LF-008

AMUSE: An Agent-based Middleware for QoS-aware Ubiquitous Services

Takuo Suganuma Yoshikazu Tokairin Hideyuki Takahashi Norio Shiratori

1. Introduction

In recent years, researches on ubiquitous computing environment and service provisioning on the environment has been greatly accelerated[1,2]. In the future, user demands on these ubiquitous services will move into much richer ones such as multimedia communication services. Thus, we are promoting research and development on fundamental technologies aiming at post ubiquitous computing environment, including multimedia service provisioning by cooperative behavior of audio-visual home electric appliances. To provide necessary and sufficient QoS that satisfies user requirement is an intrinsic problem to realize such ubiquitous services. To address this, we have to consider not only user context, but also resource context of hardware, network and software. This is because resource availability tends to be poor and unstable in ubiquitous environment, including laptop PC, PDA and home appliances with wireless networks. In addition, multiple users would be given the ubiquitous services simultaneously, thus problems of effective resource sharing and assignment should be addressed.

In this research, we are aiming at investigating ubiquitous service construction scheme in order to provide QoS-aware and stable services against changes of resource status and user's situation. Our unique idea is effective handling of multiple contexts including user context and resource contexts. To accomplish the objective, we apply agent-based middleware approach. The remarkable feature of this approach is agentification of each entity in overall ubiquitous environment. We embed context management ability and cooperation ability for conflict resolution on multiple contexts to each agent. Furthermore, we give maintenance mechanism for long-term context to accumulate and reuse history and experiences of past cooperation among agents. The individual behavior and the cooperative behavior of agents would make the QoS-aware service provisioning possible.

In this paper, we propose a multiagent-based middleware for ubiquitous computing environment, called AMUSE (Agent-based Middleware for Ubiquitous Service Environment). Moreover, we describe design of AMUSE focusing on the service construction scheme for QoS-aware service provisioning considering the multiple contexts. Finally we evaluate our proposal by results of some simulation experiments.

2. Concept of AMUSE

2.1 Target issues

Following three technical issues need to be addressed to provide QoS-aware services on ubiquitous computing environment that consists of computers and audio-visual home electric appliances.

(P1) Resource context maintenance

Here we define ``context'' as situation of target entity at time *t* and temporal changes of the situation after/before time *t*. The situation of target entity is represented as internal representation model of the entity. Previous works for context awareness have been mainly focusing on user context acquisition scheme including locating information of the user. However, in terms of resource of entity, it was treated as only a value of the target resource parameter at time *t*, not as ``context''. In ubiquitous environment that consists of many kinds of entities in different level of functionality and performance, it is important to consider resource context efficiently as well for proper QoS control.

(P2) Multiple context coordination

In ubiquitous environment on which heterogeneous entities coexist, QoS should be maintained in the range from entity level to overall system level. To do so, we have to consider not only input/output specification of the entity, but also multiple context coordination including user context and resource context.

(P3) Non-deterministic property of service construction

Furthermore we need to cope with problem of mutual dependency and interoperability among entities that are not resolved deterministically from analysis of static specifications of entities. This is because each entity is basically designed to work by itself, not designed to work with other entities cooperatively. Thus, services constructed from the entities would not work properly in accordance with the specifications.

2.2 AMUSE framework

In AMUSE, we solve the technical issues P1 to P3 described in the previous section with the following two approaches.

(R1) Agentification of each entity

``Agentification'' is a process to make a target entity workable as an agent by adding knowledge processing mechanism to the entity. In our proposal, we agentify each entity respectively. Also we add context management ability and cooperation ability to resolve context conflict to the agents. Moreover, we embed long-term-context maintenance ability to the agents to accumulate cooperation history and experiences. This is a solution to P1.

(R2) Multi-context-based Service Construction scheme

To realize QoS-aware ubiquitous service construction considering multiple context, we propose contract-based service construction scheme of agents. This will be a solution to P2. Agents make organization based on CNP (Contract Net Protocol). Moreover, we model heuristics and dependency information on cooperation history in past among agents as long-term context among agents. This kind of context is also managed by the agent. By using this context, agents can construct more advanced services employing lessons learned. This would be a solution to P3.

