Design of Multiagent-based Asynchronous Messaging System

Takuo Suganuma*, Jiro Sekiba*, Gen Kitagata*,
Tetsuo Kinoshita*, Ken-ichi Okada** and Norio Shiratori*

*Research Institute of Electrical Communication /
Graduate School of Information Sciences, Tohoku University
2-1-1, Katahira, Aoba, Sendai 980-8577, Japan
E-mail: {jir,minatsu,suganuma,kino,norio}@shiratori.riec.tohoku.ac.jp
**Information and Computer Science Department, Keio University
14-1, Hiyoshi 3-chome, Kohoku-ku, Yokohama 223-0061, Japan
E-mail: okada@inst.keio.ac.jp

Abstract

Asynchronous Messaging Systems like e-mail systems today need some advanced features, such as intelligence, controllability and scalability, to accomplish more effective and sophisticated message handling. In this paper, we propose a framework of Flexible Asynchronous Messaging System (FAMES), which consists of autonomous and collaborative software agents. In FAMES, various messaging functions composed by agents can be utilized to integrate heterogeneous user environment. Moreover, FAMES operates message flow in intelligent manner, considering the receiver's own convenience and the flexible message delivery can be achieved. We designed and implemented the proposed system based on multi-agent technology, and have shown its effectiveness through the experimental studies using the prototype system.

1. Introduction

As the Internet and personal computers are widely used, lot of people come to have their own individual environments on computers wired to the net at their home or office. On those environments, asynchronous messaging systems, standing for e-mail systems, has become one of the most popular tools to communicate with others. The asynchronous messaging system is useful in various applications, however, many problems remain in terms of user's viewpoint. For instance, when a user wants to use extended messaging functions other than ordinal ones, such as message cancellation or circulation, it may be impossible unless all recipients' messaging environments support the specific functions. Since the asynchronous messaging system is realized as a loose coupled distributed system, their environments tend to be highly heterogeneous. Such heterogeneity makes the enhancement of messaging functions very difficult. On the other hand, if a reply to a received message is delayed due to absence or convenience of a recipient, messaging process will not be completed as sender's wish. Although such inconvenience originates from the intrinsic property of the asynchronous systems, any systematic supports are not considered.

All these problems may be caused by inflexibility in traditional asynchronous messaging systems against various types of changes of both systems' situations and users' demands. In this paper, to solve these problems, we propose a framework of Flexible Asynchronous Messaging System (FAMES), which is applied the concept of "Flexible Networks" [1,2] on traditional e-mail environment. FAMES is a system aimed for more effective and sophisticated asynchronous message processing, by adding following mechanisms as, (a)adaptive service reconfiguration mechanism, (b)user-centered flexible messaging mechanism, and (c)function abstraction mechanism, on traditional system. In FAMES, various messaging functions realized as software agents are utilized to integrate heterogeneous user environments. Moreover, since FAMES provides intelligent message controls for users considering receiver's own convenience, flexible message delivery can be attained. We have designed and implemented the proposed FAMES as a multi-agent system on bases of agent-oriented computing environment called ADIPS framework[3], developed by authors. Using a prototype system, we have confirmed the effectiveness of the proposed FAMES.

2. Framework of FAMES

2.1. Limitations of traditional messaging systems

In traditional asynchronous messaging systems, following limitations have been pointed out, i.e.,

(P1) Lack of dynamic service reconfiguration capability: When a user requires an extended function to his/her own familiar e-mail client, both of sender and receiver e-mail clients have to select and utilize the proper function modules tailored to their own environments. Due to heterogeneity of individual e-mail environment customized to respective users, it is difficult to adopt or change the functions at run-time dynamically. Although several e-mail clients which the additional function modules can be "plugged in" have been proposed, the problems such as how to choose, install and maintain the adequate modules are still open.

(P2) Lack of user-centered effective messaging capability: The advanced messaging features, such as cancellation, delivery confirmation or circulation of messages, can not be supported in a systematic way. Several groupware applications can offer some of these functions, but it requires to replace the users' messaging environments. Therefore, it may not be a good approach from user's point of view. Furthermore, when a member of message circulation is absent, the circulation task may hang up at the place. This may cause inconvenient situation in ongoing business process.

