Fumio Hattori, Takeshi Ohguro, Makoto Yokoo, Shigeo Matsubara, and Sen Yoshida

Socialware: Multiagent Systems for Supporting Network Communities

Many of the social issues that live in real neighborhoods also reside on the Net. Now, with a little help from software agents, people can be linked together to form a multitude of new colonies.

As the Internet continues to grow, social activities through the Net will become more and more common. Although several systems exist for supporting communities over the Net,[1] these systems have limitations that will pose problems in supporting forthcoming network communities. To resolve these problems, we are developing multiagent systems to assist in various social activities on network communities. We call these systems "socialware."

There seems to be three major issues in the support of network communities. The first is how to bring people together, that is, how to link people with others and with communities that share similar interests [1, 7, 8]. Search engines or directory services for communities can partially help, but more sophisticated systems are needed.

The second issue revolves around support for smooth communications [4], including support for visualizing and sharing common contexts,[2] as well as

[1] CommunityWare (www.communityware.com) or Firefly (www.firefly.net) can be viewed as typical examples of community providers.
[2] The term "context" here includes such aspects as the depth of knowledge of individuals, their attitudes toward the subjects, and mutual knowledge and manners (ethics) for the community.

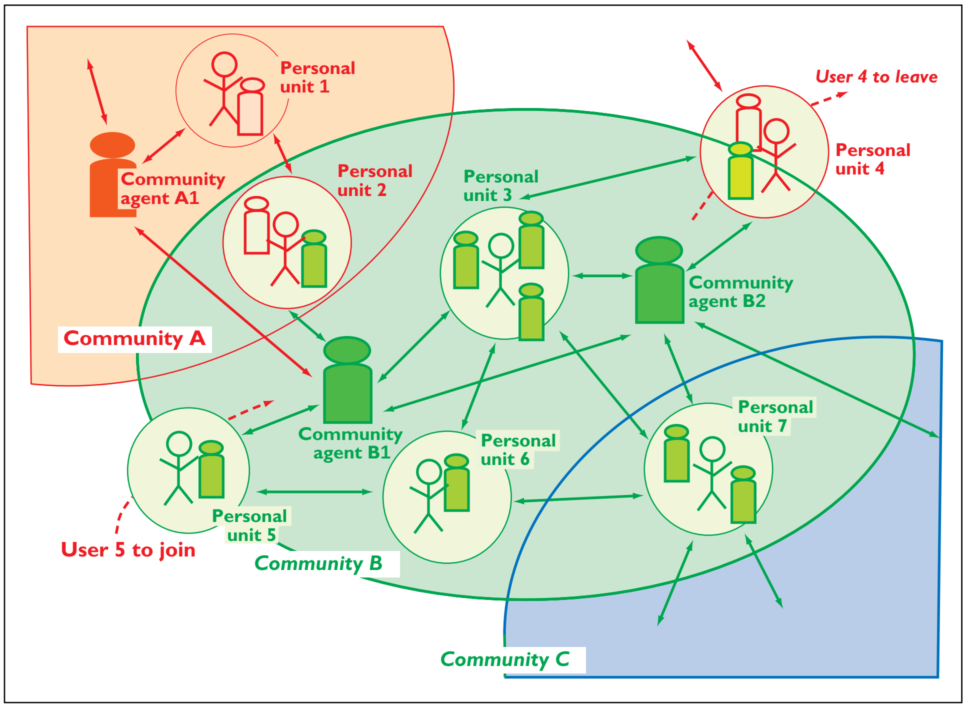


Figure 1. A general architecture of socialware as a multiagent system

for identifying the flow of conversations/discussions [5]. A lack of context or a failure in identifying the flow of a discussion may cause, for example, *flame wars,* which, unfortunately, often happen in network communications. Another example is where a newcomer may have difficulty understanding what is going on in a community.

