Web Intelligence and Agent Systems: An International Journal 9 (2011) 161–178
DOI 10.3233/WIA-2011-0213
IOS Press

Configuration and control design model for an agent based Flexible Distributed System

Akiko Takahashi [a,*] and Tetsuo Kinoshita [b]

[a] *Sendai National College of Technology, 4-16-1 Ayashi-Chuo, Aoba-ku, Sendai 989-3128, Japan*
E-mail: akiko@sendai-nct.ac.jp

[b] *Cyberscience Center, Tohoku University, 2-1-1 Katahira, Aoba-ku, Sendai 980-8577, Japan*
E-mail: kino@riec.tohoku.ac.jp

Abstract. Realizing application systems that match people's expectations and providing adequate network services for various users under various network and platform environments are difficult challenges. To overcome these problems, we have studied application systems based on multiagent systems and design models of multiagent systems. We propose a method to observe and control the behavioral characteristics of multiagent systems to support the design, development, and operation of such systems. More specifically, we propose a Flexible Distributed System and a Behavioral Characteristics Model, and we apply these two models to the design of an actual multimedia communication system and evaluate the effectiveness of the proposed scheme.

Keywords: Multiagent system, Flexible Distributed System, Behavioral Characteristics Model, multimedia communication system

1. Introduction

As network services have spread, various people, from children to the elderly, routinely use and even rely upon network services. Consequently, it is necessary for applications to be easier to use and closer to people's expectations. Furthermore, with the spread of ubiquitous environments, it is necessary to provide adequate network services for various users on various networks and platform environments, including environments in which resources are restricted. To solve these problems, applications based on multiagent systems have been proposed.

In a multiagent system, the behavior of the overall system is extremely context-sensitive and nondeterministic because the behavior of each agent is decided dynamically by using the embedded knowledge of individual agents. Moreover, multiagent systems are distributed systems that compose an organization by the agent group that operates autonomously and behaves as an entire system through the cooperation of agents. Consequently, it is difficult to define the functions of the system and eliminate undesirable actions in the initial design phase of the system. This is a significant obstacle in the construction of usable multiagent systems. To overcome this obstacle, it is necessary to solve the following problems.

(P1) It is difficult to manage and update the agent's knowledge.
It is difficult to manage and update the knowledge of an individual agent systematically because the relations between an individual agent's operation specification and actual control are indefinite.

(P2) It is difficult to observe and control the overall system.
In some situations, control of the system becomes an infinite loop with insufficient control because each agent operates based on an individual specification and is controlled experientially.

In the design models of network service systems in previous studies, a semantic model for specifying

*Corresponding author.

and reasoning the required components for open distributed systems has been proposed [14]. A quantitative performance model has also been reported to predict the performance of a component-based server-side application in the design phase [22]. In addition, models of multiagent systems have been proposed, such as an organizational model that defines the knowledge of the system's organizational structure and capabilities to enable dynamic system reorganization, and a mathematical model prescribing that the agents act by Markov Chains to describe a system's emergent behavior as a Markov Decision Process [11,16]. However, the above issues remain as obstacles to realizing the flexible QoS control functions for some network service systems that are designed and implemented as multiagent systems.

The multiagent system requires a self-organizing model and self-control model [6,13]. Though self-organizing models have been proposed [4,9,21], these specialize in composing the agent organization. Additionally, as self-control models [5,10,12,20] are provided, the candidates are the control scheme or trust assurance of the overall agent system. However, to realize an adequate multiagent system, it is necessary that the multiagent's design model considers both the self-organizing model and the self-control model. Consequently, the models can realize a manageable, systematic and reliable multiagent system.

In this paper we propose a method to observe and control the behavioral characteristics of multiagent systems to support the design, development and operation of multiagent systems. In particular, to overcome the problems described above, we propose two models, as follows:

(S1) Flexible Distributed System (FDS).
This model is intended to support the construction of multiagent systems and respective agents in a distributed system. We can design functions of FDS systematically to manage information concerning not only the application functions and their QoS, but also the platform and network functions and their QoS based on the model.

(S2) Behavioral Characteristics Model (BCM) of a multiagent system based on FDS.
This is a support model to observe the service provision and situation of an agent, and to determine a necessary control and an actual control while providing service through the multiagent system. By setting the control criteria of BCM, it is possible to realize situation-oriented control of FDS.

By using FDS, a multiagent system can be designed and constructed systematically. Furthermore, by using BCM, a multiagent system with flexible QoS control capability can deal with changes in the system's operational situations and thereby maintain the required QoS as well as its behavioral characteristics.

In this study, to evaluate the effectiveness of the proposed models, we apply the models to the design of an actual multimedia communication system. We confirm that stable multimedia communication services can be provided by a system that is designed and controlled based on the proposed models. We also compare these results with results obtained using a conventional multimedia communication system. Subsequently, we evaluate the effectiveness of FDS and BCM.

The remainder of this paper is organized as follows. We propose FDS in Section 2 and then propose BCM in Section 3. In Section 4, a multimedia communication system is designed and implemented based on FDS and BCM. Section 5 presents the experimental results of the multimedia communication system to show the effect of the proposed scheme. Finally, we conclude the paper in Section 6.

2. Construction model of multiagent system

2.1. Outline of the Flexible Distributed System (FDS)

In this section, we propose the Flexible Distributed System (FDS) as a construction model of a multiagent system.

FDS autonomously configures a service that satisfies user requirements and functions during run time to provide the service to the users. During this service provision, the model activates processing to reduce undesirable influences on the service when changes occur inside and outside the system. According to these characteristics, the system can maintain service automatically.

The FDS function is realized by integrating the knowledge-processing abilities of each agent according to agent-based computing technology; that is, each agent manages its function and situation. The flexibility to address various requirements is attained through the cooperation of multiple agents. FDS function can handle the changes in system resources at the overall system level.

2.2. Flexible Distributed System

In this section, we represent FDS in a syntactic manner. FDS consists of requirements, available services and possible services.

Definition 1. A Flexible Distributed System, denoted as FDS, is designed based on an R, an AS and an S. The FDS can be described in the following expression:

$$FDS = \langle R, AS, S \rangle,$$

where the R is a set of user requirements and system requirements, the AS is a set of available services for the user and the system and the S is a set of possible services that can be realized in the system.

Detailed elements of an FDS are shown below. An R contains $r(i)$ $(i = 1, 2, \ldots, n)$ and is classified into an R_U and an R_S, as follows:

$$R = R\_U \cup R\_S,$$

where the R_U is a set of user requirements and the R_S is a set of system requirements. The R_U contains $r\_u(i)$ $(i = 1, 2, \ldots, n)$, and an $r\_u(i)$ is determined task Ts which is a set of user tasks $t(i)$ $(i = 1, 2, \ldots, n)$ and the quality of a task QoT which is a set of functions to determine the quality of a task $qot(i)$ $(i = 1, 2, \ldots, n)$, as follows:

$$\begin{aligned} R\_U = \{ & r\_u(i) \mid r\_u(i) = \langle t(i), qot(i) \rangle, \\ & t(i) \in Ts, \\ & qot(i) \in QoT, \\ & i = 1, 2, \ldots, n\}. \end{aligned}$$

The R_S is classified into an R_S_Ap, an R_S_Pt and an R_S_Nt, and described in the following expression:

$$R\_S = R\_S\_Ap \cup R\_S\_Pt \cup R\_S\_Nt,$$

where the R_S_Ap is a set of requirements of application, the R_S_Pt is a set of requirements of platform and the R_S_Nt is a set of requirements of network. The R_S_Ap contains $r\_s\_ap(i)$ $(i = 1, 2, \ldots, n)$, and an $r\_s\_ap(i)$ is the requirement description of an application function $ap\_f(i)$ that is defined by a required application function $req\_ap\_f(i)$, a required quality of application service $req\_ap\_f\_q(i)$, an R_S_Pt and an R_S_Nt, as follows:

$$\begin{aligned} R\_S\_Ap = \{ & r\_s\_ap(i) \mid r\_s\_ap(i) \\ & = \langle req\_ap\_f(i), req\_ap\_f\_q(i), \\ & R\_S\_Pt, R\_S\_Nt \rangle, \\ & i = 1, 2, \ldots, n\}. \end{aligned}$$

Similarly, the R_S_Pt contains $r\_s\_pt(i)$ $(i = 1, 2, \ldots, n)$, and an $r\_s\_pt(i)$ is the requirement description of a platform function $pt\_f(i)$ that is defined by a required platform function $req\_pt\_f(i)$ and a required quality of platform service $req\_pt\_f\_q(i)$. The R_S_Pt is represented as follows:

$$\begin{aligned} R\_S\_Pt = \{ & r\_s\_pt(i) \mid r\_s\_pt(i) \\ & = \langle req\_pt\_f(i), req\_pt\_f\_q(i) \rangle, \\ & i = 1, 2, \ldots, n\}. \end{aligned}$$

