# ADIPS Framework for Flexible Distributed Systems

Tetsuo Kinoshita[1] and Kenji Sugawara[2]

[1] Research Institute of Electrical Communication, Tohoku University,
2-1-1 Katahira Aoba-ku, Sendai 980-8577, Japan
kino@riec.tohoku.ac.jp
[2] Dep. of Network Science, Chiba Institute of Technology,
2-17-1 Tsudanuma, Narashino 275-0016, Japan
suga@suga.cs.it-chiba.ac.jp

**Abstract.** A next generation distributed system is expected to be flexible in the sense that the system is able to deal with various changes of both the users' requirements and the operational conditions of system's environment. The aim of our research is to establish a new design methodology of the flexible distributed systems based on agent-based computing technology. To do so, we propose an agent-based distributed information processing system (ADIPS) as a design model of flexible distributed systems. Furthermore, we have developed an agent-based computing framework called ADIPS Framework which supports the design and implementation of flexible distributed systems. In this paper, we discuss the architecture and functions of ADIPS Framework together with the applications realized by using ADIPS framework.

## 1 Introduction

The next generation distributed systems have to provide various services for the users living in the networked information spaces realized on the large-scale distributed systems. In the real world, the distributed systems confront with various perturbations of their functions which are caused by the changes of both the users' requirements and operational conditions of the system's environment. As a result, the quality of service of such a distributed system is changed and degraded along with the perturbations of the systems.

Recently, the studies aiming this problem have been started in the fields such as the advanced information networks [2,7,9,10] and the distributed mullet-media application systems [1,8,14,15]. In order to provide the user-oriented, intelligent and stable services for users, a distributed system should have a mechanism to maintain the quality of service against various changes observed in the system's environment. With this mechanism, a distributed system has a capability to deal with the fluctuating operational environment, and we call such a distributed system a flexible distributed system.

The essential functions of flexible distributed system are defined from a view point of changes of both the users' requirements and the operational conditions of the systems' environment. In order to design and implement a flexible distributed system with new functions, we adopt the agent-based computing technology as one of the promising technologies. Although many kinds of architecture and mechanisms of agent-based systems have been proposed and implemented, the effective methodology and tools have not been provided for designers and developers of agent-based systems. Hence, we have proposed and implemented an agent-based computing framework for developing the agent-based flexible distributed systems.

In section 2, we propose a notion of a flexible distributed system (FDS), and discuss the essential functions of FDS. In section 3, an agent-based distributed information processing system (ADIPS) is introduced as a design model of FDS. Then, the ADIPS framework is proposed as a new framework to design and implementation of the agent-based flexible distributed systems based on the ADIPS model. Next, in section 4, we explain the several applications of FDS to show the ability of the ADIPS Framework. Finally, we conclude our work and future problems in section 5.

## 2 Flexible Distributed System

### 2.1 Notion of Flexible Distributed System

A distributed system consists of various distributed computer systems and the information networks which are called a platform of the distributed system. According to the requests given by users, the services of the distributed system are realized by using functions of the platform and provided for users. From the view point of the configuration of the services of distributed systems, the following changes can be considered at the run time of the system.

• variation of the required services: many kinds of services have to be realized in accordance with various user's requests in which the quality of the required services (the required QoS) are specified.

• change of the quality of service (QoS) of a platform: due to the growth of the communication traffic or the computational load of the platform, the QoS provided for users are degraded as the computational resources of the platform are decreased.

• change of the users' requirements: due to change of both the required services and the provided QoS of the platform, the users' requirements can easily be shifted to another one.

A notion of flexibility is expressed as a system's capability to deal with these changes, i.e., the flexibility of a distributed system is that the system can modify its structure and functions against various changes of the system’s operational environment observed at the run time [8]. To do so, a distributed system has to have the following functional characteristics.

• *User-oriented*: a FDS accepts a user’s request at any time and tries to realize the services which maximally satisfy the request from a view points of the user.

• *Homeostatic*: a FDS can modify its structure and functions temporally to maintain the required QoS against various temporal changes observed at the run time in the system's operational environment.

• *Evolutional*: a FDS can change its structure and functions permanently to adapt the permanent or drastic changes of both the users' requirements and the system's operational environment.

A distributed system in which the above three characteristics have been reflected harmonically, is called a Flexible Distributed System (FDS).

### 2.2 Essential Functions of Flexible Distributed System

A distributed system, in general, consists of various functional components located on a platform. According to the notion of FDS, the following three methods would be required to make a distributed system flexible in the perturbed systems environment.

(M1) Tuning: Each component of a distributed system has the operational parameters to change its functional characteristics and behavior. To deal with a slight change of the system environment of the distributed system, a suitable component is selected and the operational parameters of the component are modified.