(P3) Lack of absorption capability to reduce functional differences: No matter what different e-mail clients have the same function, there is no way to guarantee the interoperability between them. Therefore users can not revive the required functions each other.

The reason why these problems are caused can be explained by analyzing the intrinsic properties of the system in two dimensions, that is, system architectural property and temporal asynchronous property. The system tends to be constructed in highly distributed and loose coupled architecture. This means that the system will possibly be uncontrollable and heterogeneous. To make matters worse, temporal asynchronous property allows users to do their own convenience, which may cause undesirable effects to whole system behavior.

2.2. Required functions of FAMES

We propose a framework called Flexible Asynchronous Messaging System: FAMES to solve the problems described in section 2.1. FAMES is reached by adding the following mechanisms to the ordinal messaging systems.

(M1) Adaptive service reconfiguration mechanism: This is a mechanism which percepts both the users' demands and the status of systems' environments, reconfigures by themselves automatically, and offers the most suitable services to users. For example, when a user wants to cancel a message sent to others by using his/her e-mail client which does not support a cancellation service, the most suitable module which can realize the cancellation is instantiated automatically, and incorporated to the current e-mail client. After cancel action is done, the module would be removed by itself autonomously. This mechanism solves problem(P1).

(M2) User-centered flexible messaging mechanism: This is a mechanism to realize an intelligent decision making to control message flow, based on users' demands, their own conveniences, and status of ongoing tasks. For example, suppose that user-A wants to circulate a message in an order of user-B, user-C, user-D and user-E, and that user-D plans to make a business trip tomorrow. Then sequence of the circulation may be reordered automatically to give a priority to user-D. This will solve problem(P2).

(M3) Function abstraction Mechanism: This is a mechanism to bridge functional gaps between various different e-mail clients by abstracting their functions. Using this mechanism, interoperability between the different implementations of the same functionality can be guaranteed. This will solve problem(P3).

2.3. Applying agent-based computing technology

To realize the mechanisms described in section 2.2, we adopt agent-oriented approach, in which software agents provide the required functions based on the following characteristics, i.e., autonomy, controllability, self-recognition, adaptability, mobility, cooperative problem solving capability and function abstraction.

(1)Realization of adaptive service reconfiguration (M1): A personal environment of a user is realized as an organization of agents. According to the agents' characteristics such as self-recognition and adaptability, they can percept the environment autonomously. The agents are instantiated to the user's environment on demand, based on its autonomy and mobility. These agents reconfigure their organization, and provide the required services to users.

(2)Realization of user-centered flexible messaging (M2): Making whole system as an organization of agents, the message flow can be controlled with respect to users' demands based on cooperative problem solving capability of agents.

(3)Realization of function abstraction (M3): Reforming the conventional e-mail clients as agents by agentification operation, each peculiar function of e-mail client is abstracted, therefore it can get to connect together using common interfaces and protocols among agents.

3. Design of FAMES

3.1. ADIPS framework

We make use of ADIPS (Agent-based Distributed Information Processing System) framework[3] as a platform of design and implementation of an agent-based prototype system of FAMES. ADIPS framework is an agent-based computing infrastructure proposed by the authors aiming for construction of flexible distributed systems[4]. ADIPS framework has the following advantages to realize the agent-oriented facilities of FAMES we need, i.e., (i) a system is autonomously organized/reorganized in user-driven and event-driven manner, (ii) agents use expertise of domain experts, such as designer, administrator and operator of distributed system, (iii) legacy computational processes or application softwares can be used as reusable components by agentification operation. In ADIPS framework, user agents request tasks to the ADIPS Repository on users' demands, then the most suitable agents in ADIPS Repository are bided based on contract net protocol[5], and the selected agents are instantiated in a personal user environment called ADIPS Workspace. Agents which are instantiated onto Workspaces communicate with other agents in their own Workspaces or in other users' Workspaces, and they cope with users' demands and change-full environments flexibly.