The R_S_Nt contains $r\_s\_nt(i)$ $(i = 1, 2, \ldots, n)$, and an $r\_s\_nt(i)$ is the requirement description of a network function $nt\_f(i)$ that is defined by a required network function $req\_nt\_f(i)$ and a required quality of network service $req\_nt\_f\_q(i)$. The R_S_Nt is represented as follows:

$$\begin{aligned} R\_S\_Nt = \{ & r\_s\_nt(i) \mid r\_s\_nt(i) \\ & = \langle req\_nt\_f(i), req\_nt\_f\_q(i) \rangle, \\ & i = 1, 2, \ldots, n\}. \end{aligned}$$

An AS contains $as(i)$ $(i = 1, 2, \ldots, n)$, and an $as(i)$ is determined and selected from an S based on an $r(i)$. The AS is given as a result of a design process D in the following expression:

$$\begin{aligned} AS = \{ & as(i) \mid as(i) \leftarrow D(r(i), S), \\ & i = 1, 2, \ldots, n\}. \end{aligned}$$

The AS is classified into three kinds of services: a set of available application services AS_Ap, a set of available platform services AS_Pt and a set of available network services AS_Nt, as follows:

$$AS = AS\_Ap \cup AS\_Pt \cup AS\_Nt.$$

Definition 2. Because an AS consists of an AS_Ap, an AS_Pt and an AS_Nt, a design process D also

consists of the design process of an application ApD, the design process of a platform PtD and the design process of a network NtD:

$$D = ApD \cup PtD \cup NtD.$$

A possible application service S_Ap contains $s\_ap(j)$ $(j = 1, 2, \ldots, m)$, and an $s\_ap(j)$ is specified by an $ap\_f(j)$, an $ap\_f\_q(j)$, an N_S_Pt and an N_S_Nt, where the $ap\_f(j)$ is application function, the $ap\_f\_q(j)$ is the quality of the $ap\_f(j)$, the N_S_Pt is a set of platform services that is necessary for the $ap\_f(j)$ and the N_S_Nt is a set of network services that is necessary for the $ap\_f(j)$. The S_Ap can be described in the following expression:

$$\begin{aligned} S\_Ap = \{&s\_ap(j) \mid s\_ap(j) \\ &= \langle ap\_f(j), ap\_f\_q(j), N\_S\_Pt, \\ &N\_S\_Nt\rangle,\ j = 1, 2, \ldots, m, \\ &N\_S\_Pt \subseteq S\_Pt, \\ &N\_S\_Nt \subseteq S\_Nt\}. \end{aligned}$$

Given an $r\_s\_ap(i) \in R\_S\_Ap$, an available application service $av\_s\_ap(i)$ is selected from the S_Ap, $AS\_Ap \subseteq S\_Ap$ in an ApD:

$$av\_s\_ap(i) \leftarrow ApD(r\_s\_ap(i), S\_Ap).$$

As a result, an AS_Ap is derived in the ApD, denoted as following expression:

$$\begin{aligned} AS\_Ap = \{&av\_s\_ap(i) \mid av\_s\_ap(i) \\ &= \langle ap\_f(j), ap\_f\_q(j), \\ &AS\_Pt, AS\_Nt\rangle, \\ &s\_ap(j) \in S\_Ap, \\ &req\_ap\_f(i) = ap\_f(j), \\ &req\_ap\_f\_q(i) \leq ap\_f\_q(j), \\ &R\_S\_Pt \subseteq N\_S\_Pt \subseteq AS\_Pt, \\ &R\_S\_Nt \subseteq N\_S\_Nt \subseteq AS\_Nt, \\ &i = 1, 2, \ldots, n\}. \end{aligned}$$

Similarly, a possible platform service S_Pt contains $s\_pt(k)$ $(k = 1, 2, \ldots, r)$, and an $s\_pt(k)$ is specified by a $pt\_f(k)$, a $pt\_f\_q(k)$ and an N_Rs_Pt, where the $pt\_f(k)$ is a platform function, the $pt\_f\_q(k)$ is the quality of the $pt\_f(k)$ and the N_Rs_Pt is a set of the necessary resources of the $pt\_f(k)$ and is a part of the platform resources Rs_Pt. The S_Pt can be described in the following expression:

$$\begin{aligned} S\_Pt = \{&s\_pt(k) \mid s\_pt(k) \\ &= \langle pt\_f(k), pt\_f\_q(k), N\_Rs\_Pt\rangle, \\ &N\_Rs\_Pt \subseteq Rs\_Pt, \\ &k = 1, 2, \ldots, r\}. \end{aligned}$$

Given an $r\_s\_pt(i) \in R\_S\_Pt$, an available platform service $av\_s\_pt(i)$ is selected from the S_Pt, $AS\_Pt \subseteq S\_Pt$ in a PtD:

$$av\_s\_pt(i) \leftarrow PtD(r\_s\_pt(i), S\_Np).$$

As a result, in the PtD, if an A_Rs_Pt is a set of available resources of a $pt\_f(i)$, the following AS_Pt is derived:

$$\begin{aligned} AS\_Pt = \{&av\_s\_pt(i) \mid av\_s\_pt(i) \\ &= \langle pt\_f(k), pt\_f\_q(k), A\_Rs\_Pt\rangle, \\ &s\_pt(k) \in S\_Pt, \\ &pt\_f(k) = req\_pt\_f(i), \\ &req\_pt\_f\_q(i) \leq pt\_f\_q(k), \\ &N\_Rs\_Pt \subseteq A\_Rs\_Pt\}. \end{aligned}$$

A possible network service S_Nt contains $s\_nt(l)$ $(l = 1, 2, \ldots, t)$, and an $s\_nt(l)$ is specified by an $nt\_f(l)$, an $nt\_f\_q(l)$ and an N_Rs_Nt, where the $nt\_f(l)$ is a network function, the $nt\_f\_q(l)$ is a quality of the $nt\_f(l)$ and the N_Rs_Nt is a set of the necessary resources of the $nt\_f(l)$ and is a part of the network resources Rs_Nt. The S_Nt can be described in the following expression:

$$\begin{aligned} S\_Nt = \{&s\_nt(l) \mid s\_nt(l) \\ &= \langle nt\_f(l), nt\_f\_q(l), N\_Rs\_Nt\rangle, \\ &N\_Rs\_Nt \subseteq Rs\_Nt, \\ &l = 1, 2, \ldots, t\}. \end{aligned}$$

Given an $r\_s\_nt(i) \in R\_S\_Nt$, an available network service $av\_s\_nt(i)$ is selected from the S_Nt, $AS\_Nt \subseteq S\_Nt$ in a NtD:

$$av\_s\_nt(i) \leftarrow NtD(r\_s\_nt(i), S\_Nt).$$

As a result, in the NtD, if an A_Rs_Nt is a set of available resources of the $nt\_f(i)$, the following AS_Nt is derived:

$$\begin{aligned} AS\_Nt = \{&av\_s\_nt(i) \mid av\_s\_nt(i) \\ &= \langle nt\_f(l), nt\_f\_q(l), A\_Rs\_Nt\rangle, \\ &s\_nt(l) \in S\_Nt, \\ &nt\_f(l) = req\_nt\_f(i), \\ &req\_nt\_f\_q(i) \le nt\_f\_q(l), \\ &N\_Rs\_Nt \subseteq A\_Rs\_Nt\}. \end{aligned}$$

Next, we consider a variable QoT. Two kinds of QoT are definable: a required QoT and a realized QoT.

Definition 3. The required quality of a task Req_QoT is defined by required qualities of an R_S_Ap, an R_S_Pt, and an R_S_Nt, i.e., R_Ap_QoS, R_Pt_QoS and R_Nt_QoS, denoted as follows:

$$\begin{aligned} Req\_QoT = G(&R\_Ap\_QoS, R\_Pt\_QoS, \\ &R\_Nt\_QoS). \end{aligned}$$

For example, if α_1, α_2 and α_3 are priorities and φ is a coefficient constant, using a linear combination operation, a Req_QoT is calculated in the following:

$$\begin{aligned} Req\_QoT &= \varphi(R\_Ap\_QoS + R\_Pt\_QoS \\ &\quad + R\_Nt\_QoS) \\ &= \varphi\Bigg(\alpha_1 \sum_i req\_ap\_f\_q(i) \\ &\quad + \alpha_2 \sum_j req\_pt\_f\_q(j) \\ &\quad + \alpha_3 \sum_k req\_nt\_f\_q(k)\Bigg). \end{aligned}$$

Definition 4. Because a required task is realized by available functions of an AS_Ap, an AS_Pt and an AS_Nt, the realized quality of a task Rlz_QoT is defined by qualities of the AS_Ap, the AS_Pt and the AS_Nt, i.e., A_Ap_QoS, A_Pt_QoS and A_Nt_QoS, as follows:

$$\begin{aligned} Rlz\_QoT = G(&A\_Ap\_QoS, A\_Pt\_QoS, \\ &A\_Nt\_QoS). \end{aligned}$$

