Building a Network Community Support System on the Multi-Agent Platform Shine

Sen Yoshida, Koji Kamei, Takeshi Ohguro, Kazuhiro Kuwabara, and Kaname Funakoshi

NTT Communication Science Laboratories
{yoshida,kamei,ohguro,kuwabara,kf}@cslab.kecl.ntt.co.jp

Abstract. An increasing number of applications have been developed for supporting network communities. The authors have developed *Community Organizer*, which supports people in forming new network communities by providing places where people sharing interests and concerns can meet and communicate. The authors are also developing a platform named Shine to reduce the tasks needed to implement a variety of network community support systems such as Community Organizer. Shine has a multi-agent architecture because it is effective for network community support systems that have to adapt to dynamic changes in community organizations. This paper explains both Community Organizer and Shine, and then gives a description of building Community Organizer on top of Shine.

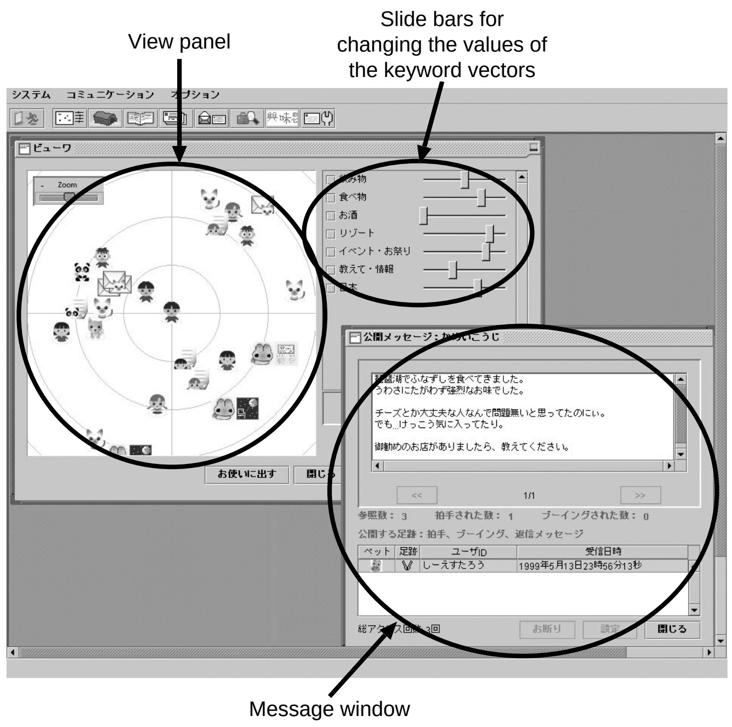
1 Introduction

With the advance of public communication networks such as the Internet, a new type of community, called a network community or a virtual community [18], is beginning to emerge on the networks. Members in a network community actively and intimately communicate with each other by using e-mail, electronic chat rooms, bulletin board systems, and so on. Furthermore, a multitude of projects are devoted to developing systems to support communications especially in network communities [7]. These systems are called *socialware* [5].

The authors have developed *Community Organizer* to support people in forming new network communities by providing places where people sharing interests and concerns can meet and communicate.

The authors are also developing a platform named Shine [23] to reduce the tasks needed to implement a variety of socialware application programs such as Community Organizer. Shine has a multi-agent architecture because it is effective for socialware that has to adapt to dynamic changes in community organizations.

We explain Community Organizer in Section 2 and Shine in Section 3. We then give a description of building Community Organizer on top of Shine in Section 4.



琵琶湖でふなずしを食べてきました。
うわさにたがわず強烈なお味でした。

チーズとか大丈夫な人なんで問題無いと思ってたのにぃ。
でも…けっこう気に入ってたり。

御勧めのお店がありましたら、教えてください。

参照数：3　拍手された数：1　ブーイングされた数：0

公開する足跡：拍手、ブーイング、返信メッセージ

ペット	足跡	ユーザID	受信日時
	V	しーえすたろう	1999年5月13日23時56分13秒

総アクセス回数：3回

Fig. 1. Sample screenshot of Community Organizer (Japanese version)

2 Community Organizer

Unlike traditional communities where geographical and institutional properties define the boundary of the community, most network communities are *communities of interest* [6], which consist of people who share common interests. Considering this characteristic of network communities, Community Organizer regards people sharing interests as members of a potential community and provides them with a place where they can meet and communicate.