(M2) Replacement: When a function of a distributed system has to be altered to deal with the changes in the system environment, a component which realizes the function is identified and replaced with a new component.

(M3) Reorganization: To deal with a drastic change of the system environment, a subsystem of a distributed system is redesigned and reorganized into a new subsystem which fits the whole system to the altered environment.

In the design of a FDS, we assume that the FDS consists of a distributed system (DS), a component repository (CR) which holds and manages the components to be used in the DS, and the following five kinds of the functions.

(F1) *Request Acquisition Function (RAF)* receives and analyses the user's requests to detect the changes of the user's requirements.

(F2) *Platform Sensor Function (PSF)* monitors the operational condition of a platform and sends the reports to other functions.

(F3) *Parameter Tuning Function (PTF)* is a function based on the method M1. Receiving the reports on the changes from RAF and PSF, PTF makes a plan to find a component of DS to be tuned, and modify the parameters of the selected component.

(F4) *Component Replacement Function (CRF*) is a function based on the method M2. According to the reports from RAF and PSF, CRF makes a plan to exchange a component of DS with a new component selected by CR.

(F5) *DS Reconstruction Function (DRF)* is a function to deal with a drastic change of the system environment based on the method M3. As same as CRF, receiving the reports from RAF and PSF, DRF determines a subsystem to be redesigned, and then DRF makes a plan to build a new subsystem. According to the plan, DRF selects the components from CR, combines them into a new subsystem, and reorganize the whole system.

To realize these functions, it is convenient to treat the functional components of the distributed systems as high level modules which can be combined each other and organized as the subsystems of the FDS. Hence, the functional components will be transformed to the intelligent agents using knowledge of respective components. In the next section, we propose an agent-based computing framework to support the design and implementation of various agents of the FDS.

## 3 Agent-based Computing Framework for FDS

### 3.1 ADIPS Model

As explained in previous section, the functional components of a distributed system are formalized and realized as the agents which work cooperatively to realize the users' requirements based on knowledge of both the structure and functions (services) of respective agents. However, it makes a design task of the distributed system difficult that the system consists of many kinds of the functional components which the designers have to realize by the new design of components or the reuse design of existing components based on the users' requirements. It is also difficult for the designers to develop various agents and agents' organization from scratch without effective design tools. Hence, we propose a design model of *an agent-based distributed information processing system (ADIPS)* and a design support environment called *ADIPS Framework* [3,4,5].

The features of the ADIPS model can be summarized as follows;

(1) *Agentification of components*: the functional components of distributed systems have been designed and implemented as the computational processes run on the platforms of distributed systems. Acquiring knowledge regarding the design of a computational process called *base process* in this paper, the functions to manage and control the base process as an agent are defined and combined with the base process. This kind of agent is called *primitive agent* in the ADIPS model, and the operations to define the primitive agent is called *agentification* of base process. The ADIPS agent architecture provides an agentification mechanism for designers.

(2) *Requirements-driven design*: according to the users' requirements, the primitive agents can be combined each other to make an organization of agents which provides the requested services for users. In an organization of agents, there exists an agent which is responsible to manage and control an organization and its members. This kind of agent is called *organizational agent* in the ADIPS model. The hierarchical construction of services can easily be designed by defining an organizational agent which holds several organizational agents as its members. Hence, the design of distributed system is reformed into the design of the agents and the organization of agents based on the users' requirements.

(3) *Reuse of assets*: it may be useful to provide the existing sets of both organizational agents and primitive agents for designers in advance, because the designers can select suitable agents which have the required services and functions, and also reuse

these agents to construct the target distributed system. The reuse-based design is one of effective methods to develop the distributed systems in an efficient and systematic way. Using agentification operation, the existing useful base processes can be defined as the primitive agents. The organizational agents can also be memorized as the design cases of services/functions of distributed systems. Accumulating and reusing the assets based on the ADIPS model, the reuse-based design of distributed can be promoted.

Under the ADIPS model, an ADIPS is designed as an organization of agents which consists of both the primitive agents and the organizational agents. The organization of agents can work as an intelligent software robot to provide the required services for users. Moreover, an ADIPS can cooperate with the other ADIPSs and organize a group of ADIPSs to deal with the complex tasks. Such an organization of ADIPSs can be considered as a society of intelligent software robots.

### 3.2 ADIPS Agent Architecture

An ADIPS agent architecture depicted in Fig.1 is introduced to design and implement the primitive agents and the organizational agents. An agentification mechanism consists of three functional modules, i.e., 1) *Cooperation Mechanism* (CM), 2) *Task Processing Mechanism* (TPM), and 3) *Domain Knowledge base* (DK).

