An Approach to Developing an Agent Space to Support Users' Activities

Kenji Sugawara
Faculty of Computer and Information Network Science
Chiba Institute of Technology
Narashino, Japan
suga@net.it-chiba.ac.jp

Jean-Paul A. Barthès
UMR CNRS Heudiasyc,
Université de Technologie de Compiègne
Compiègne, France
barthes@utc.fr

***Abstract*— Information and communication technology and ubiquitous technology allowed developing smart services for supporting users in their daily life. However, users experience great difficulties for finding rapidly the information they need when they need it, essentially because the current information systems seems to be based on a system-centric approach. In this paper, we discuss a new approach centered on users' requests, using an Agent Space that contains Personal Agents, Social Agents and a Mediation Agent. We distinguish between Real Space, the space in which the user lives and Digital Space, the space containing social information. A personal agent interacts with a user to recognize the user's activity in real life and a social agent interacts with objects in Real Space and with social information data in Digital Space. A Mediation Agent saves social information according to its role and distributes it to all interested agents. The agents are designed to run on the OMAS platform, a distributed multi-agent platform.**

Keyword-multi-agent system; personal assistant; agent platform; agent space.

I. Introduction

The growth of Information and Communication Technology enables us to receive various kinds of information from the Internet using high performance personal computers and recent advanced network technology. Many people can easily send (store) and receive (use) information in a Digital Space (DS). After a while however, the DS ends up containing a large amount of data and information.

Furthermore, the rapid development of ubiquitous technology also enables information systems to access Real Space (RS) through mobile communication devices like smart phones. Various kinds of sensor devices can also be deployed anywhere in the RS and large amounts of multimodal information about users and their surroundings can be acquired and saved in the DS. Using such information, very convenient services called "anytime" or "anywhere" can be provided for users, daily and widely [1].

However, users have a difficult time finding the information and help they need when they need it, because it may be hidden in an enormous amount of useless information in the DS [2]. When a user has to look for some event or for somebody in the RS through the information the DS provides, the user may have to operate different applications and interfaces. Unskilled users may loose important opportunities they could exploit if they had the skill to work with complex information systems.

The agent-based technology is one of the possible solutions for distributing information from the DS to an unskilled user in the RS [8]. An Agent Space (AS), built as a well-designed collection of agents, can be a platform to meet users' requests in the RS using proper social information from the DS [6]. The users' requests change continuously following their activities in the society. Similarly, the contents and organization of the social information in the DS change according to the changes in the society in the RS. Thus, meeting users' requests with social information can be very difficult for conventional static algorithms when they try to solve the problem autonomously. This is a reason why users need skills to handle the computer and the network efficiently. Moreover, skilled users waste their time watching web sites and data in the DS.

Because current information systems have been designed using a system-centric approach generally so far, users have problems finding information rapidly. Our approach on the other hand is user-centric and we design an Agent Space that is deployed between the RS and the DS in order for agents to match users' requests with proper social information.

In this paper, we aim at developing an agent space that consists of Personal Agents, Social Agents and Mediation Agents. An agent in this paper is designed as a complex cognitive multi-threaded system that has a function of recognizing, choosing goals and intending to use them, having knowledge, performing actions and using memory. Such functions are provided for agent developers by the OMAS platform and we extend the platform by connecting it to sensors and devices in the RS that behave as if they were OMAS agents [3]. Finally, we discuss an example of an Agent Space supporting work activities of coworkers at an office by watching them through sensor agents.

II. An Agent Space Interacting With Real Space and Digital Space

A collection of agents is called an Agent Space (AS) in which all agents receive events and perform actions from/to objects in RS/DS. We assume in this paper that all agents run on distributed multi-agent platform connecting them over the Internet and LANs and includes knowledge base functions provided by the associated MOSS environment [4], and interface functions that connect agents to RS/DS.

An AS, shown in Fig. 1, consists of Personal Assistant Agents (PA), Social Agents (SA) and a Mediation Agent (MA). A PA interacts with a particular user to support the user answering requests and following the user in RS. An SA interacts with DS to acquire social information represented by data in the DS. The MA selects and saves a particular domain of social information according to its given role and it distributes it to interested agents of the AS.