The third issue is finding the relationships between people [3], including how to identify the objectives/roles of communities and individuals. Since network communities are amorphous compared to organizations like companies and schools, such relationships and networks are sometimes hidden or unknown. This causes difficulty in finding the key person or people suitable for the current concerns.

These issues also exist in real communities, but the problems are more obvious in network communities since they are not constrained by restrictions of time and place. Providing a new communication environment alone is not sufficient for resolving these problems [6]. Active support using agent technology based on interaction between software agents, and between humans and software agents, seems important and necessary. We believe that socialware is a promising area for multiagent applications.

Socialware as Multiagent Systems

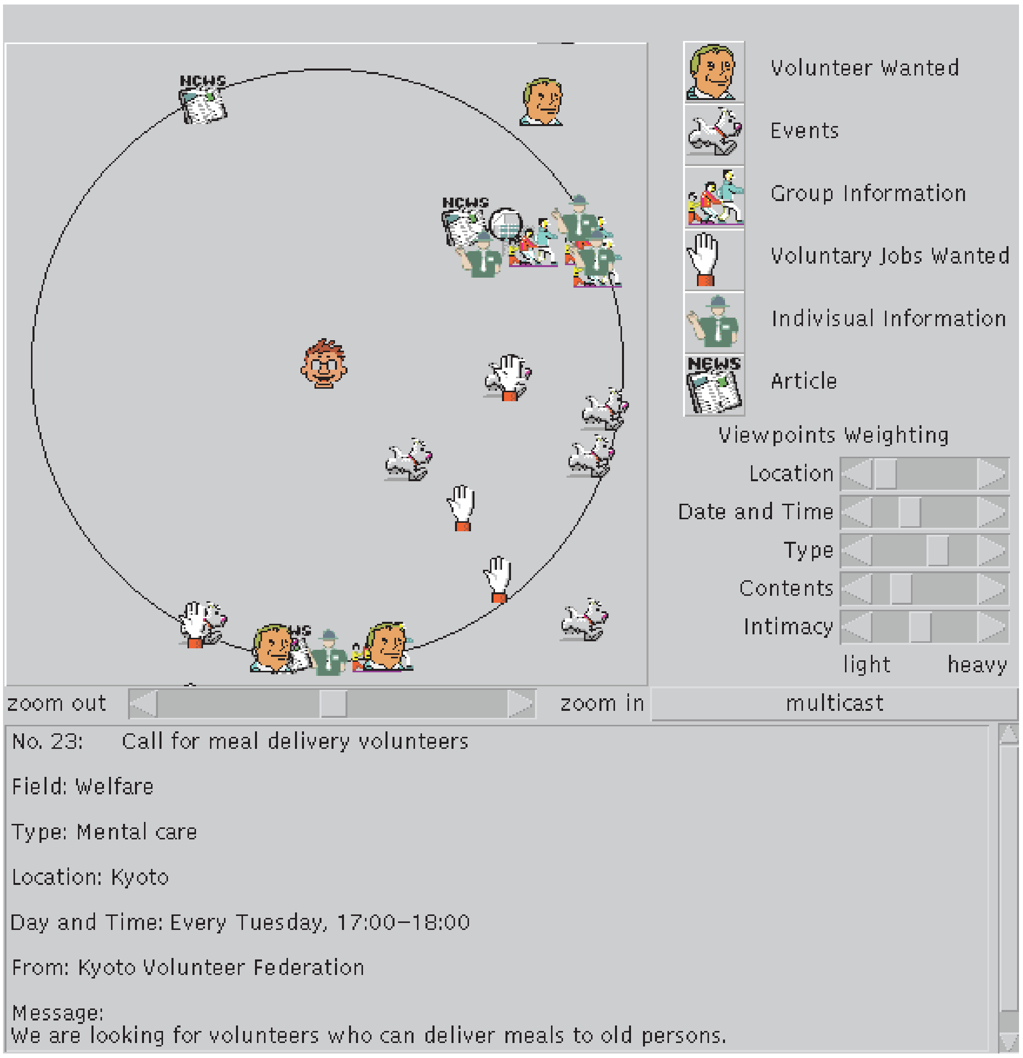
There are several characteristics specific to network communities that make a multiagent architecture attractive to use. First, the participants of a network community are widely distributed, and the number of potential participants is large. Hence, no solid, centralized, or monolithic system would be adequate. In fact, a distributed system where personalized agents for each participant cooperate with each other would be required.