The fundamental framework of AMUSE is shown in Fig.1. AMUSE consists of four layers, i.e., Primitive Agent layer (PA),

Research Institute of Electrical Communication / Graduate School of Information Sciences, Tohoku University

Agent Relationship layer (AR), Agent Organization layer (AO), and Ubiquitous Service layer (US), based on concept of symbiotic computing [3]. PA makes physical entities to agents. In PA, context of each entity is managed by corresponding agent. In AR, inter-agent relationship based on long-term context among agents is created and maintenanced. In AO, agent organization is constructed based on the context in PA and AR, when user requirement to specific service is issued. On the top layer US, actual ubiquitous service is provided to users.

Service provisioning process in AMUSE framework consists of the following six steps; (1) Entity agentification, (2) Inter-Agent Relationship updating, (3) User requirement acquisition, (4) Service construction, (5) Service provisioning and (6) User evaluation.

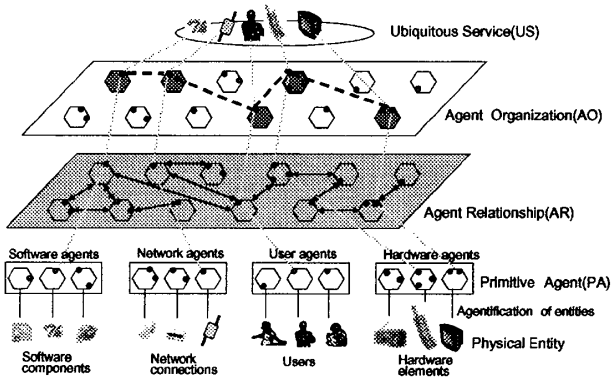

Fig1. AMUSE framework

2.3 Agent-architecture of AMUSE

The basic architecture of agent in AMUSE is shown in Fig.2. Here Cooperation Mechanism (CM) is a mechanism for exchanging messages among agents. Domain Knowledge (DK) is a knowledge-base system to store and activate various domain knowledge concerning the target entity. Based on the knowledge, agent monitors and controls the target entity, and makes actions to other agents. Entity Processing Mechanism (EPM) is an interface between DK and target entity. It passes events from entity to DK such as exceptions, and directs control instruction from DK to entity.

DK consists of three subsystems, i.e., Working Memory, Inference Engine and Rule Base. In Working Memory and Rule Base, set of Facts and Rules are stored respectively. Inference Engine refers to the Rules and Facts, and works as production system. By employing this inference mechanism, agent performs interaction with other agents and controls the target entity. DK has management functions for service construction in AMUSE. These functions are implemented as Rule set and related Facts in DK. IAR Management is module for maintenance of IAR such as creating, updating and deleting of it. Context Management is module for monitoring and recognizing context of the entity. Moreover Contract Management is module to maintenance contract conditions and contract status in case that contract relationship is created with other agents.

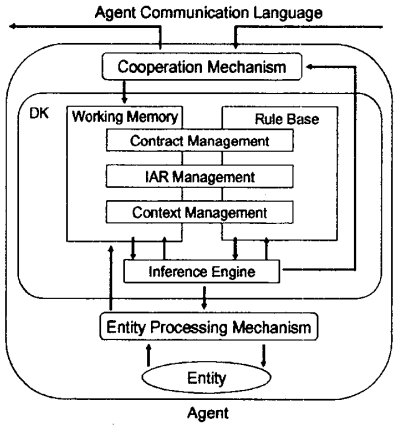

Fig2. Agent Architecture for AMUSE agent

3. Service construction scheme in AMUSE

3.1 Creation of IAR

IAR consists of three kinds of relationships, they are, Tight-Relationship (TR), Group-Relationship (GR) and Competing-Relationship (CR). These relationships are supposed to be most basic ones for constructing agent organizations.

(a) Tight-Relationship(TR)

TR is created among agents that provide some services by constructing organization of the agents. By using this relationship, the agent becomes possible to have past cases of successes and failures in cooperation.

(b) Group-Relationship(GR)

GR is given to group of agents that have some potential dependencies. For instance, there is GR among hardware entities such as desktop PC, speakers and PC displays. By using GR, it is possible that agent informs changes in their states frequently to the agents within the group.