3.2. Conceptual design of FAMES based on ADIPS framework

Figure 1 shows a conceptual model of applying FAMES to the ordinal e-mail environment. (a) in Figure 1 stands for traditional common e-mail environment. Here, sender's e-mail client sends e-mails to a mailhost by a message transfer protocol, SMTP, then the mailhost deliver e-mails to the destination mailhost. Receivers' e-mail client gets delivered e-mails from mailhost using a message retrieving protocol, such as POP and so on. We apply the proposed framework to this e-mail environment, shown in (b) of Figure 1. E-mail clients and mailhosts are agentified in accordance with ADIPS framework, and they play their roles as a part of agent system. Since the agents use traditional e-mail environment to deliver e-mails' contents, they can accomplish the enhanced message delivery platform without any introduction of new delivery infrastructures (realization of (M1)). E-mail delivery is controlled by agents in ADIPS Workspace, thus, it can make intelligent delivery control achieve(realization of (M2)). Moreover, since the agentification using ADIPS

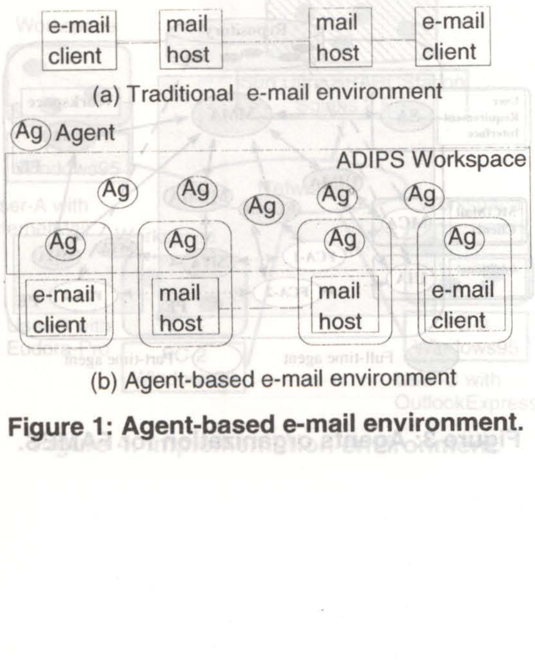


Figure 1: Agent-based e-mail environment.

framework encapsulates the differences of implementations, many types of e-mail clients can cooperate using inter-agent communication protocols defined by ADIPS framework(realization of (M3)). The reason why we apply this framework is that all of the required mechanisms for FAMES can be reached by the framework comprehensively and effectively.

Figure 2 shows examples of circulation and cancellation function realized by ADIPS framework. Circulation is a function by which a user touch up a received e-mail, and just reply to it, then it is delivered automatically to the next circulation address. In FAMES, users can utilize their favorite e-mail clients to browse, touch up and reply to. First sender specifies the circulation addresses. For an e-mail client without any circulation functions, an agent autonomously detects the absence of the function, instantiates from Repository, and offers the function to realize circulation processing. On the other hand, cancellation is a function by which a user can cancel a delivered e-mail. Receiving a cancel request issued by a user, cancellation agents are instantiated to both sender site and receiver site from Repository. Then cancellation is performed by negotiation between cancellation agents.

3.3. Design of agents for FAMES

To realize the mechanisms of FAMES described in section 2.2, we designed agents and their organization based on ADIPS framework. Agents for FAMES are organized in Personal Environments(PEs). A PE is a logical agent workspace existing in ADIPS Workspace, and it is dedicated to a user to give the individual messaging services. In each PE, two kinds of agents reside, namely full-time agents and part-time agents. Full-time agent exists in the PE permanently to perform such