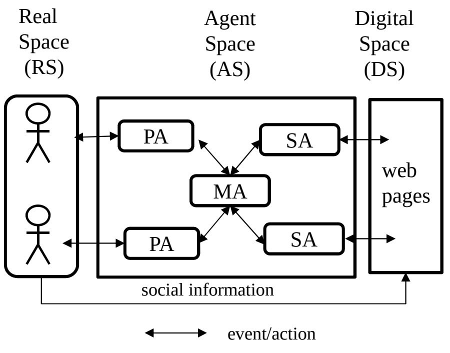

Figure 1. Agent Space Interacting with RS and DS

Until now, many research projects, such as Ambient Intelligence or Smart House, have led to and promoted ubiquitous technology using sensors and actuators like cameras, microphones, or speakers deployed in the RS. Furthermore, many research works like Web Service Computing, Semantic Web, Web Mining, have promoted the Web Service technology in DS, to make the DS data machine readable using knowledge-intensive frameworks.

The technology is bringing in functions to recognize user's actions, to help user's actions automatically, to mine social information from data in Web pages, such as the service of Social Network, and to post news automatically to interested users of a domain. The functions are incorporated in particular application to provide each particular domain a smart service for users. In this case, users must have the skills to select and utilize such services depending on their situation and what they want to do.

In this paper, we discuss the integration of the function of ubiquitous technology and web service technology through an AS from a user-centric point of view: We examine the following points:

(1) modeling a Basic Agent to define a common agent architecture in AS,

(2) agentifying programs of ubiquitous and web service functions for AS to interact with RS and DS,

(3) modeling PA, SA and MA by modifying the Basic Agent model to give each of them a role and appropriate knowledge.

III. A MODEL OF A BASIC AGENT

A. A Model of a Basic Agent

An agent space, shown Fig. 1, is a multi-agent system consisting of Basic Agents (BAG) and Event/Action Agents (EAAG) as shown in Fig. 2. A BAG is a component of PA, SA and MA implementing a logical process and using knowledge models. An EAAG is a component agent of PA, SA and MA dealing with interactions with RS and DS according to its role . A BAG receives events from agents and sends actions to agents as shown in Fig.2. An event or action is described by a message between agents that are defined in the OMAS framework. The content of an event/action is defined by an Agent Space Ontology letting agents communicate and cooperate with one another.

An EAAG receives events from agents and sends actions to agents. It also receives events from sensors in RS and descriptions from DS. It sends signals to control actuators and descriptions to write data into DS. EAAGs transform signals and descriptions into an event that is defined in the Agent Space Ontology. The EAAGs also transform actions from other agents into signals to control actuators in RS and descriptions to change data in DS.

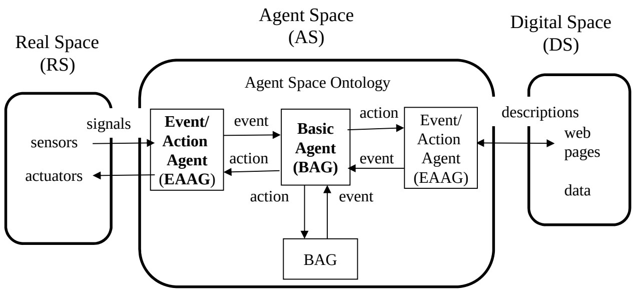

Figure 2. Basic Agent and Event/Action Agent

As shown in Fig. 2, a BAG has three kinds of knowledge named Memory, Knowledge and Goal in order to make an action to the RS and the DS using devices according to events acquired through sensors deployed in the RS/DS. The BAG also makes actions of sending messages to other BAGs and receives messages as events from other BAGs. Memory, Knowledge and Goal are all modeled by the Property Driven Model that is a frame-based knowledge model [4], and saved into knowledge bases that are managed by the MOSS knowledge base management system [5] provided in the OMAS agent platform.

```
agent BAG (event)
data:      Memory, Knowledge, Goal
  Memory <- Cognition( event,);
  action    <- Decision( Memory);
 Memory  <- Intention( action, Memory);
do action
```

Figure 3. Procedural Descriptopn of a Basic Agent

A structure for an agent was proposed by Russel and Norvig [10]. Based on the structure of agents, Fig. 3 shows a BAG procedure for selecting an action. It consists of three functions Cognition, Decision and Intention. A function of Cognition updates a Memory following the reception of an event from the RS/DS. Memory is a knowledge base that represents the current state of the world of an agent and records the history of actions the agent has done. We specify a basic model of the world for an agent, common to all agents when they begin to run on the OMAS platform. The knowledge of the environment and of the society of a user is represented as a set of class objects, and instance objects generated from the knowledge and saved in a Memory in each agent. The event in Fig. 3 is described as an instance generated from the knowledge of environment and society.