If β_1, β_2 and β_3 are priorities and μ is a coefficient constant, using a linear combination operation as an example, an Rlz_QoT is calculated in the following:

$$\begin{aligned} Rlz\_QoT &= \mu(A\_Ap\_QoS + A\_Pt\_QoS \\ &\quad + A\_Nt\_QoS) \\ &= \mu\Bigg(\beta_1 \sum_i ap\_f\_q(i) \\ &\quad + \beta_2 \sum_j pt\_f\_q(j) \\ &\quad + \beta_3 \sum_k nt\_f\_q(k)\Bigg). \end{aligned}$$

Definition 5. The margin of QoT M_QoT is defined as the difference of a Req_QoT and a Rlz_QoT, as follows:

$$M\_QoT = Rlz\_QoT - Req\_QoT.$$

Because the M_QoT has the following characteristics:

$$\begin{aligned} &\text{if } Rlz\_QoT \ge Req\_QoT, \text{ then } M\_QoT \ge 0, \\ &\qquad\qquad\qquad\qquad \text{else } M\_QoT < 0, \end{aligned}$$

the M_QoT can be used as a measure of the system's characteristics:

if $M\_QoT \ge 0$,
  then the Req_QoT is maintained
    and the user requirement is satisfied,
  else the Rlz_QoT is degraded
    and the user requirement is not satisfied.

During $M\_QoT < 0$ when using a system, various changes occur in the system. Operations to remove or recover the degradation, such as tuning the system parameters or changing the system configuration, should be executed to maintain the user requirements.

3. Control model of multiagent system

3.1. Outline of the Behavioral Characteristics Model (BCM)

In this section, we propose the Behavioral Characteristics Model (BCM) as the control model of the multiagent system.

BCM is a control model of multiagent systems and is defined based on the change in QoT and its duration time. When a change in QoT is observed, BCM exerts control according to the characteristics of the change. These characteristics are decided by specifying the element that caused the change. BCM performs sufficient control so that the control recovers from the change directly.

3.2. Observation of service changes in FDS

Changes in an FDS are classified into the following three patterns according to the causes of the changes in an AS: changes caused by a change in the AS itself, a change in an S, and a change in an R. The changes in QoT are affected by applications, platforms, and networks. Changes then occur in the AS in the overall FDS. Next we define the changes in the FDS.

Definition 6. A change in an FDS occurs because of a ΔAS. The ΔAS is a change in an AS and a ΔFDS is a change in the FDS. The change of the FDS can be described in the following expression:

$$FDS \Longrightarrow FDS + \Delta FDS,$$
$$FDS + \Delta FDS = \langle R, AS + \Delta AS, S\rangle.$$

Definition 7. A ΔAS is classified into a ΔAS_Ap, a ΔAS_Pt and a ΔAS_Nt, where the ΔAS_Ap is a change in an AS_Ap, the ΔAS_Pt is a change in an AS_Pt and the ΔAS_Nt is a change in an AS_Nt. If an FDS changes to an $FDS + \Delta FDS$, an available application QoS A_Ap_QoS is also changed to an $AS\_Ap + \Delta AS\_Ap$, as follows:

$$AS\_Ap \Longrightarrow AS\_Ap + \Delta AS\_Ap,$$
$$A\_Ap\_QoS \Longrightarrow A\_Ap\_QoS + \Delta A\_Ap\_QoS,$$

an available platform QoS A_Pt_QoS is changed to an $AS\_Pt + \Delta AS\_Pt$, as follows:

$$AS\_Pt \Longrightarrow AS\_Pt + \Delta AS\_Pt,$$
$$A\_Pt\_QoS \Longrightarrow A\_Pt\_QoS + \Delta A\_Pt\_QoS,$$

an available network QoS A_Nt_QoS is changed to an $AS\_Nt + \Delta AS\_Nt$, as follows:

$$AS\_Nt \Longrightarrow AS\_Nt + \Delta AS\_Nt,$$
$$A\_Nt\_QoS \Longrightarrow A\_Nt\_QoS + \Delta A\_Nt\_QoS.$$

For example, let us assume

$$av\_s\_ap(i) = \langle ap\_f(i), ap\_f\_q(i), AS\_Pt, AS\_Nt\rangle$$

is changed to

$$av\_s\_ap'(i) = \langle ap\_f'(i), ap\_f\_q'(i), AS\_Pt, AS\_Nt\rangle,$$
$$ap\_f'(i) \neq ap\_f(i), ap\_f\_q'(i) \neq ap\_f\_q(i).$$

An A_Ap_QoS is also changed to an A_Ap_QoS' (tentative and permanent changes in an AS_Ap). The sum of the differences of the changed qualities ΔA_Ap_QoS is given as follows:

$$\Delta A\_Ap\_QoS = \sum_i (ap\_f\_q'(i) - ap\_f\_q(i)).$$

Consequently, if $\Delta A\_Ap\_QoS < 0$, then negative changes occurred in the system (tentative and permanent changes in both an AS_Pt and an AS_Nt). Similarly, a ΔA_Pt_QoS is given as follows:

$$\Delta A\_Pt\_QoS = \sum_j (pt\_f\_q'(j) - pt\_f\_q(j)),$$

if $\Delta A\_Pt\_QoS < 0$, then negative changes occurred in the platform. A ΔA_Nt_QoS is also given as follows:

$$\Delta A\_Nt\_QoS = \sum_k (nt\_f\_q'(k) - nt\_f\_q(k)),$$

if $\Delta A\_Nt\_QoS < 0$, then negative changes occurred in the network.

Definition 8. The difference of change in an FDS ΔM_QoT is the sum of a ΔA_Ap_QoS, a ΔA_Pt_QoS and a ΔA_Nt_QoS, denoted as following expression:

$$\Delta M\_QoT = \Delta A\_Ap\_QoS + \Delta A\_Pt\_QoS + \Delta A\_Nt\_QoS.$$

The ratio of a ΔM_QoT, i.e. $rat\_\Delta M\_QoT$ is defined based on a ΔM_QoT and a Req_QoT, as follows:

$$rat\_\Delta M\_QoT = \sum \varphi |\Delta M\_QoT| / Req\_QoT.$$

3.3. Detection of service changes in FDS

In this section, we explain several characteristics of changes in an FDS.

Definition 9. A change in QoT is classified into change in QoT, fluctuation of QoT and degradation of QoT, as follows:

if $\Delta M\_QoT \neq 0$ and $rat\_\Delta M\_QoT > \psi$,
then a change in QoT $Ch(t)$ is detected at time t,
if $rat\_\Delta M\_QoT \leq \psi$,
then a fluctuation of QoT is detected,
if $Ch(t)$ and $\Delta M\_QoT < 0$ and
$rat\_\Delta M\_QoT > \varepsilon$,
then a degradation of QoT $Dg(\varepsilon, t)$ is detected at time t.

Here, ψ and ε are threshold values of the ΔM_QoT and the $rat\_\Delta M\_QoT$ respectively.

An FDS does not need to operate when a fluctuation of QoT is observed because the fluctuation of QoT does not influence the external FDS. In contrast, when a change in QoT is observed, especially when a degradation of QoT is detected, the FDS analyzes the situation and performs some operations on the system. The degradation of QoT is defined as a temporary change in QoT and a permanent change in QoT.

Definition 10. Let ξ is the duration of change in QoT:

if $\forall t', t \in [t_1, t_2), 0 \leq t_1 < t_2$, $Dg(\varepsilon, t)$,
then $\xi = t_2 - t_1$.

Here, γ is a threshold of a temporary change in QoT and $\gamma 0$ is a threshold of a fluctuation of QoT.

If $0 \leq \xi \leq \gamma 0$, then an FDS is considered to be a fluctuation of QoT.

A degradations of QoT can be classified into a temporary change in QoT and a permanent change in QoT. The details of degradations can be described in the following expression:

if $\forall \xi, \gamma 0 < \xi \leq \gamma$ and $Dg(t + \xi)$,
then a temporary change in QoT $TDg(t + \xi)$ is detected,
else $\forall \xi, \gamma < \xi$ and $Dg(t + \xi)$,
then a permanent change in QoT $PDg(t + \xi)$ is detected.

3.4. Flexibility of FDS

In an FDS, when a degradation of QoT is observed, a counter operation δFDS is activated to process the change in the FDS to cancel a ΔFDS.

Definition 11. A δAS for a degradation of QoT consists of TN_i $(i = 1, 2, \ldots, n)$ and RD_i $(i = 1, 2, \ldots, n)$, where the δAS is counter operation, and a TN_i is tuning operation and a RD_i is redesign operation. Suppose that a τ_i is the duration of counter operation, a $TN_i(t, R, AS + \Delta AS, S, \tau_i)$ provides as the δAS against a ΔAS at time t. If the QoT isn't recovered by the TN_i, an $RD_i(t, R, AS + \Delta AS, S, \tau_i)$ alters the AS of the FDS to the other AS' of an FDS' as the δAS.