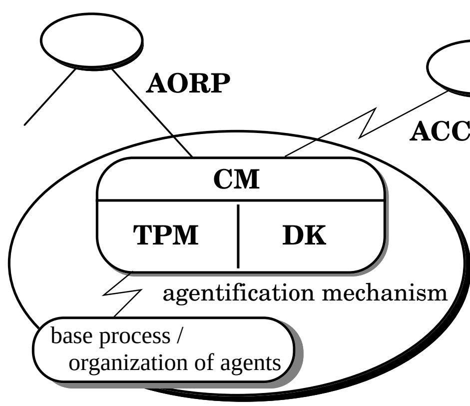


**Fig. 1.** ADIPS Agent Architecture

The CM provides the communication functions not only to exchange information between agents but also to construct /reconstruct the organization of agents. The CM uses two kinds of communication protocols, i.e., *ADIPS organization/reorganization protocol (AORP)* and *ADIPS communication/cooperation protocol (ACCP)*. The details of these protocol are explained in next section.

The TPM is responsible to execute and control a task assigned to an agent based on the intra-agent communication among the CM and DK. A TPM of a primitive agent is responsible to manage the task execution done by its base process. In an organizational agent , a TPM is responsible to manage and control the execution of subtasks assigned to the members of its organization.

The DK holds knowledge and knowledge processing mechanisms to control total behavior of an agent. As shown in Fig.2, a message analyzer receives and classifies the message of both CM and TPM, and selects a suitable message processor to deal with this message. A message processor is a knowledge processing mechanism which has knowledge to process a task specified by the received messages.

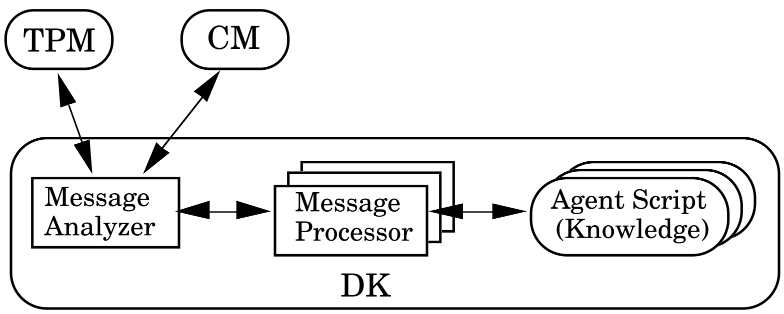


**Fig. 2 .** Structure of DK of ADIPS agent

To realize the essential functions of flexible distributed systems, various knowledge such as the functional specifications of base processes, the heuristic to control the base processes and the organization of agents, the strategies to cooperate with other agents and so on, have to be acquired and represented as the agent scripts by using *ADIPS/L scripting language*. Moreover, many kinds of knowledge processing mechanisms can be defined and utilized as the message processors to make an agent more intelligent. Depend upon the capabilities of the DK, the designers can develop various agents from deliberative type to reactive type.

### 3.3 ADIPS Framework

The ADIPS Framework is an agent-based computing environment to implement and execute the ADIPSs based on various users' requirements. The ADIPS framework consists of three subsystems, as shown in Fig.3, i.e., 1) *ADIPS Workspace* (AWS) which is an agents' operational environment, 2) *ADIPS Repository* (ARS) which is a storage with the functions to manage and utilize the reusable agents, and 3) *ADIPS Design Support* (ASP) which provides the facilities for designers to develop various agents based on ADIPS model.