The Decision function in Fig. 3 returns an action of the BAG for controlling a device, referring to Memory, Goal, Actions. The action is defined in the knowledge base named Action and can be executed by a device of Fig. 2. The Intension function in Fig. 3 updates a Memory that records a history of actions and a prediction that the BAG intends to change a state in the RS before it received an event.

In this paper, a model describing the knowledge of the RS and DS is represented by a frame-oriented knowledge representation model Property Driven Model [4]. The function of transforming signals from sensors in the RS into an event described by the model of the RS and the DS, that defines descriptions of a knowledge base named Memory in Fig. 3, is assumed to be processed in the sensor of Fig. 2 in the RS. The data in the DS are also transformed into the descriptions defined in the model M. The Memory in Fig. 3 represents a current state of the RS and DS as a memory of a BAG.

B. *Agentification of Sensors and Devices*

The concept of agentification was discussed by Shoham as a component of the Agent-Oriented Programming framework that refers to bridging the gap between the low-level machine process and the intension level of agent programs [11]. In this paper, the concept of agentification is used to connect the AS, the RS and the DS. Agentification is a method to connect a program, a sensor and devices into an agent that communicates with other agents in the AS using an Agent Communication Language as shown in Fig. 5. When a system that runs in an executing platform autonomously becomes operational for a BAG thanks to a function, then we say that the function agentifies the system. An agentified system communicates with other BAGs as a BAG with Agent Communication Language although the inner architecture of the agentified system is different from a BAG.

For example, in Fig. 4, we suppose that a system consisting of a recognition program and a sensor keeps recognizing the position and poses of a user in the RS. The recognition program receives signals from a sensor and transforms them into a perception of poses and positions that is defined by a representation of a knowledge model of a BAG in Fig. 4. An agentification program creates a message consisting of items including a description that represents an event perceived in the AS recognized by a recognition program and a sensor.

On the other hand, an action from a BAG is sent to an agentified system to control a device giving an effect to an object in the AS. A message from an agent is transformed into a command for a control program of the device. A program to transform a result of a recognition program into a message and to transform a message into a command to a control program is called a base program of EAAG.

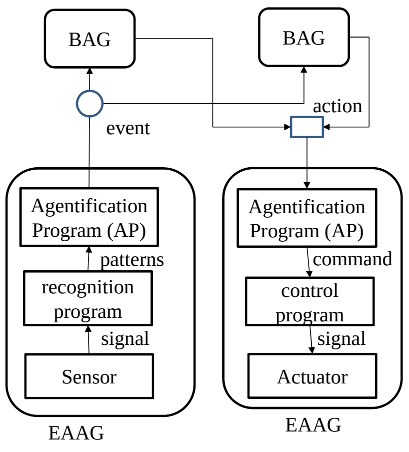

Figure 4. Structure of Event/Action Agent

A basic function in an EAAG has an inner state and decision function logically the same as for the BAG defined in Fig. 3. It receives a message from a BAG and a perception from a recognition program. It sends a message to a BAG and a command to a control program, based on functions of Cognition, Decision and Intention referring to knowledge bases in Fig. 3. However, the basic functions of an EEAG are currently implemented by a simple C++ program.

An advantage of the agentification is to be able to share a number of systems consisting of sensors and devices in the RA by agents in the AS. Agentification is an effective method for implementing a user-centric information system as shown in Fig. 1.

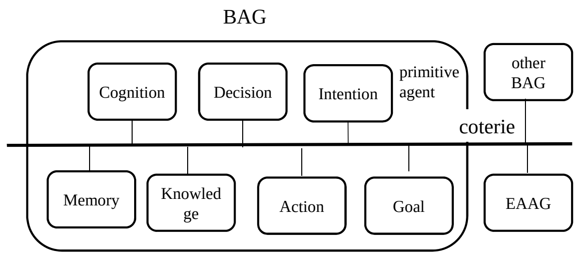

Figure 5. Multi-agent-based Architecture of a Basic Agent

Fig. 5 shows the internal architecture of a BAG as defined in Fig. 3. A BAG is a multi-agent system consisting of primitive agents that are programmed using an OMAS agent programming language. The OMAS agent platform allows to implement many agents that work in a runtime environment as a multi-agent system. Every function and every knowledge base is implemented as a BAG or a system of BAGs. A space where BAGs work using broadcast communication is called a coterie. Other BAGs and EAAGs can join a coterie to cooperate.