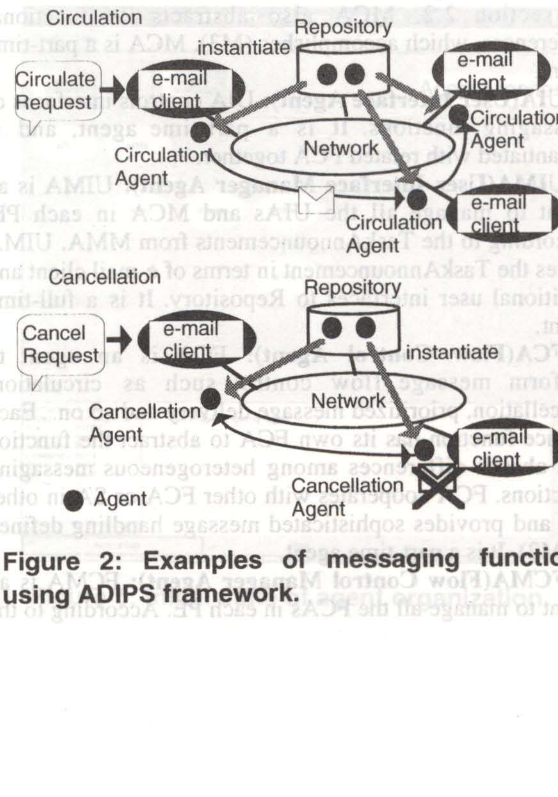


Figure 2: Examples of messaging function using ADIPS framework.

constant tasks as monitoring status of users and environments, controlling activities of agent organization, spooling the incoming messages, and so forth. While part-time agent is instantiated from Repository on demand, and exists in the PE temporary. The configuration of part-time agents is dynamically changed in run-time to provide necessary and sufficient services.

Figure 3 shows the organization of agents for FAMES. The role of each agent is defined as follows;

(1)SA(Secretary Agent): SA is a mediator between human user and software agents. Human user tells his/her requirements to agents via SA, and SA shows various information from agents to user. SA has an user interface to interact with user. When SA receives the requirement from a user, it sends the TaskAnnouncement to Message Manager Agent. SA also manages user's preference and personal information such as schedules. The information is utilized to realize user-centric functions such as (M2) in section 2.2. SA is a full-time agent.

(2)MMA(Message Manager Agent): MMA is a central controller of agents in a PE. It is responsible for managing the whole agent organization in each PE. It also replies to a query from other agents concerning information about the organization which it takes charge of. When MMA receives a TaskAnnouncement from SA, it decomposes the announcement and sends the TaskAnnouncement to UIMA, FCMA, and MTMA, respectively. If MMA judges that the required task from SA needs to change of agent organization in other PE, it also propagates the TaskAnnouncement to MMA in other PE. MMA is a full-time agent.

(3)MCA(Mail Client Agent): MCA is an agent realized by agentification of existing e-mail client software, and controls the software directly. MCA recognizes the functionality and capability of respective e-mail client, so it can analyze whether required functions can be provided or not. This capability of MCA is utilized to achieve (M1) in section 2.2. MCA also abstracts its functional differences, which accomplishes (M3). MCA is a part-time agent.

(4)UIA(User Interface Agent): UIA controls interfaces of messaging functions. It is a part-time agent, and is instantiated with related FCA together.

(5)UIMA(User Interface Manager Agent): UIMA is an agent to manage all the UIAs and MCA in each PE. According to the TaskAnnouncements from MMA, UIMA issues the TaskAnnouncement in terms of e-mail client and additional user interfaces to Repository. It is a full-time agent.

(6)FCA(Flow Control Agent): FCA is an agent to perform message flow control such as circulation, cancellation, prioritized message delivery and so on. Each service function has its own FCA to abstract the function and absorb differences among heterogeneous messaging functions. FCA cooperates with other FCA or SA in other PE, and provides sophisticated message handling defined in (M2). It is a part-time agent.

(7)FCMA(Flow Control Manager Agent): FCMA is an agent to manage all the FCAs in each PE. According to the TaskAnnouncements from MMA, FCMA issues the TaskAnnouncement in terms of message flow control to Repository. It is a full-time agent.

(8)MTA(Message Transfer Agent): MTA is an agent to perform message delivery actually. MTA receives address from FCA and contents from MCA, and sends it to addressed receiver MTA. MTA also receives messages from another MTA and spools them. It is a full-time agent.