When a QoT is recovered by these operations, the FDS becomes a new FDS' i.e., $FDS + \Delta FDS + \delta FDS$, as follows:

$$FDS + \Delta FDS + \delta FDS = \langle R, AS + \Delta AS + \delta AS, S \rangle.$$

Using these operations, the flexibilities of a temporary change in QoT and a permanent change in QoT are defined.

Definition 12. An FDS which is flexible with respect to a temporary change in QoT can be described in the following expression:

if $TDg(\varepsilon, t + \gamma 0)$, $t_s = t + \gamma 0 \leq T$ and
$\xi = \gamma 0 + \tau \leq \gamma$, where t is a time point which starts a degradation of QoT, the T is the threshold of a TDg and the τ is the duration of the counter operations,
then a tuning operation is activated to recover the QoT for the FDS:
$\delta AS \leftarrow TN_i(t_s, R, AS + \Delta AS, S, \tau)$,
as a result, $\Delta M\_QoT(t + \xi) \geq 0$,
$rat\_\Delta M\_QoT(t + \xi) \leq \varepsilon$,
$\forall t', t' > t + \xi$, $\Delta M\_QoT(t') \geq 0$,
$rat\_\Delta M\_QoT(t') \leq \varepsilon$,
then the FDS is flexible with respect to the temporary changes in QoT, and an $FDS(t)$ is change to an $FDS'(t')$:
$FDS(t) \Longrightarrow FDS'(t')$.

Figure 1 shows that if an FDS can resolve a temporary change in QoT using the procedure described

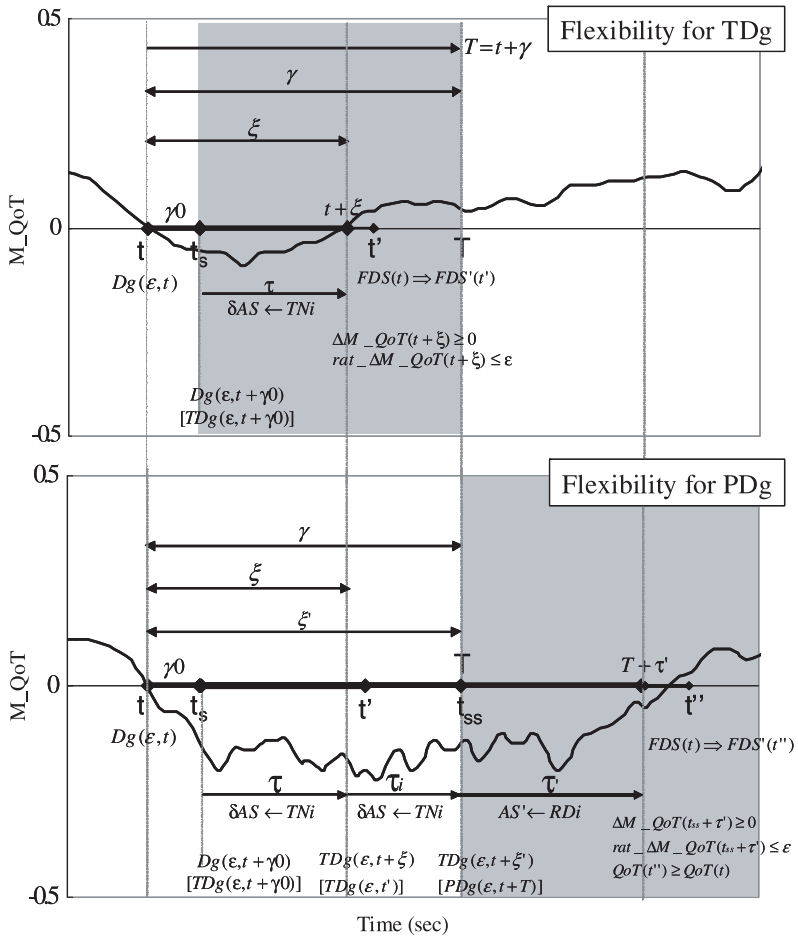

Fig. 1. Flexibility for the temporary change in QoT and the permanent change in QoT.

above, then the FDS is regarded as flexible with respect to the temporary change in QoT. Otherwise, as portrayed in Fig. 1, operations for a permanent change in QoT is required.

Definition 13. An FDS isn't flexible for a temporary change in QoT, the FDS provides a operation for a permanent change in QoT, as follows:

if $TDg(\varepsilon, t+\gamma 0)$, $ts = t+\gamma 0 \leq T$,
$\xi = \gamma 0 + \tau \leq \gamma$,
then a tuning operation is activated to recover the QoT for FDS is observed, as follows:
$\delta AS \leftarrow TN_i(t_s, R, AS + \Delta AS, S, \tau)$,
as a result, $TDg(t+\xi)$, $\Delta M\_QoT(t+\xi) < 0$ and $rat\_\Delta M\_QoT(t+\xi) > \varepsilon$,
then a temporary change in QoT continues at $t+\xi$,
if $\forall t', t+\xi < t' \leq T$, $TDg(\zeta, t')$, $\xi' = \xi + \sum \tau_i$,
where τ_i is the duration of the counter operations,
then a tuning operation is activates to recover the QoT for FDS again:
$\delta AS \leftarrow TN_i(t', R, AS + \Delta AS, S, \tau_i)$,
as a result, $TDg(\varepsilon, t+\xi')$, $\xi' = \xi + \sum \tau_i$ is observed,
then $PDg(t+T)$ is detected at $t+T$ and a RD_i operation is applied.

A premanent change in QoT is detected, a redesign operation is activated to recover the QoT for the FDS:

if $AS' \leftarrow RD_i(t_{ss}, R+\Delta R, AS+\Delta AS, S, \Delta\tau')$,
then $t_{ss} = t+T$, $\Delta M\_QoT(t_{ss}+\tau') \geq 0$,
$rat\_\Delta M\_QoT(t_{ss}+\tau') \leq \varepsilon$, $AS \subseteq AS$'
and $\forall t'', t'' \geq t_{ss}+\tau'$, $QoT(t'') \geq QoT(t)$
are observed,
then the FDS is flexible with respect to the
permanent changes in QoT, and an
$FDS(t)$ is change to an $FDS'(t'')$:
$FDS(t) \Longrightarrow FDS'(t'')(t' \geq t_{ss}+\tau')$.

If an FDS can resolve a permanent change in QoT by using the procedure described above, then the FDS is regarded as flexible with respect to the permanent change in QoT. Otherwise, FDS designers must redesign the FDS to satisfy the requirements.

Finally, the notions of a QoS-$stable\ FDS$ and a $pseudo$-QoT-$stable\ FDS$ are definable with respect to an FDS behavior.

Definition 14. A $QoT\_stable\ FDS$ exists if a changed FDS can be returned to the original FDS. The $QoT\_stable\ FDS$ can be described in the following expression:

$$
\begin{aligned}
&FDS(t) \Longrightarrow FDS'(t+\xi) \\
&\quad \Longrightarrow FDS(t+\xi+\rho+\nu), \\
&AS \Longrightarrow AS+\delta AS \Longrightarrow AS, \\
&Rlz\_QoS \Longrightarrow Rlz\_QoS+\Delta Rlz\_QoS \\
&\quad \Longrightarrow Rlz\_QoS.
\end{aligned}
$$

Therefore, FDS changes to the original FDS when the state of the system recovers. A $pseudo\_QoT\_stable\ FDS$ exists if the QoT of the changed FDS can be brought closer to the original QoT. The $pseudo\_QoT\_stable\ FDS$ can be described in the following expression:

$$
\begin{aligned}
&FDS(t) \Longrightarrow FDS'(t+\xi), \\
&AS \Longrightarrow AS+\delta AS \Longrightarrow AS', \\
&Rlz\_QoS \Longrightarrow Rlz\_QoS+\Delta Rlz\_QoS, \\
&\quad \Longrightarrow Rlz\_QoS'.
\end{aligned}
$$

The FDS provides similar services when the original FDS cannot be recovered.

4. Example of a multimedia communication system

4.1. Multimedia communication system based on FDS and BCM

This section presents an agent based multimedia communication system (AMCS) as an application of FDS and BCM [1]. AMCS consists of various multimedia communication service components that are realized as agents (multimedia processing agents). Moreover, the organization of these agents is configured dynamically at the run time of AMCS on the basis of the BCM of FDS, explained in Section 3.

By applying FDS and BCM to the design of AMCS, the detailed analyses and design of the operational situations of AMCS can be carried out, and the actual observation parameters of AMCS can be mapped into the elements of BCM.