Another characteristic is that communities have a dynamic nature. In each community, the active membership will change over time, in addition to the roles of individuals and objectives, and moreover, the community will likely change its aspect. In other words, there exists no fixed organization nor a clear goal for a network community. This characteristic contrasts the area of groupware, which helps people already organized work cooperatively, where the members, their roles, and their objectives are rather clearly defined.

In addition, the individuality of each member is preserved. That is, each member can have diverse objectives, even if all members share common interests in general. Furthermore, people can be members of several communities at the same time, depending on their various interests. Hence, support needs to be personalized to adapt individual objectives and interests. They also need to adapt to the variations and changes of interests and activities of individuals.

We propose a general architecture of socialware (Figure 1) based on these observations.

We view a community as a collection of personal units, community agent(s), and the set of relations between them. A personal unit consists of a user and his or her personal agents. Each personal agent can



No. 23: Call for meal delivery volunteers

Field: Welfare

Type: Mental care

Location: Kyoto

Day and Time: Every Tuesday, 17:00–18:00

From: Kyoto Volunteer Federation

Message:
We are looking for volunteers who can deliver meals to old persons.

Figure 2. A screen shot of the CommunityOrganizer through a personal agent

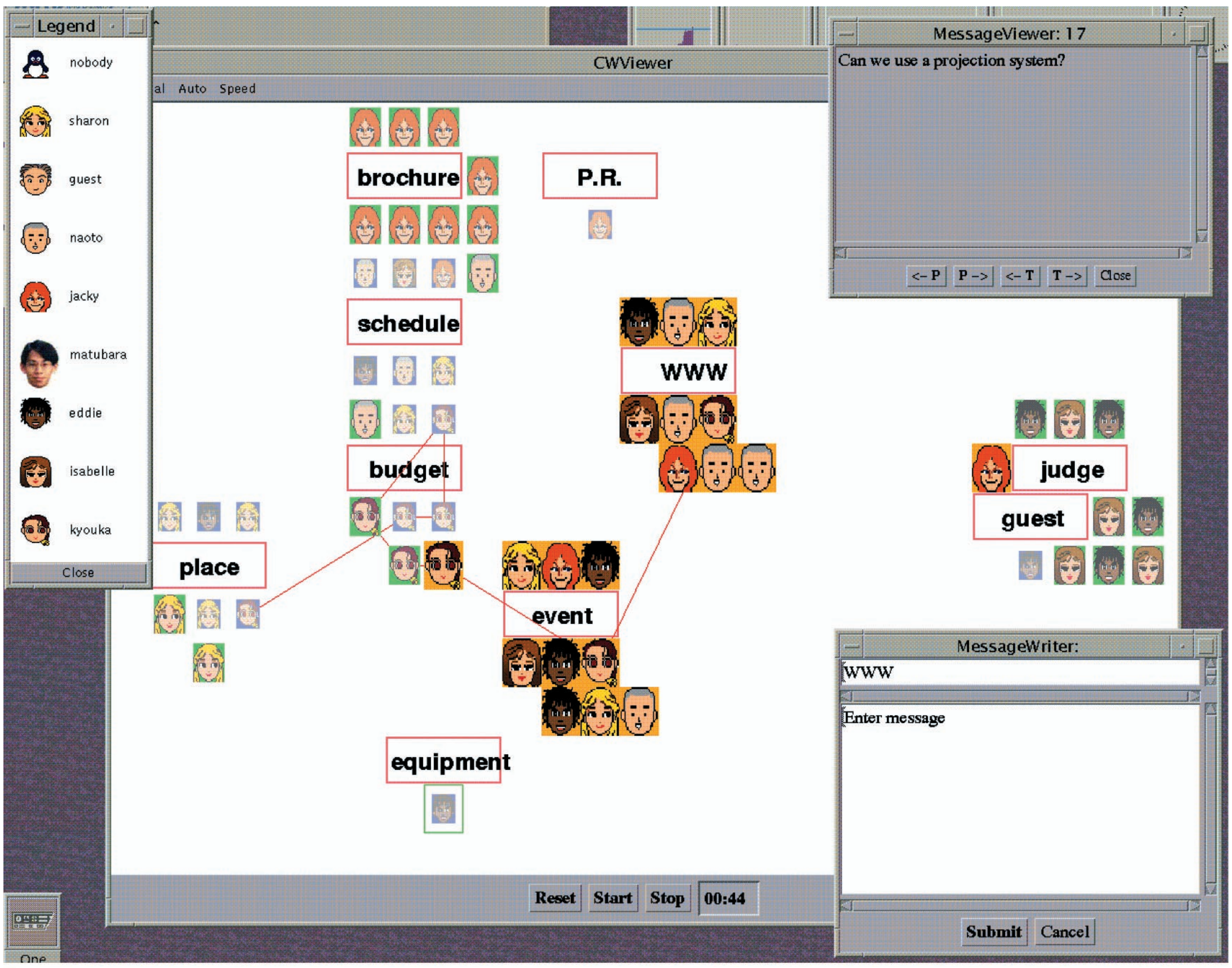
help the user by gathering and exchanging information, visualizing contexts, and recommending or assisting the user in making a choice. All of this is accomplished in personalized ways. All of the agents cooperate and act as a unit, with the user being the central figure. The community agents have the function of providing shared information, knowledge, or contexts within the community and act as mediators for informal communications between people.