(9)MTMA(Message Transfer Manager Agent): MTMA is an agent to manage all the MTAs in each PE. It is a full-time agent.

4. Implementation of FAMES

We have implemented a prototype of FAMES based on an architecture explained in Section 3. The implementation environment is shown in Figure 4. The hardware environment consists of an Sun Ultra SPARCStation with Solaris2.5.1 operating system, and three personal computers with MS-Windows95, which are all connected by LAN. As to the software environment, we utilized ADIPS/Java, the latest version of ADIPS framework written in Java language. ADIPS framework was configured by installing ADIPS Repository on the SPARCStation and ADIPS Workspaces on each personal computers. We also prepared the famous legacy e-mail clients, i.e., MS InternetMail, MS Outlook Express and Eudora-Pro. On the hardware/software environment stated previously, we implemented agents in accordance with the design in section 3. SA, MMA, UIMA, FCMA, MTMA and MTA are implemented as full-time agents. As FCA, we developed Cancellation agent and Circulation agent, which performs message cancellation and message circulation, respectively. With FCAs, UIAs for cancellation and circulation are also developed. UIAs have function-oriented user interface processes which are agentified to incorporate UIAs. MCAs are implemented by agentifying three legacy e-mail clients. Each user interface and messaging function, we call base processes, are written in Java. To construct FAMES, 12 agents are implemented

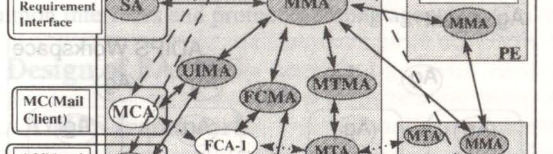


Figure 3: Agents organization for FAMES.

totally and whole code size to describe agents are approximately 2500 lines. While the total size of base processes are approximately 4000 lines.

5. Experiment

In this paper, we evaluate the effectiveness of one of the proposed mechanisms defined in section 2.2, i.e., adaptive service reconfiguration mechanism(M1), through an experimentation. Adaptive service reconfiguration mechanism is realized by cooperative behavior of agents designed and implemented in section 3 and 4.

We experimented in the environment shown in figure 4 under a situation that User-A uses InternetMail on PC-1, User-B uses Eudora-Pro on PC-2 and User-C uses Outlook Express on PC-3. All PCs are wired by LAN. Since those e-mail clients don't have circulation function and cancellation function, no users can use such additional functions. Even if OutlookExpress has an additional function such as circulation or cancellation, the user of it can not use the function because rest of two e-mail clients can not utilize such function. Furthermore, even if all of those e-mail clients have such functions, users may not be able to use the functions because there is no interoperability between the functions.

In the prototype of FAMES, as we mentioned above, SA, MMA, UIMA, FCMA and MTA are instantiated from Repository as full-time agent when the system booted up, and the agent organization is composed as shown in figure 5. Then, User-A wanted to make a circulation message. Using FAMES, User-A could make a request through an interface provided by SA. Additional functions available are presented in the interface. This information is acquired by cooperation among SA and agents in the Repository. Users can instantiate any combination of e-mail clients and additional functions, thus users can use the functions together with their favorite e-mail clients. In this experiment, User-A chosen Microsoft InternetMail to browse or compose e-mails. Then, User-A selected circulation function from the interface of SA to send a circulation message. SA perceived user request and issued a TaskAnnouncement to MMA. MMA decomposed that task and issued the TaskAnnouncement to manager agents, i.e., UIMA, FCMA and MTMA. Each manager agent autonomously detected that it could not provide the requested service for User-A, and then made a TaskAnnouncement to Repository in order to realize this function. In the Repository, Circulation FCA and Circulation UIA were awarded with respect to this request based on the contract-net protocol, and instantiated autonomously in the user's Workspace. Then agents in Workspace reorganized their own organization autonomously, and realize the circulation function which user's e-mail clients didn't have, as shown in Figure 6.