Figure 2 depicts the architecture of a media processing agent implemented as an ADIPS/DASH agent [3,7,17–19]. The agent's knowledge, which is defined based on FDS, is embedded into each agent. For example, the agent's knowledge is designed and represented based on R_S_Ap, S_Ap, and AS_Ap. An example of the S_Ap description is also presented in Fig. 3. In Fig. 3, elements of the S_Ap are represented as "facts" in the form of objects, attributes, and values (OAV). In this OAV representation, an object name follows a left parenthesis "(", an attribute name follows a colon ":", and a value follows the respective attribute names. For the fact depicted in Fig. 3, the object name "s_ap" consists of four pairs of attributes and values; this fact represents the first element of S_Ap.

4.2. Agent behavior of AMCS

The organization of agents is constructed using the QoS-based CNP (Q-CNP) [1], which is designed using an extended cooperation protocol of the Contract Net Protocol (CNP) [15].

Figure 4 illustrates the AMCS framework. In AMCS, the Agent Repository, where agents are stored, and the Agent Workplace, where agents are actually working during run time, are located in a networked environment. The agents stored in the Agent Repository are designed based on FDS. Each agent is designed and implemented as an element of S_Ap, together with the knowledge of the respective multimedia communication services. When a user requirement (R_S_Ap) is given to a User Agent (UA), the suitable agents that

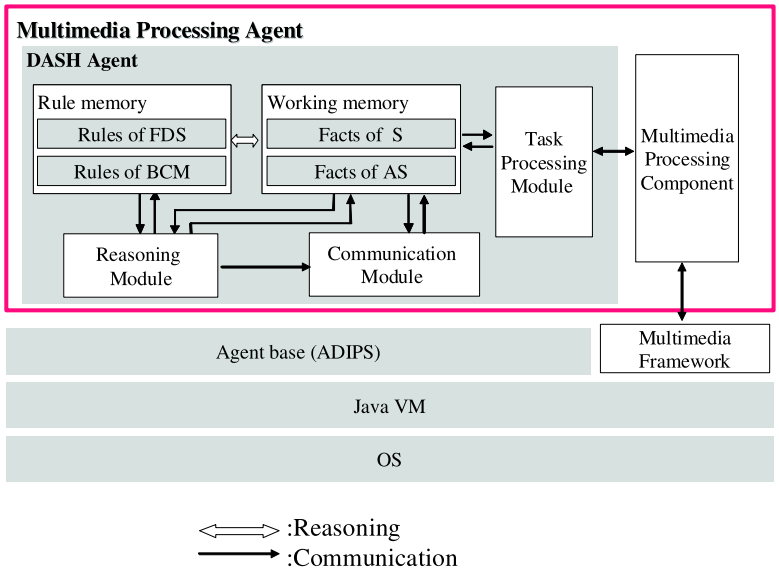

Fig. 2. Architecture of a media processing agent.

```
(s-ap :ap-f streaming-service-function
      :ap-f-q (f :fps 10 :quality 60 :size 160)
      :n-s-pt 15
      :n-s-nt 3000)
(s-ap :ap-f streaming-service-function
      :ap-f-q (f :fps 15 :quality 60 :size 160)
      :n-s-pt 20
      :n-s-nt 3000)
.....
(s-ap :ap-f streaming-service-function
      :ap-f-q (f :fps 15 :quality 60 :size 320)
      :n-s-pt 25
      :n-s-nt 4000)
.....
```

Fig. 3. Example of an S_Ap description.

satisfy R_S_Ap are selected and organized using the S_Ap stored in the Agent Repository by the Manager Agent (Manager) based on FDS. Then, the organization of agents is instantiated into the Agent Workplace as the AS_Ap of the AMCS. Figure 5 shows an example of the knowledge description for selecting the AS_Ap based on the S_Ap that is implemented as one rule of the model of Fig. 2.

The syntax of this rule is identical to that of OPS5 [2]. That is, it consists of a conditional part, which precedes "$--$>", and an action part, which follows "$--$>". An element with ":" is an attribute and an element with "?" is a variable. The rule is interpreted by the first-match strategy, as with OPS5. That is, the rule is interpreted by pattern-matching of the descriptions for req_s_ap and s_ap, which are given as facts.

According to this rule, if all nine patterns match, then the bound s_ap is selected and pushed to the list of candidates of possible services that fulfill the service requirement. Then, the instantiated agents start the service provision. During this service provision, the Manager operates the service using AS_Ap, AS_Pt, and AS_Nt, which are observed by the Manager, Platform Monitoring Agent (PtMA), and Network Monitoring Agent (NtMA), respectively.

4.3. Definition of QoT and the AMCS operations

This section provides a more detailed description of the definition of QoT and the operations of AMCS. The parameters that are used to evaluate the status of the services of AMCS are given as follows:

$$Req\_QoT = \Bigg(\alpha_1 \sum_i req\_ap\_f\_q(i) + \alpha_2 \sum_j req\_pt\_f\_q(j) + \alpha_3 \sum_k req\_nt\_f\_q(k) \Bigg) / (i + j + k),$$

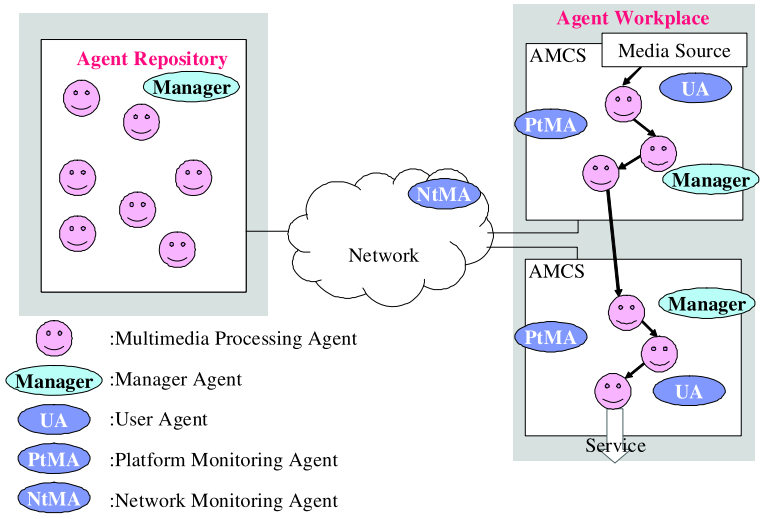

Fig. 4. Framework of the AMCS.

```
(rule select-as-ap-fact
    (req-s-ap :req-ap-f ?req-ap-f
              :req-ap-f-q (f :req-fps ?req-fps
                             :req-quality ?req-quality
                             :req-size ?req-size)
              :req-s-pt ?req-s-pt
              :req-s-nt ?req-s-nt)
    (s-ap :ap-f ?ap-f
          :ap-f-q (f :s-fps ?s-fps
                     :s-quality ?s-quality
                     :s-size ?s-size)
          :n-s-pt ?n-s-pt
          :n-s-nt ?n-s-nt)
    (as-ap-list ?as-ap-list-body)
    (== ?req-ap-f ?ap-f)
    (>= ?s-fps ?req-fps)
    (>= ?s-quality ?req-quality)
    (>= ?s-size ?req-size)
    (>= ?n-s-pt ?req-s-pt)
    (>=h ?n-s-nt ?req-s-nt)
-->
    (push ?s-ap ?as-ap-list-body)
  )
```

Fig. 5. Example of knowledge description to select the AS_Ap.

$$Rlz\_QoT = \Big(\beta_1 \sum_i ap\_f\_q(i) + \beta_2 \sum_j pt\_f\_q(j) + \beta_3 \sum_k nt\_f\_q(k)\Big)/(i+j+k).$$

According to the above definitions, the quality parameters of the multimedia communication services are mapped to the attributes of the qualities of the BCM components: the frame rate, encoding quality, and video size are ap_f, the CPU resource is pt_f, and the bandwidth is nt_f. The parameters α_1 to α_3 and β_1 to β_3 are determined depending on the QoS priorities, which are provided by the applications, platforms, and networks. For example, when a QoS of an application is prioritized, α_1 and β_1 are weighted more than α_2, α_3, β_2, and β_3.

After the services of AMCS are provided, ΔM_QoT and $rat\_\Delta M\_QoT$ are calculated according to the changes in QoT that occurred in the system, as follows:

$$\begin{aligned}\Delta M\_QoT = {} & \beta_1 \sum_i \Delta ap\_f\_q(i) + \beta_2 \sum_j \Delta pt\_f\_q(j) + \beta_3 \sum_k \Delta nt\_f\_q(k) \\ & - \alpha_1 \sum_i \Delta req\_ap\_f\_q(i) - \alpha_2 \sum_j \Delta req\_pt\_f\_q(j) - \alpha_3 \sum_k \Delta req\_nt\_f\_q(k),\end{aligned}$$

$$\begin{aligned}rat\_\Delta M\_QoT &= rat(\Delta M\_QoT, Req\_QoT) \\ &= |\Delta M\_QoT|/Req\_QoT.\end{aligned}$$

The counter operations are selected and executed against these changes when a temporary change in QoT or a permanent change in QoT is observed using ΔM_QoT and $rat\_\Delta M\_QoT$. The tuning operations of the agents' working conditions should be activated as a TN operation when a temporary change in QoT is observed. However, when a permanent change in QoT is observed, a reorganization operation of the agent organization is activated as an RD operation. From these operations, the undesirable operational situations of AMCS are recovered. Consequently, the flexible QoS control strategies in AMCS can be expressed and designed based on BCM. Moreover, using FDS, the QoS control knowledge can be designed and implemented systematically as the operational knowledge of the agents of AMCS.