Figure 1 is a sample screenshot of Community Organizer. Community Organizer displays representations of potential community members and communications among them.

Community Organizer has as its main feature a view panel that visualizes the relationships among users in real-time [5, 22]. A user is represented as an icon in the view panel. Its placement in relation to others reflects the closeness of users' interests. A user's interest is represented as a keyword vector, which is a set of pairs of a keyword (e.g., 'travel' or 'foods') and a numerical value (i.e., the degree of his/her interest in the keyword). The system calculates the degree of closeness for each pair of keyword vectors from the cosine measure

[19] and places icons on the view panel using a physically-based model adopted from Ref. [1]. As a result, icons with similar keyword vectors are placed closer to each other. By observing how icons are distributed, a user can envision potential communities of interest.

The center point of the view can be set by the user. The center point has its keyword vector too, and the user can determine its values by adjusting the slide bars next to the view panel. The placement of the icons will change according to the positions of the slide bars. This function enables users to look around the display to find potential communities.

In addition, a user can easily express his/her interest to other users by placing a new icon representing the user in the center of the view panel. The newly placed icon has the same keyword vector as that of the center point. Consequently, it will appear on the display of other users whose slide bars' positions are similar to those of the placing user. A user can place multiple icons on different positions, which allows him/her to participate in multiple communities.

Community Organizer integrates a communication function with this visualization function. This is done by providing an interface that allows a user to attach his/her message to an icon. Other users can read the message by clicking the icon, and reply to it by placing a new icon on a closer position and attaching a reply message to it. This integration of the visualization function and the communication function produces a utility that is quite suitable for members of a community of interest.

We have conducted a psychological experiment to explore the effect of Community Organizer's spatial representation [8]. The results indicate that this two-dimensional representation enhances community feelings of the users more than one-dimensional representation that simply displays a list of users in decreasing order of the closeness.

3 Shine: A Multi-Agent Platform for Network Community Support Systems

In this section, we explain Shine, a multi-agent platform for network community support systems that the authors are developing [23].

It is easy to think of many applications for supporting network communities. There are applications that support exploitation of the merits of forming communities. Such applications include:

- Circulation of information via word-of-mouth human networks, such as recommendations and collaborative filtering [20];
- Arrangement of informal, suggestive or creative conversations, such as brainstorming sessions [4, 11];
- Establishment of efficient collaborations for developing resources such as open source software or databases; and
- Arrangement of localized and informal economic activities like auctions [10] and flea markets.

There are also applications that support community formation, maintenance, and evolution. For example:

- Arrangement of effective encounters and support for becoming aware of potential communities [15, 22];
- Maintenance of intimacy and community feelings [17];
- Coordination of conversations [12, 13] and collaborations; and
- Planning for the establishment of individual identities in communities.

While we can think of a variety of network community applications, certain functions are commonly required by these applications. These functions include:

- Dynamic adaptation to acquaintance relations or group formation;
- Analysis of each person's features, role, and situation within a community;
- Analysis of a community's comprehensive features;
- An interface that links a user's community feeling to a system's logical information; and
- Flexible communication utilities.

At present, however, there is no platform that can provide these common functions. Consequently, most network community support systems are developed as independent systems. Therefore, there is no cooperation among application programs or sharing and reuse of program components. For this motivation, we are developing a platform named Shine as a common base for various network community applications.

In the following subsections, we examine the design of the platform's architecture.

3.1 Multi-agents for Network Community Applications

A network community has the advantage of providing communication capabilities to an actual community that are open and free from geographical or temporal constraints. However, this openness brings the difficulty of gate keeping. Because people are unable to deal with too much information or too many other humans, they tend to lose the proper perspective for acting or hesitate to act and end up isolated in a huge mass of strangers.