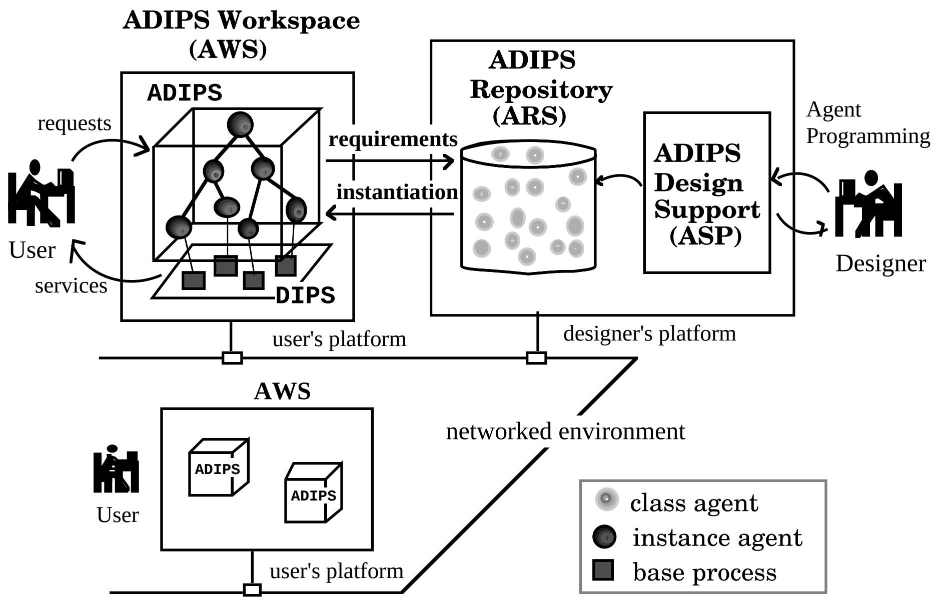


**Fig. 3.** ADIPS Framework

From a view point of implementation of the ADIPS Framework, the agents are classified to two types of agents, i.e., class agent and instance agent. An agent stored in the ARS is called a *class agent* which is designed as a primitive/organizational agent and managed as a reusable component of ADIPS in the functional class hierarchies. On the other hand, an agent run on the AWS is called an *instance agent* which is generated as an instance of a class agent in the ARS to realize an executable component of ADIPS.

An AWS is allocated as an agents' operational environment on a distributed platform. According to the structure and functions of an ADIPS to be designed, a lot of AWSs can be installed on many platforms.

According to the users' requirements, the instance agents of an ADIPS are created from the class agents in the ARS by using the ADIPS organization/reorganization protocol (AORP). For instance, a user can send a message for requiring a new service to an instance agent such as secretary agent. Then, the secretary agent sends a message for generating the requested service to an ARS run on the other distributed platform. Responding to the message, the ARS creates the suitable instance agents on the designated AWS. Activating the instance agents, the requested service is provided for the user. Due to the requests of the instance agents or the users, the AWS can also remove the useless instance agents using the AORP.

The instance agents run on the AWSs can communicate with each other by using the ADIPS communication/cooperation protocol (ACCP) which has a set of the customized performatives of the agent communication protocol of KQML.

The ARS and the AWS work together cooperatively based on the AORP. As explained above, the AWS sends a message of requesting a service to the ARS. In the ARS, the received message is sent to the class agents to construct an organization of agents which can provide the requested service. Although this process is basically the same of a task-announcement-bidding process of the contract net protocol, the AORP is defined as a unique inter-agent communication protocol for construction/reconstruction of ADIPS's agents. For instance, it is different from the original contract net protocol to await the awarding until the all members of organization have been fixed. In the ADIPS Framework, the construction of an organization of agents is regarded as a design task of the required service of the ADIPS, and the ARS is responsible not only to determine a total organization of agents which can provide the required service for users, but also to generate the instance agents of the designed organization in the AWS. Moreover, it is required to reconfigure the ADIPSs to maintain the required services against various changes of both the users' requirements and the system environment, as explained in section 2. In such a case, the ARS and the AWS work cooperatively to reconstruct the organization of instance agents based on the AORP. For example, when an organizational agent in the AWS detects the irregular situations and issues a message to replace some of members of its organization, this message is sent to the ARS. In the ARS, several class agents are selected and instantiated again to reconstruct the corresponding organization of agents based on the AORP.

AORP performatives

| | |
|---|---|
| *task-announcement* | inform a task to be done |
| *bid* | response to task-announcement |
| *award* | allocate a task |
| *directed-award* | allocate a task directly |
| *report* | send result to a manager agent |
| *initialize* | setup instance agents |
| *release* | remove useless instance agents |
| *dissolution* | report remove-operation |

ACCP performatives

| | |
|---|---|
| *ask* | send a question |
| *tell* | send an answer to a question |
| *request-information* | require information |
| *information* | response to request-information |
| *request* | send a request |
| *acceptance* | accept a request |
| *refusal* | refuse a request |
| *direction* | require an action of agent |

**Fig. 4.** Example of Performatives used in AORP and ACCP

As explained above, the AORP takes the most important role to realize the essential functions of flexible distributed systems such as the component replacement function and the DS reconstruction function explained in section 2. An example of performatives of the AORP and the ACCP is shown in Fig.4.

The ASP is responsible to help various activities of ADIPS designers. At present, the ASP provides the design support facilities to specify the class agents in the ARS and monitor the behavior of instance agents in the AWS. To support the design of class agents, an *ARS-browser* supports the designers to retrieve, inspect and modify the agent scripts based on ADIPS/L. Since it is difficult to specify knowledge of an new class agent, the standard description formats of agent scripts called *knowledge templates*, are defined and provided for designers. On the other hand, to support the on-line debugging of behavior and knowledge of agents, an *ADIPS-agent-monitor* is designed and implemented to visualize the real time behavior of both the class agents in the ARS and the instance agents in the AWS.