5. Experiment and evaluation

5.1. Purpose of experiment

In this section we evaluate the proposed scheme using two experiments. The experimental results are demonstrated using an AMCS prototype.

Through those experiments, we evaluate how the AMCS recovers the degraded QoT at run time and we show that the AMCS appropriately handles the degradation of QoT in the provision of a live streaming service. Moreover, we observe the system's response to the temporary change in QoT, which is a basic and typical operation. From these results, we can confirm whether the proposed model is useful for analysis of the behavior of multimedia network middleware.

5.2. Environment of the experiment

Figure 6 depicts the configuration of the experimental system of experiment 1. A sender-side PC is connected to a 100 Mbps Ethernet LAN; a receiver-side PC is connected with an 11 Mbps link of IEEE802.11b via a wireless access network.

The initial experimental conditions are as follows:

$R_S_Ap = \{\langle$streaming-service-function,
fps: 15 (fps)
quality: 80 (%)
size: 320 (pixel)
sender-CPU: 60 (%)
receiver-CPU: 60 (%)
bandwidth: 5000 (kbps)$\rangle\}$

$R_S_Pt = \{\langle$CPU usage, sender-CPU: 60 (%)
receiver-CPU: 60 (%)$\rangle\}$

$R_S_Nt = \{\langle$bandwidth,
bandwidth: 5000 (kbps)$\rangle\}$

Thresholds: $\gamma 0 = 3.0$ s, $\gamma = 20$ s

All were given as fixed values.

Three systems were selected for experiment 1:

system 1 A multimedia communication system without QoS control functions,

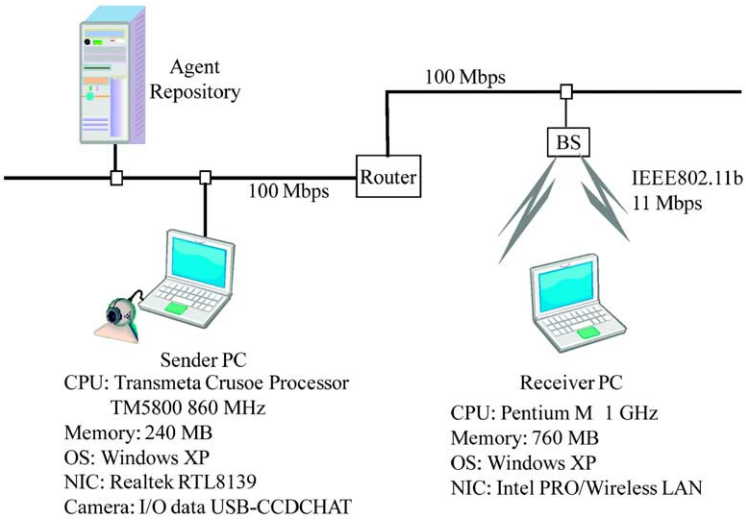

Fig. 6. Configuration of experimental system 1.

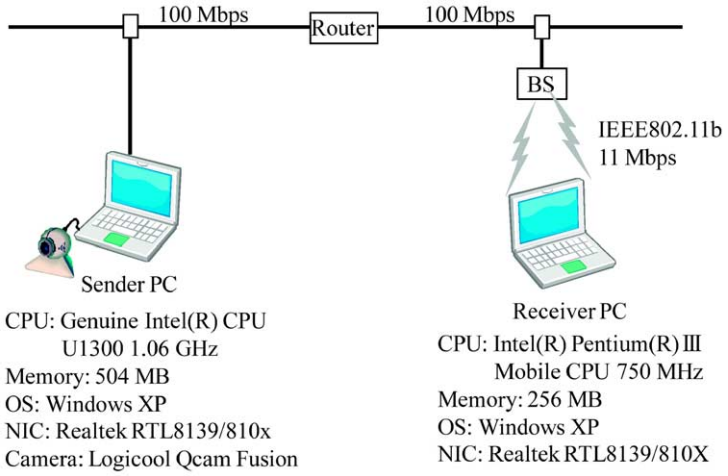

Fig. 7. Configuration of experimental system 2.

system 2 An AMCS with FDS, and
system 3 An AMCS with FDS and BCM.

Figure 7 depicts the configuration of the experimental system used for experiment 2. A sender-side PC is connected to a 100 Mbps Ethernet LAN; and a receiver-side PC is connected with an 11 Mbps link of IEEE802.11b via a wireless access network.

The initial experimental conditions are as follows:

$$
\begin{aligned}
R\_S\_Ap = \{\langle &\text{streaming-service-function,}\\
&\text{fps: 15 (fps)}\\
&\text{quality: 60 (\%)}\\
&\text{size: 280 (pixel)}\\
&\text{sender-CPU: 40 (\%)}\\
&\text{receiver-CPU: 50 (\%)}\\
&\text{bandwidth: 200 (kbps)}\rangle\}\\
R\_S\_Pt = \{\langle &\text{CPU usage, sender-CPU: 40 (\%)}\\
&\text{receiver-CPU: 50 (\%)}\rangle\}\\
R\_S\_Nt = \{\langle &\text{bandwidth, bandwidth: 200 (kbps)}\rangle\}
\end{aligned}
$$

All were given as fixed values.

Two systems were selected for experiment 2:

system 2 An AMCS with FDS, and
system 3 An AMCS with FDS and BCM.

In this experiment, we used the Java Media Framework (JMF) [8] as a basic multimedia communication system framework. Although JMF provides component-based multimedia processing capabilities, it has no QoS control functions by itself.

For system 1, we used JMF as it is and observed the service provided by JMF. In JMF, when a user requirement is issued, a multimedia communication system is constructed using media processing components, thereby realizing the required service. This configuration is determined using a "graph-building algorithm" based on JMF. This algorithm selects the smallest organization from the possible organizations configured by using a connectable component chain with consideration of the input and output specifications of the components. The selected configuration is performed with no QoS consideration. In system 2, we used an AMCS, which is designed based on FDS. In this case, BCM is not used. The system is constructed by adding the configuration function of components based on QoS to system 1. Although the system is provided in a start-up sequence, this system does not tune the QoS after the service provision starts. In contrast, in system 3, at the design phase of the system, the QoS tuning operations based on BCM are realized in addition to the functions of both system 1 and system 2. Therefore, because of the changes in the system's operational conditions, the system can tune the QoS flexibly after the service provision starts.

The experiment was conducted using the following procedures.

(1) Provide a user requirement to the system.
(2) Start the service automatically.
(3) Observe the system's behavior and the M_QoT.

The M_QoT values of each system were observed with respect to the changes in AS_Ap, AS_Pt, and AS_Nt under the initial conditions of R_S_Ap, R_S_Pt, and R_S_Nt. Moreover, we assumed that the user requirement was satisfied and that M_QoT

was controlled appropriately when the observed M_QoT was in the range of $-0.05 \leq M\_QoT \leq 0.05$.

5.3. Results of experiment 1

The results of experiment 1 for system 1, system 2, and system 3 are shown. In the figures shown, the x-axis is the time and the y-axis is the M_QoT value.

First, we describe the experimental result for system 1: service provision via the multimedia communication system without the QoS control function. The result is presented in Fig. 8.

After service provision started, a decrement of M_QoT was observed immediately and degradation of M_QoT continued because system 1 was incapable of controlling the situation of the service provision by itself. System 1 was designed with no policy to handle the user requirement and the actual platform and network environment. Therefore, stable service provision could not be ensured. It was difficult to provide adequate services for the users.

Next is the result of the service provision for system 2, which is based on AMCS with FDS, The result is shown in Fig. 9.

After the service provision started, a decrement of M_QoT was observed immediately, and it continued thereafter. Compared to system 1, M_QoT reached a higher level due to the effect of FDS. For period [A] in Fig. 9, the decrement of M_QoT continued. Subsequently, M_QoT improved automatically by itself. However, degradation of M_QoT was observed again and continued for period [B]. Although M_QoT recovered similarly to the recovery shown after period [A], M_QoT decreased again. Oscillation was observed, showing that system 2 is unstable.

The result shows no beneficial effect of FDS to maintain an adequate M_QoT. Therefore, this situation is not preferable because the assumed environment in the design of system 2 differs from the actual environment. The QoS of the system decreased overall.

Although the QoS of system 2 recovered because some change occurred after period [B], the service provision is not stable because the essential cause of degradation was not removed at that time. Such a weakness of system 2 is attributable to the inability of the system to process the differences between the operational conditions of the assumed environment at the design phase and the actual environment during run time.