Unfortunately, most of the network community support systems available today do not attempt to tackle this gate keeping problem. They were designed as client-server (CS) systems, which assume that a single host machine serves thousands or millions of clients accessing from anywhere on the Internet. In this architecture, the server program holds a huge database, in which the personal information of all users is stored and from which it is retrieved. The quantity of such a system's users group is beyond the human capability to develop community feeling. Or otherwise, in case there were a lot of sites that serves small number of people, each user's chance to enlarge his/her acquaintance relation is restricted.

In real society, people meet unknown others by mediating or introducing themselves. Consequently, they flexibly change their acquaintance relations and

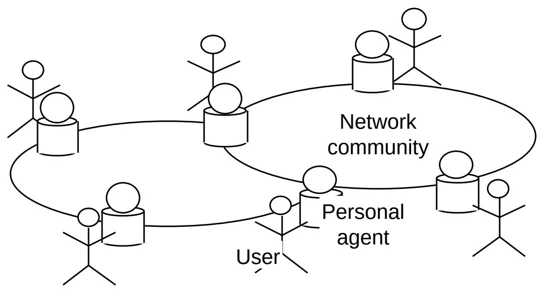


Fig. 2. Personal agents

reorganize their communities. Humans do these tasks of gate keeping in a quite decentralized and cooperative way. This decentralized gate keeping mechanism used in real society can also work efficiently in network communities. Therefore, in Shine, we adopt a multi-agent architecture.

In our approach, the control mechanisms and data for human relations are distributed to each person. Namely, the Shine system provides each user an agent that is dedicated to the person's social interactions as shown in Fig 2. A Shine agent exchanges personal information with other agents, and they all collaboratively perform the necessary actions such as mediation or self-introduction to dynamically change community formation.

When we introduce mediation mechanism to Shine, we must consider privacy control mechanism. For example, when agent *a* tells a private issue to agent *b*, and *a* wants *b* not to tell the issue to anyone else, it must be forbidden for *b* to tell it to anyone. Shine will provide this kind of privacy control mechanism in future.

3.2 Internal Structure of a Shine Agent

Figure 3 shows an overview of a Shine agent's internal structure. In a Shine agent, there are several modules, i.e., Person Database, Planner, Post, and the user interfaces of application programs.

Person Database The Person Database holds data on others and the user whom the agent is associated with. These data are appropriately circulated via communication among agents.

To analyze a person's features, role, or situation in a community systematically, a network community application has to adopt a kind of user model. By following a specific user model, the application can store data on the attributes and status of a person into a database and then use the data for analysis.

One of a number of user models can be used depending on how it regards a human. For example, when a user model considers a human to be a knowledge source, it can describe the person in a knowledge representation language [16].