For several years, the prototypes of the ADIPS Framework have been designed and implemented by using different kinds of programming languages. The first generation prototypes was designed in Smalltalk environment, and the second generation prototypes was implemented by using C++ and Tcl/Tk programming languages. The third generation prototype is now implemented in Java environment. The AWS, ARS and ASP are realized on the distributed platforms such as UNIX workstations and Windows-based personal computers with the TCP/IP-based network environment.

## 4 FDS Application Experience

Several applications of flexible distributed systems have been designed and implemented using the ADIPS Framework. In this section, we explain the following three applications briefly to show the capability of the ADIPS Framework.

### 4.1 Flexible Videoconference System

According to the growth and spread of networked environment, various distributed real-time multimedia applications such as videoconference system have been developed and utilized, e.g., CU-SeeMe, VdeoPhone, NetVideo, INRIA Videoconference system. However, it is difficult for naive users to utilize such a videoconference system, because (i) users cannot express their request correctly in detail to get the services, (ii) users cannot select or combine the suitable communication services which require the complex operations for users, (iii) users feel a lot of inconvenience to coordinate and accomplish their ongoing tasks when the quality of the required service is degraded due to the changes of operational conditions of videoconference systems. To solve these problems, a *flexible videoconference system (FVCS)* [11,12] is proposed and implemented based on the ADIPS Framework.

An agent-based architecture of a FVCS is depicted in Fig.5. The FVCS consists of FVCS modules realized as the organization of agents which work in the respective ADIPS workspaces on the distributed platforms. Each FVCS module is designed as an organization of FVCS agents such as service agent, sensor agent and user agent, as shown in Fig.5. There exists two kinds of the service agents, i.e., *service primitive agent* and *service manager agent*.

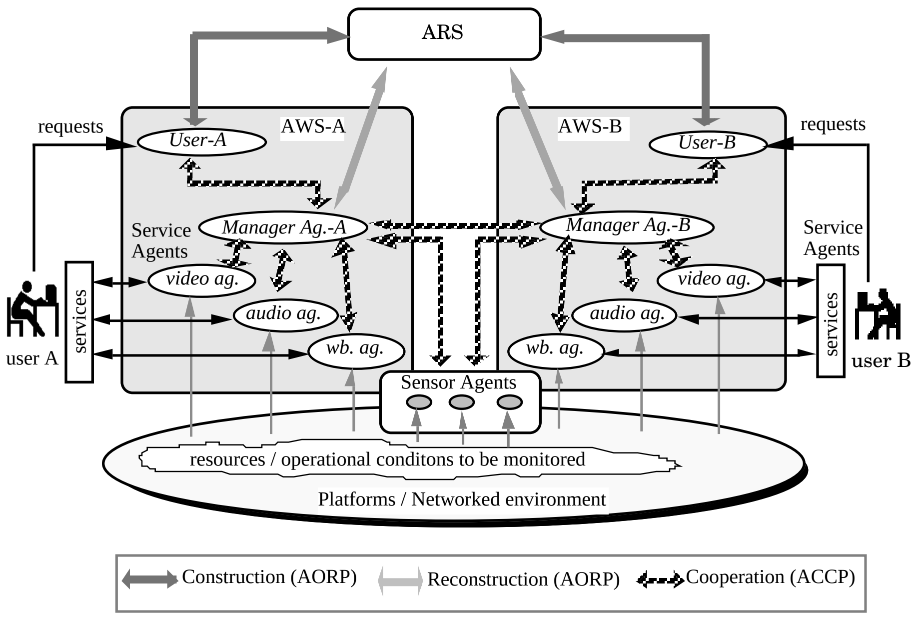


**Fig. 5 .** An architecture of FVCS

A service primitive agent is defined as a primitive agent of ADIPS which has knowledge to monitor and control a base process of multimedia communication. We can utilize the existing software modules as the base processes, for instance, the vic, the nv @and the Vtalk for real-time video services, the Vat for audio service, and the wb for shared text-editing service. A service primitive agent can tune the parameters of a base process to maintain the required QoS under the observed conditions.

A service manager agent is defined as an organizational agents of ADIPS which has knowledge to create and manage the organization of service primitive agents. For instance, a videoconferencing Manager-agent-A in Fig.5, holds knowledge of the real-time communication services and selects a vic-service-agent as a video communication service agent which can provide the most suitable service for users under the operational situations at that time. A service manager agent can make the contracts with a user agent, the sensor agents and the other service manager agents to exchange information on the changes of both the users' requirements and the system's operational situations.

A user agent communicates with its user, accepts the user's requests, and transforms the requests into the descriptions of users' requirements. Furthermore, several kinds of sensor agents are defined as the primitive agents to monitor the static/dynamic operational conditions of the system and detect the changes occurred in the system environment. For instance, a CPU-sensor agent uses a sar-command of UNIX to monitor the current CPU-utilization-rate and a Net-sensor agent also monitors the current status of available bandwidth of communication between platforms by using netperf-command.