Finally, the result of the service provision for system 3 is presented in Fig. 10.

When the user requirement was issued and service provision started, the decrement of M_QoT was observed immediately, and then it continued. Therefore, the control operations to recover M_QoT were activated for period [A], as portrayed in Fig. 10. The resource was not degraded rapidly and remained sufficient. The cause of the degradation of M_QoT was determined to be a decrement of the other AS_Ap because the value of the quality of M_QoT at that time

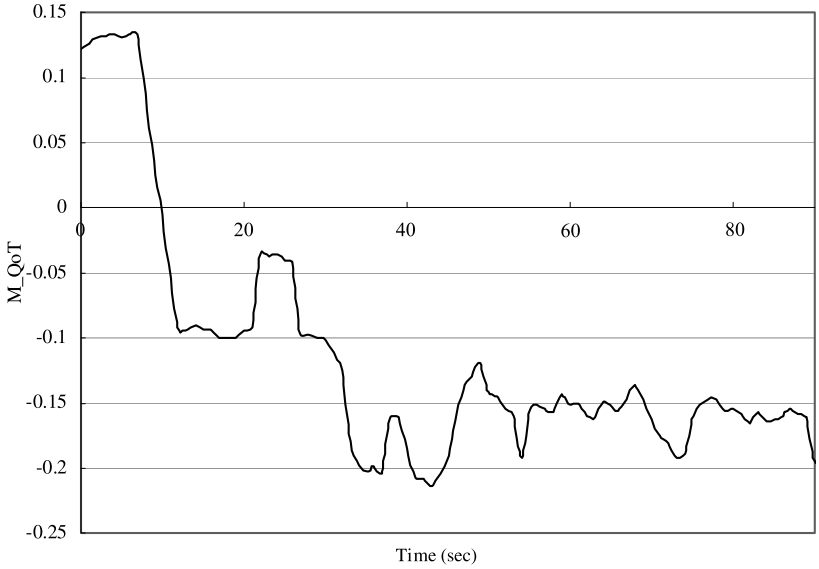

Fig. 8. Behavior of multimedia communication system without a QoS control function.

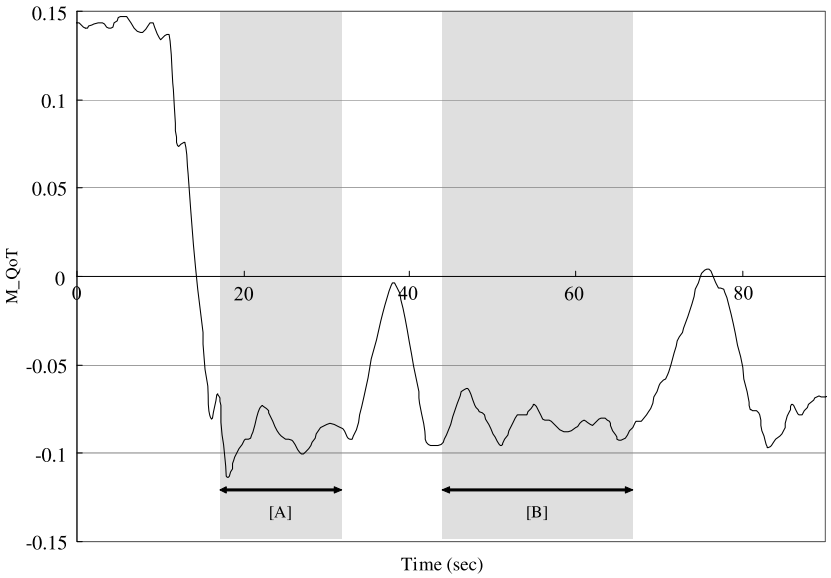

Fig. 9. Behavior of AMCS with FDS.

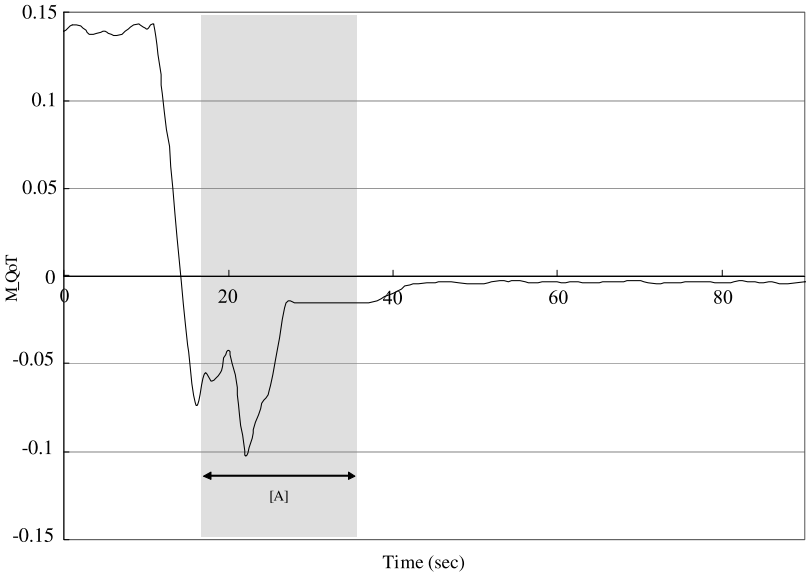

Fig. 10. Behavior of AMCS with FDS and BCM.

was high. The other AS_Ap was found. It recovered by decreasing its quality slightly, and M_QoT was thereby improved. Subsequently, the stable service of system 3 recovered because the AMCS with BCM can detect changes and execute counter operations based on BCM, even when the actual environment at run time is different from the assumed environment at the design phase.

These results demonstrate that the proposed scheme resolves the causes of degradation by considering other possible service components that can omit the excess quality. The proposed scheme, therefore, realizes stable service provision when the service provision starts.

5.4. Results of experiment 2

The results of experiment 2 for system 2 and system 3 are shown. In the figures shown, the x-axis is the time and the y-axis is the M_QoT value.

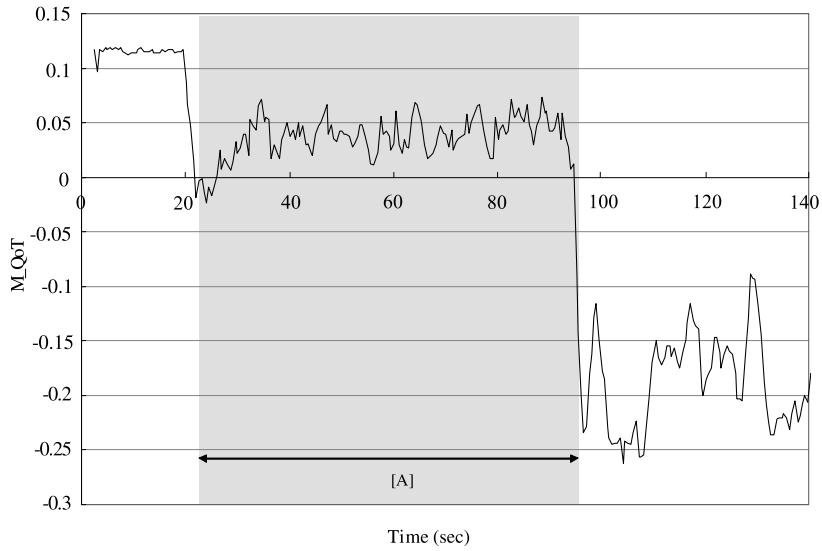

Fig. 11. Behavior of AMCS with FDS.

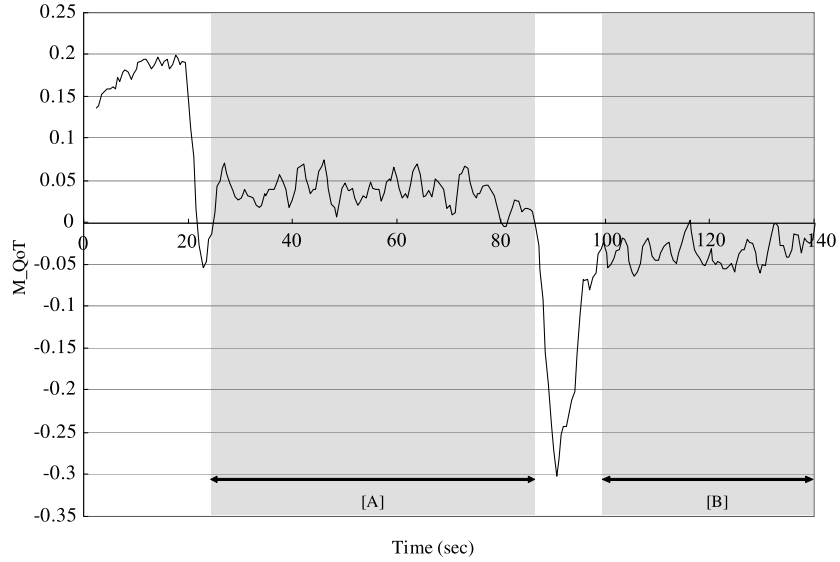

Fig. 12. Behavior of AMCS with FDS and BCM.