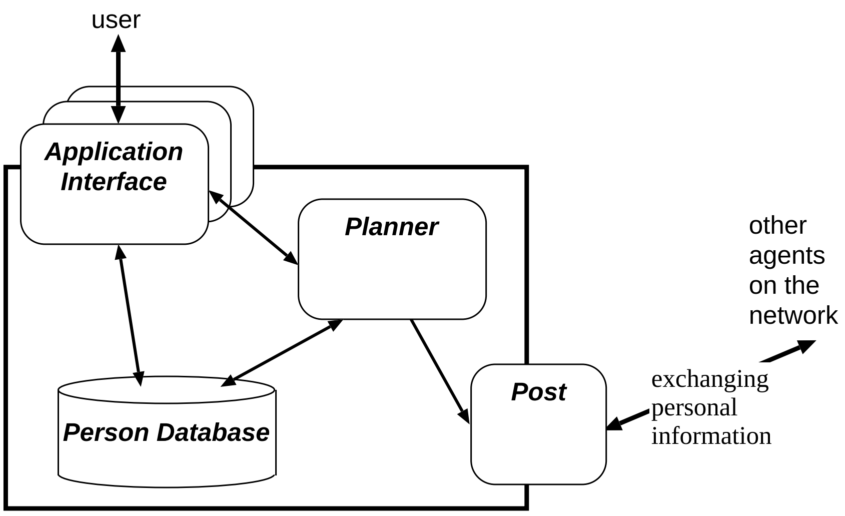


Fig. 3. Internal structure of Shine agent

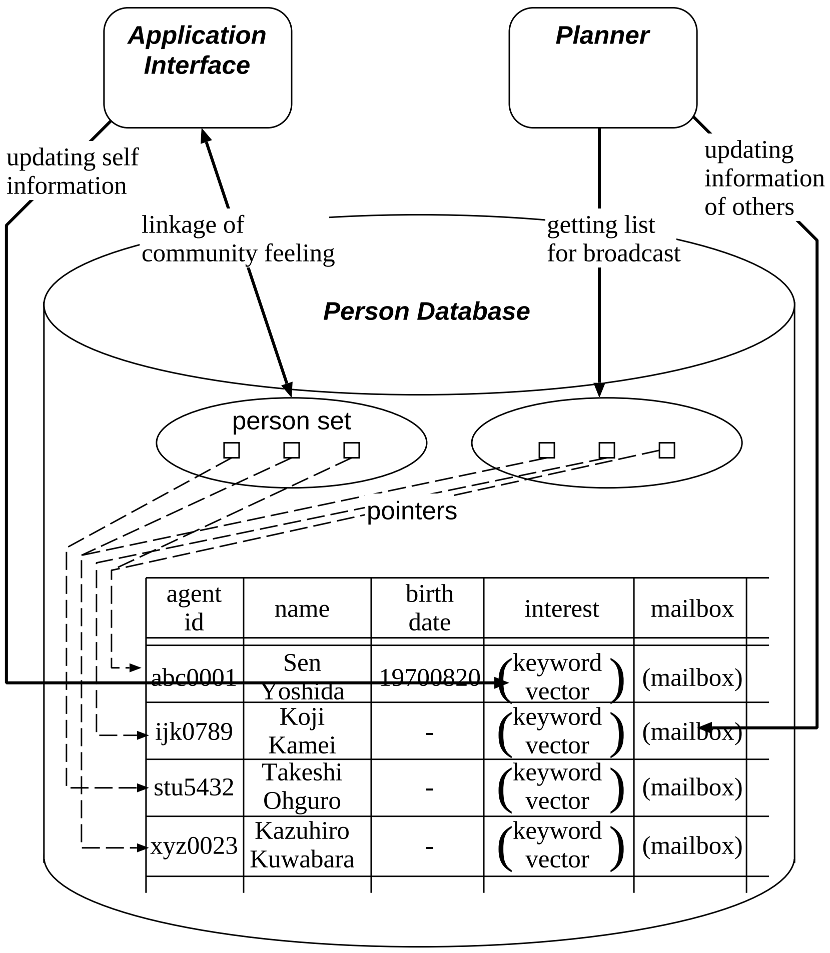
To roughly represent a person's interests, keyword vectors of the vector space model are suitable [22]. In addition, there is a method that describes communication entities as state transition systems to model each person's situation and conversation flow [21].

Each current network community support system uses a suitable user model. For Shine to achieve cooperation among application programs and the sharing of software components in application development, it has a generic architecture that can process various user models in a unified way. Moreover, it can handle multiple user models simultaneously. An application programmer makes his/her own user model in a style that Shine accepts and installs it in the Shine system. Shine also provides a library of popular user models such as the vector space model.

Practically speaking, Person Database is an object-oriented relational database (Fig. 4). Persons are stored in a table, which has a number of attributes concerning the user models used. For example, 'agent id,' 'name,' 'birth date,' 'interest,' etc. are thought of. Each line of the table represents a person and the agent that the person is associated with.

Some attributes of the table are *key* attributes used for identifying a person from a value. For instance, 'agent id' is a key attribute, and its values are used as the addresses for inter-agent communication.

We adopt an object-oriented database so that we can store not only simple strings or numeric values but also structured objects into a table. For example, we can make an 'interest' attribute whose data are keyword vectors, i.e., sets of pairs of a keyword string and a numeric value. We can also make a 'mailbox' attribute, whose data are collections of mails received from corresponding persons.