Through the circulation interface, User-A set an order of circulation, such as User-A -> User-B -> User-C. In this case, User-B receives first, and next User-C receives the circulation message. Request of circulation was accepted by Circulation FCA of User-A, then this agent started to negotiate with Circulation FCA of User-B which is instantiated autonomously from Repository. User-B used Eudora-Pro which does not support circulation function. User-B received the circulation message as if it had been just an ordinary e-mail for Eudora-Pro. When User-B just replied, Circulation FCA perceived that it is a circulation message to be sent to User-C. This experiment represents that a circulation function was achieved among InternetMail, Eudora-Pro and OutlookExpress, which haven't got circulation function.

In the same way, when User-A made a cancellation request, the request was received by SA. Agents on the Workspace autonomously detected that they can not provide the requested service for User-A, and issued a TaskAnnouncement to realize this function to Repository. Cancel FCA and Cancel UIA were instantiated

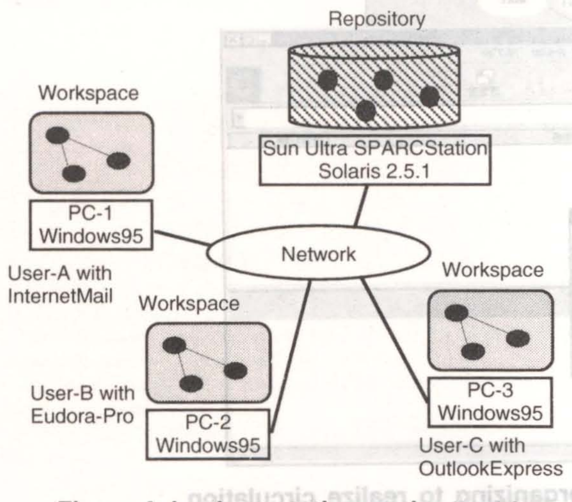
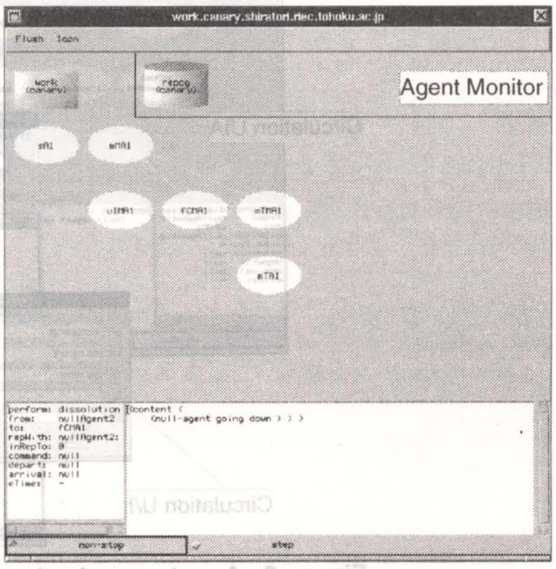


Figure 4: Implementation environment.



```
perform:  dissolution  (content (
from:     nullAgent2       (null-agent going down ) ) )
to:       fCMA1
repWith:  nullAgent2:
inRepTo:  0
command:  null
depart:   null
arrival:  null
eTime:    -
```

Figure 5: Initial state of agent organization.

autonomously in the user's Workspace. Then agents in Workspace tried to reorganize their organization and realized the cancellation function which user's e-mail clients didn't have. Here User-A requested cancellation message through the interface of Cancel UIA. The request was accepted by Cancel FCA, then it started to negotiate with Cancel FCA of other users. Cancel FCA of other user were autonomously instantiated from Repository to negotiate, accepted the cancel request and removed the message from e-mail spool if message was not read. Through this experiment, we confirmed that cancellation function can be achieved among InternetMail, Eudora-Pro and OutlookExpress, which doesn't support the cancellation function.

As a results, we confirmed that the adaptive service reconfiguration mechanism(M1) was realized, and it is useful and effective from user's view point. Using this mechanism, the system can provide services adaptively which are not supported by e-mail clients in accordance with various users' requests.