Fig. 6 shows the architecture of a primitive generated on an OMAS agent platform. A primitive is a multi-threaded system. It has an internal knowledge base management system to manage several knowledge bases, because the models introduced previously like Memory or Knowledge will be huge and will have to be divided into several sub-models. A scan process deals with an input message (event) to analyze the message as shown in Fig. 6. If it is a message asking for information or requesting to execute a task, then the primitive agent searches its knowledge base generating dynamically a task process (a new thread) then eventually sending a message to another primitive agent in the BAG as a subtask.

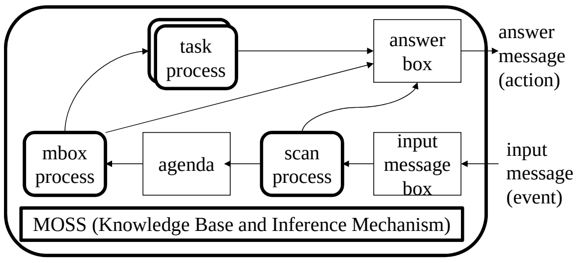

Figure 6. Structure of a Primitive Agent of OMAS platform

IV. DESIGN OF AGENT SPACE

Fig. 7 shows the structure of an AS consisting of three types of BAGs that are PA, SA and MA. Users live in the RS and work supported by PAs that find information in the DS. In this paper, we assumed that PAs and SAs in the AS can interact with users and objects through EAAGs that control sensors and devices deployed in the RS/DS.

A PA is a BAG that interacts with a single user, called its master, in order to support her persistently. A PA also interacts with objects and events through sensors and actuators deployed in the RS. A PA has a user model (UM) of its master described in a knowledge base for interacting with its master tightly and continuously.

A series of signals from a sensor is transformed into a set of descriptions modeled by the Memory component of a BAG. It is a list of symbols defined by a protocol between the devices and the PA. An action is a list of symbols defined by a set of descriptions modeled in the Memory component of the BAG. It is specified by a protocol between the output devices and the PA. The action is transformed into a series signals for controlling the devices [9]. A PA has ontologies that represent a level of cognition. A set of descriptions modeled by the Memory component is a list of symbols defined by the higher level of the ontology. It

represents the cognition of the PA and is shared by other PAs and BAGs. Messages may include the set of descriptions modeled by the Memory component in their content, and are exchanged among agents that are related.

The goal of a PA is to support its master to find the user’s requests by conversing with her and following her through the sensors. When the PA recognizes a request from the user, it sends a message to another agent asking it to process the request. Agents process tasks cooperatively with other agents in the AS. They can be pre-installed or added dynamically according to provider needs.

A Social Agent SA is a BAG meant to search and save information concerning a particular social domain in the RS/DS. It provides it to other BAGs by accessing resources in the DS, using their communication protocols [7]. SAs also search and save information concerning a particular social domain in the RS by accessing sensors embedded in the RS. SAs acquire social information from sensors deployed in the RS directly like PAs.

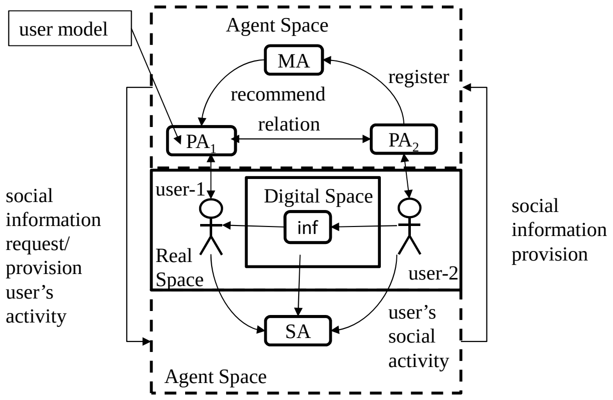

Figure 7. Structure of Agent Space

The agent space AS is a collection of agents resulting of a bottom up approach like web sites and new users. Users of the AS create their own PAs and SAs to their convenience and according to their interests. Some users may want to use the AS as a place to deliver advertising information. A Mediation Agent is a BAG for finding relations between BAGs in the AS and providing information about such relations.