We have developed a prototype of FVCS using the ADIPS Framework/C++ version and confirmed that the following functions can be realized.

a) *Automatic composition of services*: according to the users' requirements which specify the service functions together with the required QoS. Using the AORP of ADIPS Framework, the organization of instance agents of FVCS are generated on the AWS and a session of videoconference with the suitable QoS parameters are started by the autonomous operations of FVCS agents.

b) *Autonomous control of the required QoS*: due to the changes of the system environment, the operational conditions of the FVCS have been changed and the required QoS is also fluctuated. Moreover, the users can issue the new requests on QoS to the FVCS freely during the videoconferencing session. The FVCS have to deal with the requests dynamically. Hence, the FVCS can tune the operational parameters of service agents and maintain the required QoS of the videoconferencing session by the cooperative behavior of FVCS agents based on the ACCP.

c) *Autonomous reconfiguration of services*: when some kinds of changes which cannot be handled with the tuning the operational parameters of service agents, the manager agents decide to change the organization of FVCS agents. Using the AORP and ACCP, the manager agents negotiate with each other to replace some of members of organization or to reconfigure the current organization of FVCS agents. Instantiating the new FVCS agents by using the ARS, the videoconferencing session can be continued by the renewed FVCS.

As explained above, the FDS for the real-time (synchronous) distributed applications can be designed and implemented in a systematic way based on the ADIPS Framework.

### 4.2 Flexible Asynchronous Messaging System

An asynchronous messaging system such as electric mail systems, is appreciated as one of useful tools to communicate and exchange information among people reside in distant places. Although many kinds of tools have been developed and utilized by many people, they get confused by the functional difference or heterogeneity of the communication tools which run on different distributed platforms.

For instance, due to a lack of message cancellation functions of recipients' messaging tools, a sender cannot remove or update the wrong messages before the recipients open their mail boxes. Such an inconvenience of users may be one of intrinsic properties of asynchronous messaging systems, however, it is essential to enhance the capabilities of the messaging tools and reduce the burden of users for handling messages within heterogeneous environment. Hence, a *Flexible Asynchronous Messaging System (FAMES)* [6] is proposed based on the ADIPS Framework.

A key concept of the FAMES is the abstraction by agentification of conventional mail hosts in order to realize the adaptive customization of messaging services according to the types of messages. As same as the FVCS, the FAMES consists of the FAMES agents, i.e., *mailing task agent* and *manager agent*, as shown in Fig.6.

A Message Manager Agent (MMA) is a top-level organization agent in a *personal messaging environment* (PE). A Secretary Agent (SA) mediates a human user and the MMA to specify the requests using the user's personal information. According to the requests of both the user and the other MMA of another PE, the MMA manages and controls the behavior of organization of the mailing task agents which are managed by a Flow Control Manager Agent (FCMA), a Message Transfer Manager Agent (MTMA) and a User Interface Manager Agent (UIMA), respectively.

A FCMA organizes and manages a set of Flow Control Agents (FCAs) and a FCA executes a message flow control task such as circulation, cancellation, prioritized message delivery and so on. The FCMA is responsible to create and control of instances of FCAs based on the cooperation with the ARS.

A MTMA manages the Message Transfer Agents (MTAs) in the PE. A MTA is an instance agent which performs a message delivery task together with another MTA of the other PE. The FCA provides a receiver's address for the MTA. The spooling of received messages is also done by the MTA.

A UIMA manages the instance agents of both the Mail Client Agent (MCA) and the User Interface Agent (UIA) in the PE. A MCA is an agent realized by the agentification of existing e-mail client software. The MCA holds knowledge of the user's e-mail client to decide whether the required messaging service can deal with the e-mail client or not. When a new messaging service is required, the MMA creates a new instance of suitable FCA together with a new UIA which provides a user interface of the created FCA.

