documents by using the concept-base to establish the feedback loop automatically.

4.5 Discussion

The applications described above have, except for KVP2P, developed once prior to the personal repository, and have been experimentally integrated. The experience influenced the design of personal repository architecture, which is a common framework based on a semi-structured and active database.

Shine, for example, is a peer-to-peer agent framework that provides basic vocabularies about agents, users, communication and communities [21]. Community Organizer and Gleams of People have been integrated on it. The vocabularies defined in Shine provide a start point for defining vocabularies in this personal agent frameworks. In 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*.

Another example is the integration of Community Organizer and Nakif [8]. The implementation has changed the information sharing scheme from a server-centered model to a partially distributed model. Here, information sharing is achieved not merely by forwarding all the information obtained, but by filtering the information to be forwarded based on the metadata that describes both the evaluation and the feature of the content. Although the efficiency of collaboration filtering depends on the quality of the content feature vector, in the previous implementation of Nakif the content feature of the document was represented by word frequency. By integrating with personalized concept-base in KVP2P, the efficiency of collaborative filtering is expected to be improved.

5 Summary and Further Issues

In this paper, we described a framework for a communication support agent. In this framework, one personal agent exists for each user and supports his or her communication activities, especially information exchange and sharing. We considered collaboration between individuals as the results of shared annotations. It leads us to propose an RDF-based personal repository framework, the need for vocabularies about interaction, and information design with those vocabularies.

We described the current design and implementation of the personal repository onto which users can store information resources, personally create annotations to those resources, and with which they can exchange and share information resources and annotations. We also described four existing applications being re-implemented on the personal

agent framework. They indicated example usages of our repository-based personal agent framework, and direction to define vocabularies for describing interaction.

The next step will require us to add ontology mapping and an approximation mechanism [22] on top of our framework, since application designers will themselves define or improve vocabularies. Vocabularies in this paper are extracted from existing weakly-tied applications, and as a result, the relationship between vocabularies is weakly defined. To support an ontology mapping mechanism, more investigation is required to precisely define the relationship between vocabularies. Future work will also include extracting base classes or properties for ontology structures, and defining causal relationships between vocabularies.

Acknowledgments. The concept of a personal repository arose 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. KVP2P is a research result from collaboration with Dr. Takashi Yukawa at Nagaoka University of Technology. In addition, we wish to thank Mr. Takashi Kawachi and Mr. Koji Saito for the implementation of the personal repository framework and applications.

References

1. Abiteboul, S., Quass, D., McHugh, J., Widom, J., Wiener, J.L.: The Lorel query language for semistructured data. International Journal on Digital Libraries **1** (1997) 68–88
2. Buneman, P., Fernandez, M., Suciu, D.: UnQL: A query language and algebra for semistructured data based on structural recursion. VLDB Journal **9** (2000) 76–110
3. Lassila, O., Swich, R.R.: Resource description framework (RDF) model and syntax specification. Technical report, W3C Recommendation, http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/ (1999)
4. Stojanovic, L., Motik, B.: Ontology evolution within ontology editors. In: EON2002 Evaluation of Ontology-based Tools. (2002) 53–62
5. Jacobson, R., ed.: Information Design. MIT Press (1999)
6. Kamei, K., Fujita, K., Jettmar, E., Yoshida, S., Kuwabara, K.: Effectiveness of spatial representation in the formation of network communities: Experimental study on community organizer. Interacting with Computer **14** (2002) 739–759
7. Ohguro, T.: Towards agents which are suggestive of "Awareness of Connectedness". Trans. IEICE **E84-D** (2001) 957–967
8. Funakoshi, K., Kamei, K., Yoshida, S., Kuwabara, K.: Incorporating content-based collaborative filtering in a community support system. In: Intelligent Agents: Specification, Modeling, and Applications (PRIMA2001 Proceedings). Number 2132 in LNAI (2001) 198–209
9. Funakoshi, K., Ohguro, T.: Evaluation of integrated content-based collaborative filtering. In: 2001 ACM SIGIR Workshop on Recommender Systems. (2001)
10. Yukawa, T., Yoshida, S., Kuwabara, K.: A vector space equalization scheme for a concept-based collaborative information retrieval system. In: Proc. of 16th International FLAIRS Conference. (2003)
11. Luck, M., McBurney, P., Preist, C.: Agent Technology: Enabling Next Generation Computing. AgentLink, http://www.agentlink.org/roadmap/index.html (2003)
12. Adar, E., Karger, D.R., Stein, L.A.: Haystack: Per-user information environments. In: Proc. of 8th International Conference on Information and Knowledge Management (CIKM1999). (1999) 413–422

13. Huynh, D., Karger, D., Quan, D.: Haystack: A platform for creating, organizing and visualizing information using RDF. In: Semantic Web Workshop. (2002)
14. Koivunen, M.R., Swick, R.: Metadata based annotation infrastructure offers flexibility and extensibility for collaborative applications and beyond. In: KCAP 2001 workshop on knowledge markup and semantic annotation. (2001)
15. Nejdl, W., Siberski, W., Simon, B., Tane, J.: Towards a modification exchange language for distributed RDF repositories. In: ISWC2002. Number 2342 in LNCS, Springer-Verlag (2002) 236–249
16. Widom, J., Ceri, S.: Active Database Systems. Morgan Kaufmann (1996)
17. Hayes-Roth, B.: A blackboard architecture for control. Artificial Intelligence **26** (1985) 251–321
18. Gelernter, D.: Generative communication in Linda. ACM Trans. on Programming Languages and Systems **7** (1985) 80–112
19. Schütze, H., Pedersen, J.O.: A cooccurrence-based thesaurus and two applications to information retrieval. Information Processing & Management **33** (1997) 307–318
20. Yoshida, S., Yukawa, T., Kuwabara, K.: Constructing and examining personalized cooccurrence-based thesauri on Web pages. In: Proc. WWW2003. (2003)
21. Yoshida, S., Kamei, K., Ohguro, T., Kuwabara, K.: Shine: A peer-to-peer based framework of network community support systems. Computer Communications (2003) 1199–1209
22. Akahani, J., Hiramatsu, K., Satoh, T.: Approximate query reformulation based on hierarchical ontology mapping. In: Proc. International Workshop on Semantic Web Foundations and Application Technologies. (2003) 43–46