When an agent is generated in the AS, it has an initial set of relations to some MAs for sharing information. It must acquire knowledge by watching actions and exchanging information. Knowledge acquisition by combining users' actions and shared knowledge is a growing and active area of study [12].

V. DESIGN OF PERSONAL AGENT AND SOCIAL AGENT TO SUPPORT SOCIAL ACTIVITIES OF A USER

A. *Design of Personal Agent*

A PA is designed for a particular user who owns it and installed in the AS that interacts with the RS and the DS. The PA supports its master according to the user's requests based on a user model. It cooperates with other PAs, TAs and SAs as shown in Fig. 7. If the user model is not adapted to the agent's goal, the user cannot receive better services from the DS. Actually, a PA for a user is designed with goals, knowledge and properties different from other PAs because each user has different goals, knowledge and properties.

Therefore, a PA should have the ability to acquire a user model and preferences after it is installed and runs in the AS. Acquiring a user model is still a difficult problem in artificial intelligence. However, it is very important for the user-centric program to be able to acquire user's properties to make a user-centric service.

In order to develop a PA dedicating its actions to supporting its master, we have to design the following BAG functions adding to the conventional fundamental functions of agents:

(1) several built-in knowledge bases of a general owner user model and a general society model adding to Memory, Knowledge and Goal,

(2) set-up function of the models of an owner user when it is installed in the AS,

(3) function to update the model of the owner user regarding actions in the RS and requests to it,

(4) function to update the social model regarding actions of the owner user with other users in the RS.

Therefore, we extended the BAG decision procedure defined Fig. 3 to define a new procedure for a PA as shown in Fig. 8. A PA incorporates two knowledge bases: UserModel (UM) and a SocialModel (SM) adding to the knowledge bases shown Fig. 8 to answer the item (1) described above. A PA also incorporates two functions: UpdateUM and UpdateSM in the procedure of a BAG as shown in Fig. 8 to answer item (3) and (4). A variable named userModelFlag is introduced to invoke a conversation graph to initialize the Memory, UserModel and SocialModel through a dialog with the master to answer item (2). While the userModelFlag is ON, the PA continues the dialog with the user to add new information to the knowledge bases.

B. *Design of Social Agent*

An SA keeps watching social relations between users in the RS using event in the RS and the DS sent by the agentification program (EAAG) as shown in Fig. 4. An SA shares messages concerning events from the RS with PAs in order to keep watching over dedicated user domains, like a family, an office, a section of a factory, or a special interest group.

The SA incorporates a SocialModel shared by PAs having interests in a domain they are designed to support. The SocialModel is a knowledge base representing relations between users belonging to the domain, social activations of each user and domain knowledge to describe special configurations, roles in a society and so on. The UpdateSMl is a knowledge-based procedure in particular a definition of the domain, maintenance of the SocialModel, interrelations between domains.

When the SA recognizes a relation between users, the relation is saved in the SocialModel by a pattern of events sent by sensors. Therefore, in the process of recognizing a social relation, the SA begins to build a hypothetical relation and waits for the following events from the AS to verify the hypothesis as shown in Fig. 9. The hypothesis is saved and managed in a knowledge base of SocialHypotheses and is changed and verified by a function Watch shown Fig. 9. After the hypothesis is verified, a SocialModel of the SA is maintained by the UpdateSM function.

```
agent PA (event)
referTo: Memory, Knowledge,  Goal, UserModel,  SocialModel;
  Memory      <- Cognition( event);
  UserModel  <- UpdateUM(Memory);
  SocialModel<- UpdateSM( Memory, UserModel );
  action         <- Decision( Memory, UserModel, SocialModel);
  Memory  <- Intention( action, Memory);
do action
```

Figure 8. Procedure of a Personal Assistant of OMAS platform

```
agent SA (event)
referTo: Memory, Knowledge,  Goal,  SocialModel, SocialHypothesis;
  Memory      <- Cognition( event);
  SocialHypothesis <- Watch( Memory);
  SocialModel <- UpdateSM(Memory, SocialHypothesisl);
  action          <- Decision( Memory, SocialModel, SocialHypothesis);;
   Memory  <- Intention( action, Memory);
do action
```