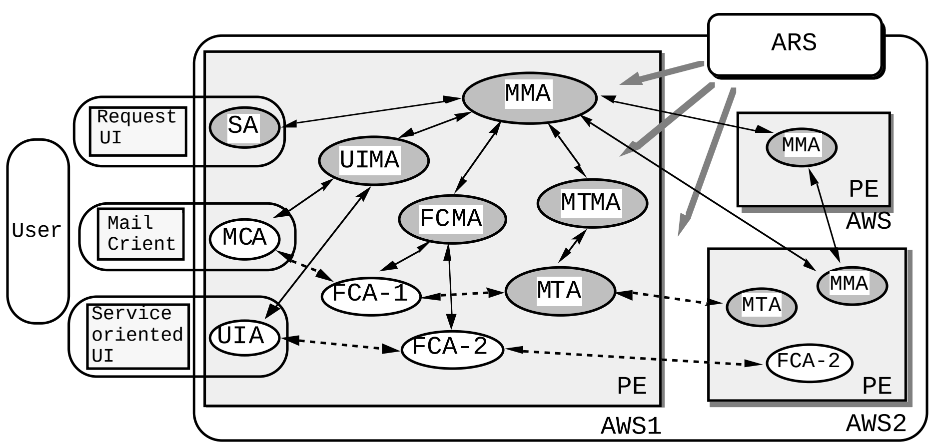


**Fig. 6.** An architecture of FAMES

A prototype of the FCMA is implemented by using the ADIPS Framework/Java version. In the prototype, the mail hosts such as EudraPro, InternatMail, and OutlookExpress, have been agentified and utilized as members of organization of FAMES agents of the users' personal messaging environment (PE), and the following functions can be provided for users.

a) *User-adaptive service customization*: the users of FAMES can give their requests to utilize a messaging service which cannot be supported by the mail hosts at that time. The required service is realized as the organization of FAMES agents and installed in the users' personal messaging environment. Then, the users can utilize the new service together with the messaging services of the mail hosts which the users get familiar.

b) *Requirements-driven service configuration*: a mail host of a user have to prepare a new messaging function to deal with the requests of the other users. Facing such an unexpected request of the users, the manager agents of respective personal messaging environment try to negotiate with each other to establish a cooperative group to process the required messaging tasks such as circulation and cancellation.

Under the ADIPS Framework, various functions for realizing the intelligent messaging services can easily be defined and added as the messaging service agents to enhance the capabilities of the FAMES.

### 4.3 CyberOffice

Recently, an information space over the global networked environment provides a new work place for people so called cyberspace based on the web-technology. To make such a new work place useful and fruitful, various functions to augment and enhance the *digital reality* of the work place have to be provided for people. A *CyberOffice* [13] is proposed as such a new work place based on the ADIPS Framework.

The CyberOffice consists of a *CyberOffice-unit* in a *Symbiotic Space* (SS) and a *Symbiotic Space Base* (SSB) built on a distributed environment, as shown in Fig.7. A cyber office unit is defined based on both knowledge and the models of work/tasks done by people in the real world, and a SS provides a virtual information space with the *social reality* which is one part of the digital reality of the work place. On the other hand, a SSB is realized by the AWSs in which various agents execute the tasks in the CyberOffice-unit. The people in the real world and the agents in the AWSs can collaborate and cooperate with each other as the members of the CyberOffice-unit in the SS.

Since a virtual information space cannot be seen by people in the real world, the cyber office unit has an office room view presented using the VRML in order to enhance the *perceptual reality* which is another part of the digital reality to make people easy to access and collaborate with the members reside in the SS. Therefore, in the CyberOffice, people can recognize the other people and the agents as the human-like avatars which move around the office rooms.

A prototype of the CyberOffice is developed using the ADIPS Framework/Java version which is a recent version of our framework. According to a service request of a member in a CyberOffice-unit, the organization of agents are generated in the SSB using the AORP and the ACCP of ADIPS Framework, and then, the required service is provided for the member. For instance, a human member can ask a secretary agent to get information from a library. The secretary agent sends an information-retrieval request to a manager agent of the library to retrieve the required information. The manager agent sends the request to suitable librarian agents or creates a new organization of librarian agents based on the ARS to deal with the request. Then, the librarian agent sends the retrieved information to the secretary agent and the human member.