Application Interface
Planner
updating self information
linkage of community feeling
getting list for broadcast
updating information of others
Person Database
person set
pointers

agent id	name	birth date	interest	mailbox
abc0001	Sen Yoshida	19700820	(keyword vector)	(mailbox)
ijk0789	Koji Kamei	-	(keyword vector)	(mailbox)
stu5432	Takeshi Ohguro	-	(keyword vector)	(mailbox)
xyz0023	Kazuhiro Kuwabara	-	(keyword vector)	(mailbox)

Fig. 4. Person Database

Person Database has an event generation function. When an attribute value of a person is added, changed, or removed, an event occurs. Other modules such as Planner observe occurrences of these events, and do appropriate actions in response to them. This event notification mechanism enables cooperation among multiple modules and federation of multiple applications.

In most multi-agent systems, although an agent holds and handles information on other agents, it isn't conscious that a set of agents, or people whom the personal agents are associated with, comprise a community. A network community support system, however, always uses information on the community. Therefore, a Shine agent needs a flexible framework for dealing with information on not only each agent but also a community of agents.

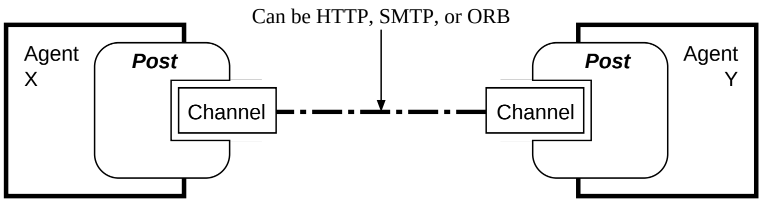


Fig. 5. Channel

To handle communities, a Shine agent defines for each community a *person set*, and provides operations for dealing with a set of personal information. A person set observes each member's data stored in the table for adjusting to dynamic changes in the organization of the real community.

When we use a network community support system to determine whom to communicate with, especially when we want to broadcast a message to the members of the community, it is essential that the communications adapt to all dynamic changes in the community formation and be linked to the community feelings of users. Shine can provide a function that can flexibly determine the range of group communication because it uses person sets of Person Database as destination lists for broadcasting.

Post The Post module exchanges messages with other agents. It abstracts lower-level communication procedures as an internal channel submodule (Fig. 5). Specifically, it encodes and decodes messages to adjust the lower-level communication protocol, e.g., HTTP, SMTP, or ORB. The channel submodule is hidden from other modules or application programmers.

The format of the messages is similar to KQML [2] or other agent communication languages. Each message has a message type, a user model indication, and contents described following the indicated user model. A user model in Shine corresponds to an ontology in KQML.

Planner The Planner decides the action of the agent. Figure 6 shows the structure of the Planner. The agent acts in response to external events such as messages received from other agents, inputs from the user, and changes of values stored in the Person Database.

Each application has one or more plans, and the Planner executes them concurrently. There are also plans provided by Shine that do fundamental or common tasks. The Planner contains a message dispatcher, which dispatches each incoming message to appropriate plans according to message type. Plans can cooperate via Person Database: When a plan changes an attribute value of a person, another plan can perceive the change by catching the event occurred according to it.

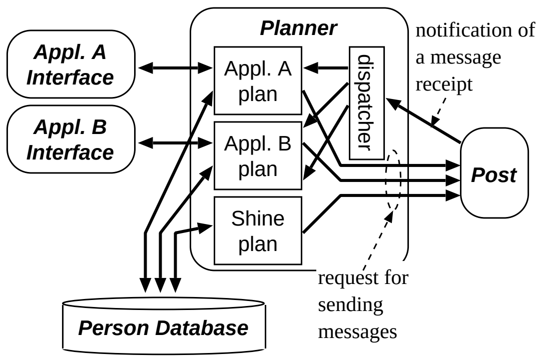


Fig. 6. Planner

Application Interface The application interface connects the user with the Person Database and the Planner. There can exist multiple applications on one Shine agent, and each application has its own interface module.