First, we describe the experimental result for system 2, which is AMCS with FDS, as shown in Fig. 11.

When the user requirement was issued and service provision started, stable M_QoT was observed. However, after period [A], a decrement of M_QoT was observed and the degradation of M_QoT continued. This happened because system 2 was incapable of controlling the service situation after the service provision conditions changed. Therefore, this situation is not preferable because the assumed environment in the design of system 2 differs from the actual environment in the service provision. The QoS of the system decreased overall during the service provision.

Next, the result of the service provision of system 3, which is AMCS with FDS and BCM, is presented in Fig. 12.

After the service provision started, stable M_QoT was observed and continued. However, after period [A], a decrement of M_QoT was observed. The control operations to recover M_QoT were activated and it

recovered. The cause of the degradation of M_QoT was determined to be a decrement of AS_Pt because the value of the sender CPU at that time was degraded. Therefore, M_QoT improved by decreasing its quality slightly in period [B]. The stable service of system 3 recovered because the AMCS with FDS and BCM can detect a change and execute a counter operation based on BCM when the actual environment changes at run time.

These results demonstrate that the proposed scheme realizes stable service provision against service degradation during the service provision.

5.5. Discussion

In a multimedia communication system, it is important to provide suitable services with respect to the given user requirements and the actual platform and network environment during run time. As described in Sections 5.3 and 5.4, it is difficult to provide stable services for users if the requirements and environments are not considered and the operations after the start of the service provision are ignored, as they were by system 1 and system 2. In contrast, stable service provision can be achieved by using system 3 with the appropriate design and operations based on the BCM of FDS.

Generally, because the organization of a multiagent system is realized dynamically during run time, the relations between the design specifications of individual agents and the actual behavior of the system cannot be clearly associated. Therefore, it becomes difficult to tune the knowledge of each agent based on the behavior of the actual system. However, under the proposed scheme, a designer of multiagent systems can specify the behavioral characteristics and operations for the required QoS control of the target system; as a result, the designer can also define the behavioral knowledge using the operations as design specifications of the agents. Then, testing and debugging of the agents can be conducted by using the design specifications based on FDS. This practical solution solves (P1) regarding the management of knowledge of the multiagent system.

From the viewpoint of multiagent systems for multimedia communication applications, QoS control is performed based on the heuristics of the system operators. Sometimes the system could go into an infinite loop or stop the system behavior without sufficient QoS control capabilities. In contrast, by using the BCM of FDS, it is possible to determine the characteristics of the change in the system behavior and control operations to address the change in QoS during the system run time. The scheme proposed in this paper presents clear insight into the policy of design and control of behavior of a multiagent system. It also can improve the overall system performance, as confirmed by the experiments described in Sections 5.3 and 5.4. In addition, an agent of a multiagent system behaves based on its own design specification. However, it is difficult to observe and control the overall behavior of a multiagent system. In the proposed scheme, an individual agent can be designed and managed systematically based on FDS and its BCM. Each agent observes its own behavioral situation and controls the situation independently to maintain the behavioral characteristics of the whole system. Consequently, monitoring some aspects of the system behavior, such as the changes in Fig. 3 QoS of the whole system, is simplified. Moreover, we can find the root cause of the degradation of QoS in order to maintain the system during run time. The proposed scheme has the capabilities to observe and control the behavior of the multiagent system with flexible QoS control. Therefore, a solution to problem (P2) is available.

6. Conclusion

As described in this paper, we proposed the FDS and its BCM as new models for observing and controlling the behavior of a multiagent system. Using the proposed scheme, a multiagent system was created with flexible QoS control capabilities that can correctly process changes in the operational situations of a system to maintain the required QoS as well as its behavioral characteristics. To evaluate the effects of the proposed scheme, we applied these models to a multimedia communication system constructed as a multiagent system. Experiments were performed using a multimedia communication prototype system. The results confirmed that the proposed models provide an effective scheme for the designers of multiagent systems. The models resolve several of the current problems described in Section 1.

In future work, we intend to construct a multimedia communication system of a multiagent system that can adapt to the permanent changes in QoT. This work includes many technical challenges that we must further investigate. Additionally, we will apply the proposed models to the service provision systems of other types of applications in a ubiquitous environment. We

aim to achieve a multiagent system that provides user-oriented, stable service by enhancing the proposed models. A series of studies similar to this one would contribute to the practical use of multiagent based approaches in many application domains.

References

[1] A. Takahashi, S. Konno, T. Suganuma, T. Kinoshita and N. Shiratori, A design of agent-based multimedia component in flexible network layer, in: *Proc. of the Int. Conf. on Advanced Information Networking and Applications*, 2003, pp. 570–573.

[2] C. Forgy, OPS5 User's Manual, *Technical report CMU-CS*, 81–135, 1981.

[3] DASH, Dash-distributed agent system based on hybrid architecture!, http://www.agent-town.com/dash/index.html.

[4] F. Zambonelli, N.R. Jennings and M. Wooldridge, Organizational abstractions for the analysis and design of multi-agent system, *Lecture Notes in Computer Science* **1957** (2001), 235–251.

[5] G. Tesauro, D.M. Chess, W.E. Walsh, R. Das, A. Segai, L. Whalley, J.O. Kephart and E.R. White, A multi-agent systems approach to autonomic computing, in: *Proc. Int. Conf. on Autonomous Agents and Multiagent Systems*, 2004, pp. 464–471.

[6] IBM, Autonomic vision & manifesto, http://www.research.ibm.com/autonomic/manifesto/.

[7] IDEA, Idea-interactive design environment for agent system, http://www.ka.riec.tohoku.ac.jp/idea/index.html.

[8] JMF, JMF home page, http://java.sun.com/products/java-media/jmf/.

[9] J.M. Serrano and S. Ossowski, A compositional framework for the specification of interaction protocols in multiagent organizations, *Web Intelligence and Agent Systems* **5**(2) (2007), 197–214.

[10] L.K. Wickramasinghe and L.D. Alahakoon, Dynamic self organizing maps for discovery and sharing of knowledge in multi agent systems, *Web Intelligence and Agent Systems* **1**(3) (2005), 31–48.

[11] M. Randles, D. Lamb and A. Taleb-Bendiab, Engineering autonomic systems self-organisation, in: *Proc. of the Fifth IEEE Workshop on Engineering of Autonomic and Autonomous Systems*, 2008, pp. 107–118.

[12] M. Xu, L. Padgham, A. Mbala and J. Harland, Tracking reliability and helpfulness in agent interactions, *Web Intelligence and Agent Systems* **1**(5) (2007), 31–46.

[13] N.R. Jennings, On agent-based software engineering, *Artificial Intelligence* **177**(2) (2000), 277–296.

[14] N. Venkatasubramanian, C. Talcott and G.A. Agha, A formal model for reasoning about adaptive QoS-enabled middleware, *ACM Trans. on Software Engineering and Methodology* **13**(1) (2004), 86–147.

[15] R.G. Smith, The contract net protocol: High-level communication and control in a distributed problem solver, *IEEE Trans. Comp.* **C-29**(12) (1980), 1104–1113.

[16] S.A. DeLoach, W.H. Oyenann and E.T. Matson, A capabilities-based model for adaptive organizations, *Autonomous Agents and Multi-Agent Systems* **16**(1) (2008), 13–56.

[17] S. Fujita, H. Hara, K. Sugawara, T. Kinoshita and N. Shiratori, Agent-based design model of adaptive distributed system, *The International Journal of Applied Intelligence, Neural Network and Complex Problem-Solving Technologies* **9**(1) (1998), 57–70.

[18] T. Kinoshita and K. Sugawara, ADIPS framework for flexible distributed systems, in: *Multiagent Platforms*, T. Ishida, ed., LNAI, Vol. 1599, 1998, pp. 18–32.

[19] T. Uchiya, T. Maemura, L. Xiaolu and T. Kinoshita, Design and implementation of interactive design environment of agent system, in: *Proc. 20th Int. Conf. Industrial, Engineering and Other Applications of Applied Intelligent Systems*, LNAI, Vol. 4570, 2007, pp. 1088–1097.

[20] W.H. Oyenan and S.A. DeLoach, Design and evaluation of a multiagent autonomic information system, in: *Proc. Int. Conf. on Intelligent Agent Technology*, 2007, pp. 182–188.

[21] W. Jiao, J. Debenham and B. Henderson-Sellers, Organizational models and interaction patterns for use in the analysis and design of multi-agent systems, *Web Intelligence and Agent Systems* **3**(2) (2005), 67–83.

[22] Y. Liu, A. Fekete and L. Gorton, Design-level performance prediction of component-based applications, *IEEE Trans. on Software Engineering* **31**(11) (2005), 928–941.