Having an architecture where each user has personal agents that communicate with each other enables the community to spread over the Net. In a personal unit, it is possible for some agents to be domain specific (for example, an information retrieval agent specialized for financial news), and others more generic (for example, an interface agent for navigating and reading documents). Adaptation to multiple aspects of a user and the user's changes in interests can be achieved by changing the system dynamically and autonomously. For example, a domain-specific agent can clone itself to produce a new agent and make additional communication channels when the user's interest has changed. This architecture also maintains the user's individuality. When the user belongs to several communities simultaneously, he or she may have multiple domain-specific agents for each community.

The whole system needs to be flexible and adaptive so it can be tailored to the dynamic nature of a community where the community members, as well as their relationships, will likely change over time.

Experimental Systems of Socialware: CommunityOrganizer

We have developed a prototype application for the purpose of linking people, which we call the "CommunityOrganizer" [8]. The system consists of personal agents for each user and a community agent. Each personal agent has functions to acquire the user profile and to visualize potential communities around the user. The community agent has func-



Can we use a projection system?

www

Enter message

Figure 3. A screen shot of the viewspace for the CommunityBoard

tions to collect the user profiles and to maintain the information on potential communities.

Upon a request from a personal agent, the community agent first computes potential communities around the owner of the personal agent, and then sends the necessary data (users in the potential communities and relevances among them) to the personal agent. Next, the personal agent locates users (indicated by icons) in a 2D space, where the distances among the users reflect the relevances (degrees of common interest) among them. In any viewspace, the owner of the personal agent is located at the center.

The relevances between users are calculated by the community agent from the users' profile data. These profiles can be obtained from each user's input, from archives of mailing lists using keyword extraction techniques, or from user information on the Web. (The system acquires profile data automatically when the URL for a user's home page is given.)

Each personal agent has slidebars which temporarily adjust the weightings of the viewpoints since the degree of common interest consists of multiple aspects. When the domain is voluntary actitivities, there are such aspects as location, date and time, type, content, and intimacy (Figure 2). When the user changes the weightings of the viewpoints using the slidebars, the personal agent automatically recalculates the appropriate locations of the other users using a weighted cosine measure and redisplays them.