A network community support system needs a user interface that allows users to relate their feelings about acquaintances or communities to the system and to understand intuitively the meaning of the logical information stored in the Database in accordance with a user model. Therefore, Shine not only provides popular user models but also interface libraries for those models. Such interface libraries of user models include visualization mechanisms and other mechanisms that use available human-computer interaction media.

The application interface also helps interactions between the user and the Planner by providing dialogue messages and other aids.

3.3 Comparisons with Other Multi-agent Platforms

MINDS [14] is a multi-agent system for collaborative document retrieval. In this system, each personal agent knows documents stored in its user's workstation, and also has metaknowledge about who possibly owns documents about what topic. When a user asks his/her agent for retrieving documents about a topic, the agent intelligently submits queries to other agents using this metaknowledge. MINDS is one of pioneers in collaborative personal agents, but it's domain is limited to information retrieval.

KQML [2] specifies a message format and a set of message handling protocols that are suitable for knowledge sharing among agents. There are a number multi-agent platforms that implement this specification. KQML can transfer various kinds of knowledges because it provides a function to select an ontology based on the domain of each knowledge. However, in KQML framework, agents don't have capabilities to dynamically reorganize their social relation, for example by mediation. To perform such a dynamic reorganization, an agent has to be able to exchange not only knowledge about the application domain but also information about the agent itself or its acquaintances. Furthermore, this reorganization must be linked to the change of the user's community feeling. Therefore, Shine includes human-computer interface libraries besides knowledge exchange functions.

The ADIPS Framework [9] is a platform for flexible distributed systems. Several network applications such as a flexible video conference system, a flexible asynchronous messaging system, and a system named CyberOffice have been developed on top of the ADIPS Framework. The main feature of the ADIPS Framework is that each node of an ADIPS application system can be flexibly and dynamically reconfigured to adapt to changes in its environment such as QoS or the user's preference. To achieve this adaptability, each module of an ADIPS system's node is agentified.

The difference between ADIPS and Shine is the policy of agentification. The ADIPS system agentifies the modules in a node to obtain adaptability to their environment. On the other hand, we call a node of Shine an agent because it is personalized and communicates with other personal agents to adapt to changes in the user's social situation.

4 Community Organizer on Shine

This section gives a description of revising Community Organizer by porting it to the Shine platform. This revision brings certain advantages to Community Organizer.

- The implementation of Community Organizer can make use of the libraries of program components included in the Shine package. What we need to make for Shine-based Community Organizer are only a plan installed to the Planner, a user interface, and some additional attributes for the Person Database.
- The architecture of the system changes from the conventional client/server architecture to a more flexible multi-agent architecture.
- It is possible to integrate Community Organizer with other network community applications by sharing data on people with those applications.

In the Community Organizer, each icon's feature is represented as a keyword vector. In other words, the Community Organizer adopts the vector space model of an information retrieval technique as the user model. Because Shine provides a library for this model, the program of the Community Organizer can make use of prepared methods such as the cosine measure [19]. Furthermore, the user interface of the Community Organizer can choose its visualization mechanism from Shine's community visualization library, which includes not only the physically-based model but also other mechanisms such as the dual-scaling method [11].

As described in Sec. 3.1, Shine adopts a flexible multi-agent architecture. When Community Organizer is ported to Shine, each client program is agentified. Such an agent exchanges personal information with other agents, and they all collaboratively perform mediation or self-introduction to find potential communities. The architecture of the Shine-based Community Organizer is shown in Fig. 7.