Figure 9. Procedure of Social Agent of OMAS platform

VI. An Example of an Agent Space for Office Workers

Fig. 10 shows an example of a system for applying our AS design to support workers at an office space. Agents in Fig. 10 run persistently in the AS playing different roles. In a simple situation we assumed that a worker user-1 and a worker user-2 are working in the same office. A personal agent PA_1 watches user-1 and another, $PA_{2,}$ watches user-2 continuously through sensors. PA_1 and PA_2 have models of their users and user's social activities. For example the user is working in the office now. However, if PA_1 doesn't know that user-2 is a colleague of user-1 at a company he belongs to, then the PA_1 does not know the name of the PA of user-2. PA_2 seems to be in the same situation. A problem in this example is how the two agents can cooperate in order to support a collaborative relation between their masters working in the office in the RS.

We are aiming at solving the problem to develop an SA to watch the office space using sensors. The SA has a social model of collaboration at an office and regards the two users as a pair of coworkers when a user comes close to the desk of another user to engage in a conversation. The information of a relation that user-1 and user-2 work together at an office may be sent to an MA that supports a group of PAs of workers in an office space. We suppose that, when a worker begins to start working in the office space, the PA of the user registers to an MA. Then the MA knows that PA_1 and PA_2 are personal agents of coworkers and sends that information to PA_1 and PA_2. Then, the two PAs include the knowledge into their social model. Using that knowledge, the two PAs can support the collaboration of the two users as shown in Fig. 11.

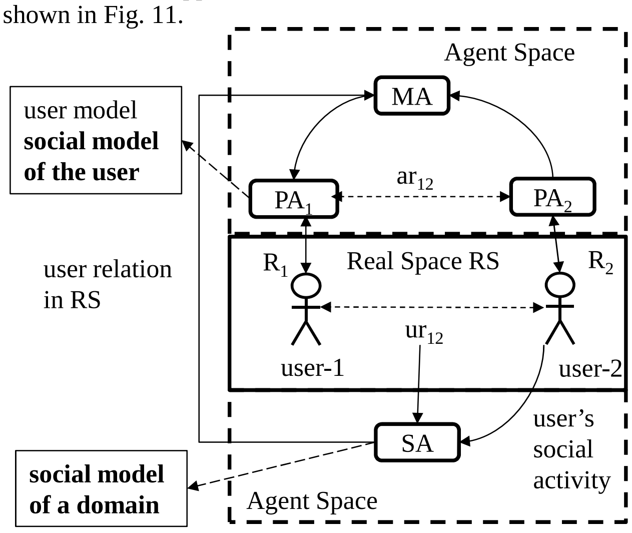

Figure 10. Architecture of a Primitive Agent of OMAS platform

We are developing an example of an Agent Space on the OMAS agent platform. The platform provides an agent programming language, an agent communication language and a knowledge representation language to incorporate knowledge bases in any agent for developers of agent systems to install the functions of BAGs designed in Fig. 3. PAs and SAs receive messages from sensors via Connection Programs programmed in C++. The Kinect, pressure sensors, cameras and microphones are used as sensors in this example. Each PA converse with a particular user using natural language through a keyboard and a display and through a vocal speech interface partially.

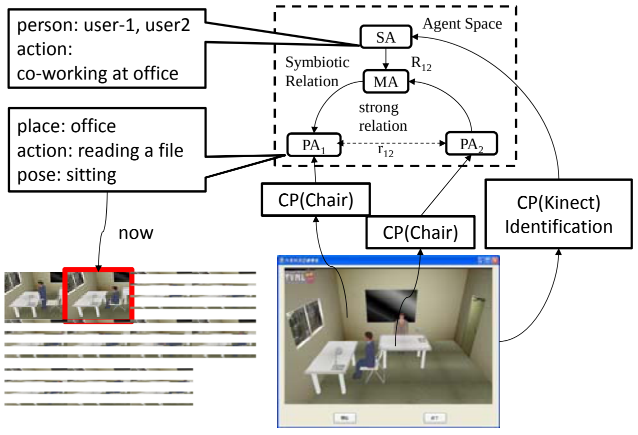

Figure 11. An Example of a Relation between two Workers

A set of descriptions that represent sitting poses of a worker in a knowledge base of a user model is common to all PAs and is initially installed in each PA. A PA should acquire social situations of work dynamically.