(c) Competing-Relationship(CR)

CR is formed among agents that have same function. Why this relationship is introduced is that these agents would compete when task announcement of the function is issued. By using CR, the competing agents inform their status each other routinely, and they can make good organization effectively when CNP-based negotiation runs.

3.2 Knowledge representation of IAR

Fig.3 shows knowledge representation of IAR. IAR is represented in set of *iar* which is an individual relation. The *iar* consists of *id*, *name*, *func*, *rel*, *state*, *ava*, and *shv*. The *id* is an integer value to identify each agent. The *name* is a name of agent against which the *iar* points. The *func* is an expression of functionality of the entity such as specification of I/O. It is a set of type. The *type* represents each function. For instance, the agent that has I/O function of the voice has audio-output and audio-input in *type* variable. The *rel* indicates type of relationship such as *TR*, *GR* and *CR*. The state indicates present state of agent. Here, *running* means state of service is being provided, *online* means state of standby, and *offline* means state of service cannot be provided, respectively. The *ava* indicates degree of effectiveness of the agent. It consists of *ace*, *us-effectivity*, etc. The *ace* (A) indicates acceptance ratio to task announcement or bid of the agent.

A=b1/t (or a/b2)

Here, b1 is number of times that agent has been received bid, *t* is number of times that agent has sent task, *a* is number of times that agent has been received award and *b2* is number of times that agent has sent bid.

The *us-effectivity* (*E*) indicates metrics of evaluation that agent receives after providing user with service.

E=p/n

Here, *p* is total number of times of good evaluation, and *n* is total number of times of good and bad evaluation based on strength value of Ja-Net[2]. The initial values of *p*, *n* and *E* are set to 1.

The *shv* is referred as common index used by each agent. For instance, there is operating rate (*O*) that indicates how often agent is usually used.

O=u/l

Here, *u* is number of times in which it is used. The *l* is fixed period which is decided by user. It can be used to understand trend of changes in the preference of the user according to the *shv*.

```
IAR ::= {iar}*
iar ::= < id, name, func, rel, state, ava, shv >
id ::= int
name ::= string
func ::= {type}*
type ::= audio-output | audio-input | image-output | image-input | ...
rel ::= TR | GR | CR
state ::= running | online | offline
ava ::= < ace, us-effectivity, ... >
shv ::= < operating rate, ...>
```

Fig.3 Knowledge representation of IAR

3.3 CNP-based service construction with IAR

Our scheme which is based on CNP builds agent organization using three kinds relationship. CNP is a mechanism to make contract relationship among agents by exchanging messages such as task announcement, bid, and award, shown. Here, we explain features of service construction scheme with IAR.

(1) Case of Tight-Relationship (*TR*)

We assume that agent A has a relationship of type *TR* with both agent B and agent C whereas no relationship exists between B and C. *TR* between A and B indicates that trouble was occurred when they cooperation in the past, and *TR* between A and C indicates no trouble in the past. When B and C receives the task announcement from A, they refer to each IAR. Here, B does not send bid because *TR* against A is bad. That means the trouble in cooperation would occur this time too. On the other hand, C sends bid because C judges from TR that it would contribute to the task announced. In fact, it is possible to reduce trouble in cooperation by agent considering coordinated relationship in the past.

(2) Case of Group-Relationship (*GR*)

We assume that agent A has no relationship with both agent B and agent C whereas relationship of type *GR* exists between B and C. Here, C recognizes that *GR* against B exists when C judges the task announcement from A. Then C sends bid if C judges that B can provide service by referring to state in IAR. On the other hand, C ignores the task announcement if B can not provide service. In fact, it is possible to reduce the trouble in cooperation by agent considering dependency of the agents.

(3) Case of Competing-Relationship (*CR*)

We assume that relationship of type *CR* exists between agent B and agent C whereas agent A has no relationship with both B and C. B and C receive the task announcement. Each agent checks IAR of type *CR* if it can process the task. When agent has *CR*, it refers to state of the *CR*. For instance, B sends bid in case that B judged the value of *us-effectivity* on this task is higher than that of C. On the other hand, C ignores the task announcement in case that it judged *us-effectivity* of B is higher than C. In fact, it is possible to efficient construction of service by consideration of state of same function agent.