6. Conclusion

In this paper, we proposed a framework of Flexible Asynchronous Messaging System (FAMES), which consists of autonomous and collaborative software agents. Asynchronous Messaging Systems like e-mail systems today need some advanced features, such as intelligence, controllability and scalability, to accomplish more effective and sophisticated message handling. In FAMES, various messaging functions composed by agents can be utilized to integrate heterogeneous user environment. Moreover, FAMES operates message flow in intelligent manner, considering the receiver's own convenience and the flexible message delivery can be achieved. We designed and implemented the proposed system based on multi-agent technology, and have shown its effectiveness through the experimental studies using the prototype system.

[References]

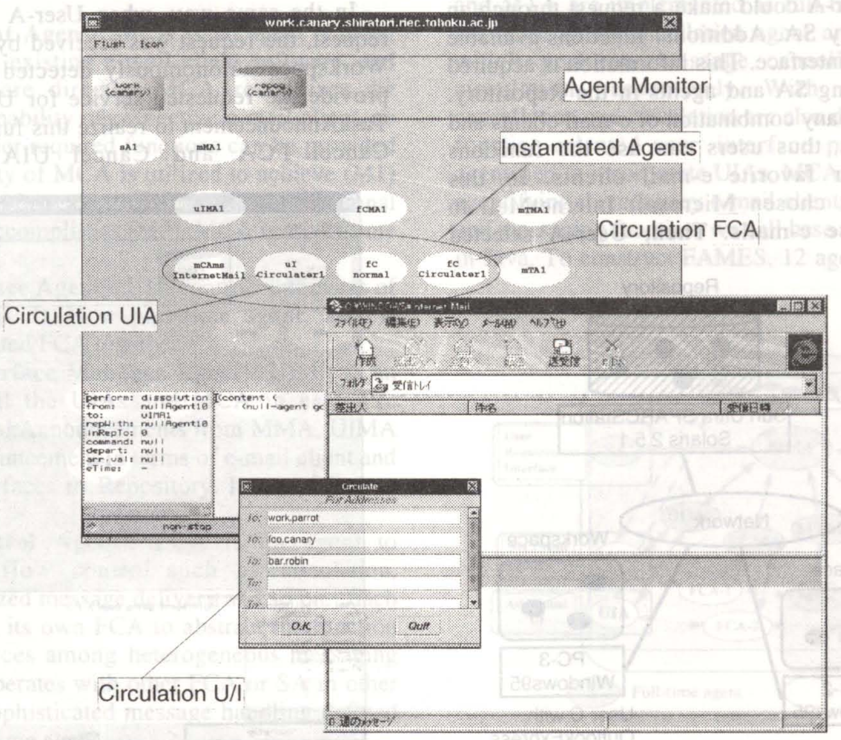
[1]N. Shiratori, K, Sugawara. T, Kinoshita. and G. Chakraborty: Flexible Networks: Basic concepts and Architecture, IEICE Trans. Comm., Vol. E77-B, No. 11, pp.1287-1294(1994).

[2]N. Shiratori, S. Suganuma, S. Sugiura, G. Chakraborty, K. Sugawara, T. Kinoshita. and E.S.Lee.: Framework of a flexible computer communication network. Computer Communications, Vol.19, pp. 1268-1275, 1996.

[3]S. Fujita, K. Sugawara, T. Kinoshita, and N. Shiratori: Agent-based Architecture of Distributed Information Processing Systems, Trans. IPS. Japan Vol. 37, No. 5, pp. 840-852, 1996.

[4]S. Fujita, H. Hara, K. Sugawara, T. Kinoshita and N. Shiratori: Agent-Based Design Model of Adaptive Distributed Systems, Applied Intelligence, Vol. 9, No. 1, 1998.

[5]R.G.Smith: The contract net protocol: High-level communication and control in a distributed problem solver, IEEE Trans. on Computers, Vol. 29, No. 12, pp. 1104-1113, 1980.



```
perform:  dissolution  [content (
from:     nullAgent10      (null-agent gc
to:       uIMA1
repWith:  nullAgent10
inRepTo:  0
command:  null
depart:   null
arrival:  null
eTime:    -
```

To: work.parrot
To: foo.canary
To: bar.robin

Figure 6: Agent organization after reorganizing to realize circulation function.