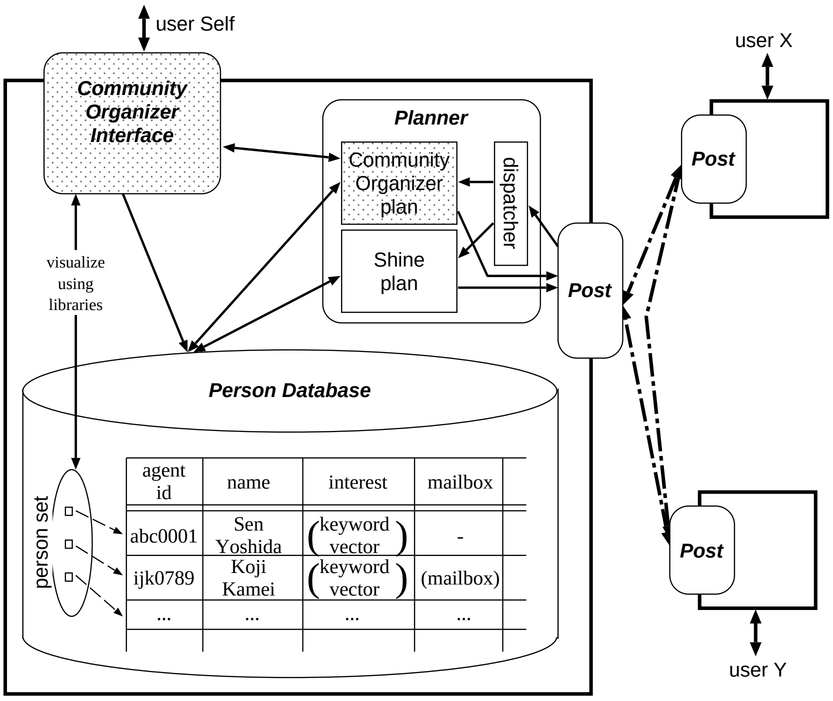


Fig. 7. Shine-based Community Organizer

In the current client/server version of Community Organizer, all data are stored in one database of the server. A client program sends the center point's keyword vector reflecting the slide bars' positions as a query. After receiving the query keyword vector, the server program searches its database for other keyword vectors close to the query's one, and then responds to the query with the found keyword vectors with corresponding icon images. Finally, the client program places these icons on the view panel.

In contrast, the Shine-based Community Organizer has no central database server. The queries are broadcasted to other personal agents. When an agent receives a query, it seeks icons whose keyword vectors are closer to the query's one from the icons of the user who owns the agent and then responds to the query with the found icons. Therefore, the computation of query processing is distributed.

5 Conclusions

In this paper, we explained both Community Organizer and Shineand then described how we built Community Organizer on top of Shine.

Much work remains to be done, however. We are going to study the architecture of Shine in more detail. We will also provide an implementation of Shine as a Java class library. We plan to do this work in parallel with the development of several network community support systems such as Community Organizer and Gleams of People [17]. There are also various other matters to deal with, including a privacy control mechanism.

Acknowledgments. We thank Advanced Business Works at NTT Communications for the implementation of Community Organizer.

References

[1] Matthew Chalmers and Paul Chitson. Bead: Explorations in information visualization. In Nicholas Belkin, Peter Ingwersen, and Annelise Mark Pejtersen, editors, *Proceedings of the Fifteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR Forum, pages 330–337. ACM Press, 1992.

[2] Tim Finin, Rich Fritzson, Don McKay, and Robin McEntire. KQML — a language and protocol for knowledge and information exchange. In Fuchi and Yokoi [3].

[3] Kazuhiro Fuchi and Toshio Yokoi, editors. *Knowledge Building and Knowledge Sharing.* Ohmsha and IOS Press, 1994.

[4] Kunihiko Fujita and Susumu Kunifuji. A realization of a reflection of personal information on distributed brainstorming environment. In Takashi Masuda, Yoshifumi Masunaga, and Michiharu Tsukamoto, editors, *Proceedings of the International Conference on Worldwide Computing and Its Applications '97*, volume 1274 of *Lecture Notes in Computer Science*, pages 166–181. Springer-Verlag, 1997.

[5] Fumio Hattori, Takeshi Ohguro, Makoto Yokoo, Shigeo Matsubara, and Sen Yoshida. Socialware: Multiagent systems for supporting network communities. *Communications of the ACM*, 42(3):55–61, 1999.

[6] John Hagel III and Arther Armstrong. The real value of ON-LINE communities. *Harvard Business Review*, 5–6 1996.