VII. CONCLUSION AMD FUTUREWORK

We aimed at developing an Agent Space containing Personal Assistant, Social Agents and Mediation Agents, in order to make information systems in the Digital Space user-centric and to reduce users' cognitive load. Our approach to develop user-centric information systems was to introduce an agentification method using an agent platform in order to connect the agent space to the Real Space and to the Digital Space. The agentification function wraps a sensor and a recognition program into an agent that works on the agent platform. By cooperating with sensing agents developed using this method, a Personal Assistant watches over a user and a SA watches over relations among users.

A Personal Assistant has several knowledge bases for recognizing actions of its master and a Social Agent has several knowledge bases for recognizing relations between users. If enough knowledge can be acquired in the knowledge bases of these agents, then they can support a group of users by understanding the situations of the users. Generally, when an agent is started on an agent platform, it has only initial knowledge bases consisting of typical knowledge. To better support its user, it should acquire proper knowledge from the Real Space and the Digital Space. A Mediation Agent is expected to support the knowledge acquisition for the agents.

Our models of Social Agents and Mediation Agents are still conceptual and just prototypes. In order for the agents to play important roles that are expected by users, enhanced methodology for knowledge acquisition and learning functions of agents will be required in future. We plan to expand the network of OMAS agent platform to global users. At the present, the agent platforms in France, Brazil, Mexico and Japan are connected and worked. In this experiment, we found that we should solve multi-lingual problems and also multi-culture problems in the Agent Space. Finally, the problem of the security and safety in Activity in Daily Life supporting by the Agent Space is very difficult and important problem we have to consider in future work.

Acknowledgment

We would like to thank colleagues working in projects of Symbiotic Computing (http://symbiotic.agent-town.com/) and OMAS framework.

REFERENCES

[1] K. Lyytinen and Y. Yoo: Issue and Challenges in Ubiquitous Computing, CACM, vol. 45, No. 12, pp. 63-65, 2002

[2] Kitsuregawa, M.: Info-plosion: Retrospection and Outlook, The Journal of the Institute of Electronics, Information, and Communication Engineers, Vol.94, No.8, pp.662–666 (2011)

[3] Barthès, J.-P.A., OMAS - A flexible multi-agent environment for CSCWD, Future Generation Computer Systems, vol. 27, pp. 78-87, 2011

[4] Barthès, J.-P.A., UTC/GI/DI/N196 PDM4 (pdf 728K), http://www.utc.fr/~barthes/references/index.html.

[5] Barthès, J.-P.A., UTC/GI/DI/N219 - MOSS 7 Primer (pdf 617K) http://www.utc.fr/~barthes/references/index.html

[6] N. Shiratori, K. Sugawara,Y. Manabe, S. Fujita, and B. Chakraborty, Symbiotic Computing Based Approach towards Reducing User's Burden Due to Information Explosion(Invited Paper), IPSJ Journal Vol. 51 No. 12 1234-1243, 2011M. Young, The Technical Writer's Handbook. Mill Valley, CA: University Science, 1989.

[7] K. Sugawara, S. Ben Yaala, Y. Manabe, C. Moulin, N. Shiratori, and J. A. Barthès, Conversation-based Support for Requirement Definition by a Personal Design Assistant, Proc. of 10th IEEE International Conference on Cognitive Informatics and Cognitive Computing, pp.262-268, Banff, August 18-20, 2011

[8] K. Sugawara, S. Ben Yaala, Y. Manabe, C. Moulin, N. Shiratori, and J. A. Barthès, Conversation-based Support for Requirement Definition by a Personal Design Assistant, Proc. of 10th IEEE International Conference on Cognitive Informatics and Cognitive Computing, pp.262-268, Banff, August 18-20, 2011

[9] K. Sugawara, and S. Fujita, Non-verbal Interface of a Personal Agent based on Symbiotic Computing Model, Proc. of 10th IEEE International Conference on Cognitive Informatics and Cognitive Computing, pp.336-339, Banff, August 18-20, 2011

[10] S. Russel, and P. Norvig, Artificial Intelligence A Modern Approach, Prentice Hall, 2003

[11] T. Suganuma, S. Imai, T. Kinoshita, K. Sugawara, and N. Shiratori, "A Flexible Videoconference System based on Multiagent Framework", IEEE Transactions on Systems, Man, and Cybernetics part A, Vol.33, No.5, pp.633-641, 2003.