For further personalization of the view according to the user's interests, the personal agent can learn the weightings from user feedback. When the user finds that a person is located at an inappropriate position (that is, too close to or too far from the user), the user can indicate this fact to his or her personal agent. Then the agent can modify the weights of keywords permanently to adjust the relevance value between the indicated person and the user.

When people or potential communities with similar interests are found, the user can multicast a message to them. The multicast area is basically within a certain distance from the user, but can also be adjusted by the user. Furthermore, users can exchange their viewpoints (weightings) through the community agent. If a viewpoint received is attractive, a user can incorporate it into his or her own weightings. This function is useful in forming a new community, since the exchanged viewpoints can make the individuals' interests more clear, and can lead to common interests shared among the users.

In this way, the system enables users to find a variety of potential communities around them, to simulate several possibilities of potential communities, and to form new communities.

CommunityBoard

We have also developed a prototype application for visualizing the structures of discussions called the "CommunityBoard" [5]. The system consists of personal agents for each user and a community agent, all of which are written in Java. Each personal agent displays the structures of discussions according to the user's interests (Figure 3). The community agent classifies messages according to several criteria, like topic, time, and reputation, and provides a shared information base in the community.

Three main dimensions are chosen to express the structure of a discussion: person, topic, and importance. In an integrated viewspace of the personal agent, each message is represented as an icon. Person, topic, and importance are represented by the type, position, and shading of the icon, respectively.

Person. Each person has his or her own icon for messages.

Topic. Each message is located according to its topic, with similar topics located nearby. The topic for a message is currently given by the writer of the message as a keyword (we plan to allow multiple keywords or to extract keywords automatically from messages). The relevances between topics are calculated using a database on the community agent. Each personal agent decides the locations of the topics from the relevances using a multidimensional scaling method. The locations are recalculated whenever a message with a new topic arrives.

Importance. Icons for messages become dimmer and smaller as the importance decreases. The importance is a function of time, reputation, and interest for topics of messages. Interests for topics as well as the importance function itself can be adjusted by the user. Reputations of messages are calculated by the community agent using collaborative filtering techniques.

Users can review the various combinations of these three aspects at a glance, without changing the viewspace itself. Such information can be used in several ways. A user can decide whether to join a discussion or can guess to whom a question relating to a topic should be asked, from the information about persons and topics. Users can also grasp the relations between topics, what topics have been discussed (or not discussed), and what topics are currently important, from the information about topics and importance. Such knowledge helps users learn the current context of the discussion and helps them anticipate responses beforehand, like before they join the discussion. Furthermore, the users can better understand the transitions between topics from the information about persons and importance. This helps them understand other users and the causal relations between their messages.

In this way, users can obtain and share the contexts of a community. This helps smooth communications between people, helps in revealing relations between people, and helps in finding the role of each person.

An architecture where each user has personal agents that communicate with each other enables the community to spread over the Net.

Future Direction

Socialware is a new concept for network community support [2]. We believe it opens up new areas of application for a multiagent framework. These areas include:

- Modeling the behaviors of communities;
- Structuring communities;
- Enabling flexible communication channels;
- Learning and collaboration of users and agents; and
- Mediating public information spaces and personal spaces, including balancing between privacy and open opportunity.

There are still several technical challenges in implementing socialware as a multiagent system, namely how to make a coalition of agents, and how humans and agents can effectively collaborate as a system. C

References

1. Foner, L.N. A multi-agent referral system for matchmaking. In *Proceedings of PAAM'96* (Apr. 22–24, London). The Practical Application Company, Lancashire, 1996, 245–261.
2. Ishida, T. Towards CommunityWare. In *Proceedings of PAAM'97* (Apr. 21–23, London). The Practical Application Company, Lancashire, 1997, 7–21.
3. Kautz, H., Selman, B. and Shah, M. Referral Web: Combining social networks and collaborative filtering. *Commun. ACM 40,* 3 (Mar. 1997), 63–65.
4. LaLiberte, D. and Woolley, D. Presentation features of text-based conferencing systems on the WWW. *Computer-Mediated Commun. 4*, 5 (May 1997).
5. Matsubara, S., Ohguro, T. and Hattori, F. CommunityBoard: Social meeting system able to visualize the structure of discussions.