[7] Toru Ishida, editor. *Community Computing — Collaboration over Global Information Networks.* John Wiley & Sons, 1998.

[8] Koji Kamei, Eva Jettmar, Kunihiko Fujita, Sen Yoshida, and Kazuhiro Kuwabara. Community organizer: supporting the formation of network communities through spatial representation. Submitted to the 2001 Symposium on Applications and the Internet.

[9] Tetsuo Kinoshita and Kenji Suganuma. ADIPS framework for flexible distributed systems. In *Proceedings of the First Pacific Rim International Workshop on Multi-Agents (PRIMA '98)*, volume 1599 of *Lecture Notes in Artificial Intelligence*, pages 18–32. Springer-Verlag, 1998.

[10] Stefan Klein. Introduction to electronic auctions. *International Journal of Electronic Markets*, 7(4):3–6, 1997.

[11] Kenji Mase, Yasuyuki Sumi, and Kazushi Nishimoto. Informal conversation environment for collaborative concept formation. In Ishida [7], chapter 6, pages 165–205.

[12] Shigeo Matsubara, Takeshi Ohguro, and Fumio Hattori. CommunityBoard: Social meeting system able to visualize the structure of discussions. In *Proceedings of the Second International Conference on Knowledge-based Intelligent Electronic Systems (KES'98)*, pages 423–428. IEEE, 1998.

[13] Shigeo Matsubara, Takeshi Ohguro, and Fumio Hattori. CommunityBoard 2: Mediating between speakers and an audience in computer network discussions. In Oren Etzioni, Jörg P. Müller, and Jeffrey M. Bradshaw, editors, *Proceedings of the Third Annual Conference on Autonomous Agents*, pages 370–371. ACM Press, 1999.

[14] Uttam Mukhopadhyay, Larry Stephens, Michael Huhns, and Ronald Bonnell. An intelligent system for document retrieval in distributed office environments. *Journal of the American Society for Information Science*, 37:123–135, 1987.

[15] Yoshiyasu Nishibe, Ichiro Morihara, Fumio Hattori, Toshikazu Nishimura, Hirofumi Yamaki, Toru Ishida, Harumi Maeda, and Toyoaki Nishida. Mobile digital assistants for international conferences. In Ishida [7], chapter 8, pages 245–284.

[16] Toyoaki Nishida and Hideaki Takeda. Towards the knowledgeable community. In Fuchi and Yokoi [3].

[17] Takeshi Ohguro, Sen Yoshida, and Kazuhiro Kuwabara. Gleams of People: Monitoring the presence of people with multi-agent architecture. In Nideyuki Nakashima and Chengqi Zhang, editors, *Approaches to Intelligent Agents — Proceedings of the Second Pacific Rim International Workshop on Multi-Agents*, volume 1733 of *Lecture Notes in Artificial Intelligence*, pages 170–182. Springer-Verlag, 1999.

[18] Howard Rheingold. *The Virtual Community: Homesteading on the Electronic Frontier.* Addison-Wesley, 1993.

[19] G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval.* McGraw-Hill, 1983.

[20] Upendra Shardanand and Pattie Maes. Social information filtering: Algorithms for automating "word of mouth". In *CHI '95 Proceedings: Conference on Human Factors in Computing Systems: Mosaic of Creativity*, pages 210–217. ACM SIGCHI, 1995.

[21] Terry Winograd and Fernando Flores. *Understanding Computers and Cognition: A New Foundation for Design.* Ablex, 1986.

[22] Sen Yoshida, Koji Kamei, Makoto Yokoo, Takeshi Ohguro, Kaname Funakoshi, and Fumio Hattori. Community visualizing agent. In *Proceedings of the Third International Conference and Exhibition on The Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM 98)*, pages 643–644. The Practical Application Co. Ltd., 1998.

[23] Sen Yoshida, Takeshi Ohguro, Koji Kamei, Kaname Funakoshi, and Kazuhiro Kuwabara. Shine: a cyber-community application platform — a proposal —. In *Short Papers Proceedings of the Second Pacific Rim International Workshop on Multi-Agents*, pages 31–40, 1999.