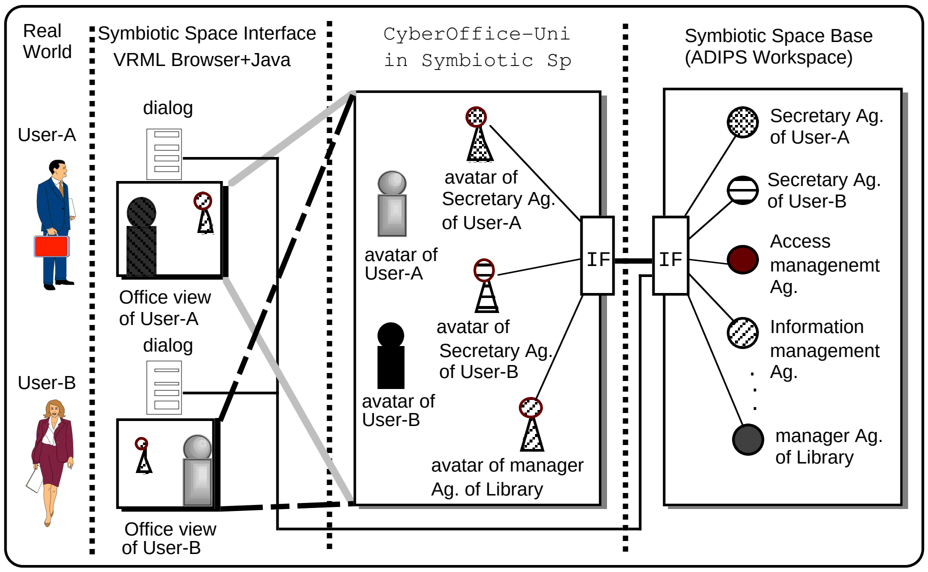


**Fig. 7 .** An architecture of CyberOffice

As explained above, the ADIPS Framework can be utilized to design and implement a new information space over the networked environment, i.e. the CyberOffice with the digital reality in which people and agents can communicate and collaborate with each other to accomplish various work/tasks in the real world.

## 5 Conclusion

In order to realize the next generation distributed systems, we introduce a notion of flexibility of distributed system and propose a design model of flexible distributed systems called agent-based distributed information processing system (ADIPS) and its design support environment called ADIPS Framework, in this paper. Various agents can be designed and implemented using knowledge embedded in respective agents together with the agentification mechanisms and the agent execution mechanisms of the ADIPS Framework. Furthermore, several examples of applications based on the ADIPS

Framework are explained to demonstrate the capabilities of the ADIPS Framework. At present, many problems remain as our future work, e.g., a ease-of-use knowledge representation and manipulation mechanisms for intelligent agent, the design support functions, an effective design method of agent-based system, and so on. However, we confirm that the ADIPS Framework is useful to make a conventional distributed system flexible in order to deal with the changes of both the users' requirements and the system environment through the experiment of prototypical applications.

## References

1. Campbell, A., Coulson, G. and Hutchison, D., "A Quality of Service Architecture," ACM SIGCOM, Computer Communication Review 24(2), pp.6-27, 1994.
2. Feldhoffer, M., "Model for Flexible Configuration of Application-oriented Communication Services," Comp. Commun, 18(2), pp.69-78, 1995.
3. Fujita, S., Sugawara, K., Kinoshita, T. and Shiratori, N., "Agent-based Architecture of Distributed Information Processing Systems", Trans. IPSJ 37(5), pp. pp.840-852, 1996
4. Fujita S., Sugawara K., Kinoshita T., Shiratori N., "An Approach to Developing Human-Agent Symbiotic Space", Proc. Second Joint Conf. on Knowledge-based Software Engineering, pp.11-18, JSAI&RAS, 1996.
5. Kinoshita, T., Sugawara, K., Shiratori, N., "Agent-based Framework for Developing Distributed Systems," Proc. CIKM'95 Workshop on Intelligent Information Agent, ACM-SIGART, 1995.
6. Kitagata G., Sekiba J., Suganuma T., Kinoshita T., Shiratori N., "Proposal and Design of Asynchronous Communication using Agents", Tech. Rep. IEICE IN98-32, pp.63-70, 1998.
7. Magedanz T., et al, Intelligent Agents: An Emerging Technology for Next Generation Telecommunications?, Proc. INFOCOM'96, 1996.
8. Nahrstedt, K. and Smith, J.M., "The QoS Broker," IEEE Multimedia, 2(1), pp.53-67, 1995.
9. Shiratori, N., Sugawara, K., Kinoshita, T., Chakraborty, G., "Flexible Networks: Basic concepts and Architecture," IEICE Trans. Comm. E77-B(11), pp.1287-1294, 1994.
10. Sugawara, K., Suganuma, T., Chakraborty, G., Moser, M., Kinoshita, T. and Shiratori, N., "Agent-oriented Architecture for Flexible Networks," Proc. the Second International Symposium on Autonomous Decentralized Systems, pp.135-141, 1995.
11. Suganuma T., Fujita S., Sugawara K., Kinoshita T., Shiratori N., "Flexible Videoconference System based on Muliagent-based Architecture", Trans. IPSJ 38(6), pp.1214-1224, 1997.
12. Suganuma T., Kinoshita T., Sugawara K., Shiratori N., "Flexible Videoconference System based on ADIPS Framework", Proc. Third Int. Conf. on Practical Application of Intelligent Agent and Multi-agent Technology (PAAM98), pp. 83-98, 1998.
13. Saga T., Sugawara K., Kinoshita T., Shiratori N., "An Approach to Developing CyberOffices based on Multiagent System", Tech. Rep. IEICE AI96-11, pp.23-30, 1996.
14. Turletti, T. and Huitema, C., "Videoconferencing on the Internet," IEEE/ACM Trans. on Networking 4(3), pp.340-351, 1996
15. Vogel, A., Kerherve, B., Bochmann, G. and Gecsei, J., "Distributed Multimedia and QoS: A Survey," IEEE Multimedia 2(2), pp.10-19, 1995.