Simulations in Economics and Management

Alok R. Chaturvedi and Shailendra R. Mehta

Economics addresses this basic problem: There are *n* agents interacting with each other in *m* markets. We want to find out the equilibrium set of actions of each of the agents in each of the markets as well as the associated prices and quantities. As agents and markets increase in number, the computing power required to solve this problem increases exponentially in *nm*, and inversely with the level of precision one wishes to achieve.

It is clear, as Rust [1] eloquently argues, that for an even moderately complex economic model, there is very little hope of actually solving the problem. This has, in fact, spawned more and more literature on *impossibility theorems* (also surveyed by Rust) pertaining to the computation of economic problems. One solution to this conundrum is to have agent-based models. Here, the power and intelligence of real-life markets is mimicked in the laboratory populated by agents acting in markets exactly as they are supposed to in real life. Rather than trying to compute the solution to models, we actually let the market mechanism find one for us. Again, as Rust points out, the computing power required now only increases exponentially in the number of markets, *m*. This route provides an acceptable theoretical alternative. This is the motivation behind the Synthetic Economy for Analysis and Simulation (SEAS) project.

SEAS is a distributed, interactive, real-time environment for conducting large-scale experiments and simulations in areas where interactions among agents need to be studied. SEAS replicates the real world in most crucial dimensions, such as competition, regulation, decision variables, and interaction dynamics. It consists of interlinked goods, stock, bond, labor, and currency markets, as the Figure in this sidebar shows. In these markets, two types of agents interact: *live*, that is, people acting as buyers, sellers, regulators, intermediaries; and *virtual*, or artificially intelligent software agents, that mimic human consumers in a narrow domain.

SEAS is a dynamically reconfigurable environment structured around the interplay of human decisions and game events that requires active involvement of participants. It helps participants understand the sources and motivations underlying the decisions by placing them in the shoes of executives running the firms in different points in time and challenging them to do better. Games dealing with current or future situations help explore the potential implications of various courses of action and raise important questions for further investigation.

A game design requires four steps: setting the geography of the game; designing the game board, pieces and scoring; customizing the database; and calibrating the artificial agents' parameters to match that of the real consumers. SEAS simulation process consists of three steps:

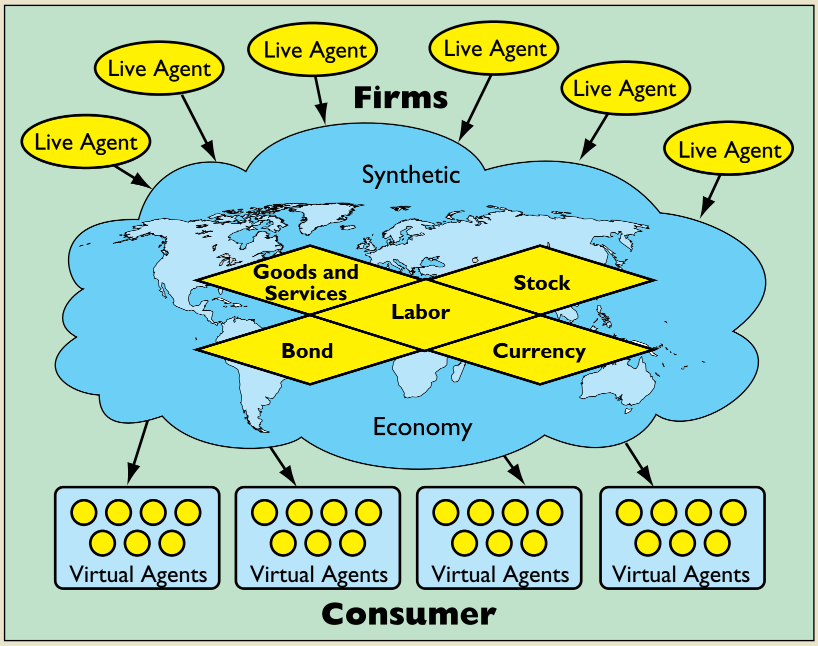
- *Pre-game briefing*. It provides an industry overview of the players and their strategic positions and the rules of engagement.

In *Proceedings of Knowledge-based Intelligent Electronic Systems (KES'98)* (Apr. 21–23, Adelaide, Australia). IEEE, Piscataway, N.J., 423–428.
6. Nakanishi, H., Yoshida, C., Nishimura, T. and Ishida., T. Free-Walk: Supporting casual meetings in a network. In *Proceedings of CSCW'96,* (Nov. 16–20, Boston.). ACM, N.Y., 1996, 308–314.
7. Nishimura, T., Yamaki, H., Komura, T., Itoh, N., Gotoh, T. and Ishida, T. Community Viewer: Visualizing community formation on personal digital assistants. In *Proceedings of IJCAI'97 Workshop on Social Interaction and Communityware* (Aug. 23–29, Nagoya, Japan). Morgan-Kaufmann, San Francisco, 1997, 25–30.
8. Yoshida, S., Kamei, K., Yokoo, M. Ohguro, T., Funakoshi, K. and Hattori, F. Community visualizing agent. In *Proceedings of PAAM'98* (Mar. 23–25, London). The Practical Application Company, Lancashire, 1998, 643–644.

Fumio Hattori (hattori@cslab.kecl.ntt.co.jp) is the executive manager of the Computer Science Laboratory at NTT Communication Science Laboratories, Kyoto, Japan.
Takeshi Ohguro (ohguro@cslab.kecl.ntt.co.jp) is a research scientist at NTT Communication Science Laboratories, Kyoto, Japan.
Makoto Yokoo (yokoo@cslab.kecl.ntt.co.jp) is a research scientist at NTT Communication Science Laboratories, Kyoto, Japan.
Shigeo Matsubara (matsubara@cslab.kecl.ntt.co.jp) is a research scientist at NTT Communication Science Laboratories, Kyoto, Japan.
Sen Yoshida (yoshia@cslab.kecl.ntt.co.jp) is a researcher at NTT Communication Science Laboratories, Kyoto, Japan.

- *Game playing*. Here, the participants try out their long-term strategic and short-term tactical moves, evaluate their performance periodically, and make adjustments.
- *After-action review*. Participants are allowed to develop strategic insight by reviewing the performance of each of the groups, analyzing the moves, countermoves and their effectiveness, and learning from collective experiences.

Several simulations and experiments have been conducted using SEAS. For details, see [2].



SEAS consists of a fully functioning synthetic economy with interlinked goods and services, bond, labor, and currency markets that closely replicates essential aspects of internal and external business environments facing a firm. Teams of human participants represent the producer side of the economy, while the demand side can be represented in desired detail by arbitrarily large numbers of artificial agents.

References

1. Rust, J. Dealing with the complexity of economic calculations. Workshop on *Fundamental Limits to Knowledge in Economics.* (Santa Fe Institute, July 31–Aug. 3, 1996).
2. Chaturvedi, A. R., and Mehta, S. R. Synthetic Environments for Analysis and Simulations (SEAS): A Technical Report. Technical Paper, Purdue University, Aug. 1998

Alok R. Chaturvedi (alok@mgmt.purdue.edu) is an associate professor of information systems in the Krannert School of Management at Purdue University, West Lafayette, Ind.
Shailendra R. Mehta (mehta@mgmt.purdue.edu) is Director, Entrepreneurship Initiative in the Krannert School of Management at Purdue University, West Lafayette, Ind.