4. Simulation and evaluation

4.1 Implementation

We implemented agents based on AMUSE to perform some simulation. In the implementation, we employed agent-based programming environment DASH that is based on multiagent programming framework ADIPS[4]. We used DASH programming environment because agent which is developed for simulation can easily be reused when we build the real-world system in future.

4.2 Evaluation method

We performed simulation of system behavior based on AMUSE. In this simulation, QoS awareness of the system is measured, and we investigate how much the QoS awareness is improved by introducing AMUSE. To measure the QoS awareness, we apply User Request Achievement (*URA*) level. Using this metrics, we can measure how much the user requirement is fulfilled with provided quality of service by the system. Details of *URA* are described later.

Here, three entities including a hardware entity, a software entity and a network entity are making organization and providing service to a User. The user issues ``User Request QoS'' and the system provides service with ``Provided QoS''. Hardware Agent (HA) monitors CPU resource context and Network Agent (NA) monitors bandwidth resource context. On the other hand, Software Agent (SA) has knowledge concerning mapping from resource availability onto actual user level QoS.

The QoS evaluation of service is based on *URA*. *URA* is calculated by comparison between User Request QoS *RU* and Provided QoS *SV*. Here, ru_i is an element of *RU* and it represents User Request QoS on service element *i*. Also sv_i is an element of *SV* and it represents Provided QoS on service element *i*. The value of ru_i and sv_i is from 1 to 10. Here, *URA* on service element i, i.e. URA_i is represented as follows:

$SV = \{sv_1, sv_2, \ldots\}$

$RU = \{ru_1, ru_2, \ldots\}$

$URA_i = (sv_i - ru_i)/10$

This means that, if URA_i is above zero, the user requirement is fulfilled, and if it is below zero, the requirement is not satisfied.

In this evaluation, the number of service elements is assumed to be two (i=1,2) and *URA* indicates the total *URA*, that is, a mean value of URA_1 and URA_2 for simplification.

We performed simulation of service construction with above conditions 500 times. The agent constructs *CR* immediately after simulation beginning. When SA constructs service, it refers to NA's bid and IAR. If SA judges that other agent is suitable, it disregards the task even if the task is acceptable. And if agent receives bid by two or more agents that can fulfill user requirement, agent sends award to agent with the highest value of *E* of IAR after referring to the value of *E*. In the actual simulation, we use only the strength of Ja-Net for the value of *E*.

User evaluation is assumed good if sv_i is within from 120% to 100% when ru_i is regarded as 100%. In this case value *E* is set to 1. It is assumed bad in case that sv_i exceeds 120% or is below 80%, and the value is set to -1. Otherwise it is regarded as usual, and the value is set to 0.

We compare three patterns of agent behaviors, i.e., our proposal IAR-based approach), the case considering only user context User-request approach), and the case considering only the maximum QoS value of agent for QoS without consideration of resource context Maximum approach). For the simulation, resources of HA and NA are assigned random values in every service construction. Also we give dependencies of user request in three patterns, which is high quality (7 to 10), middle quality (4 to 7), and low quality (1 to 4).

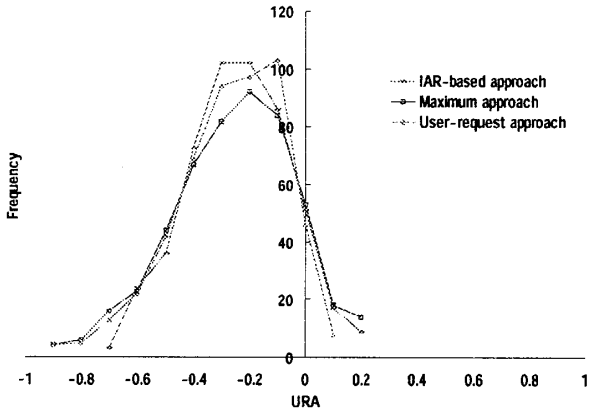

Fig.4 Results of comparison in case that *RU* is always in high

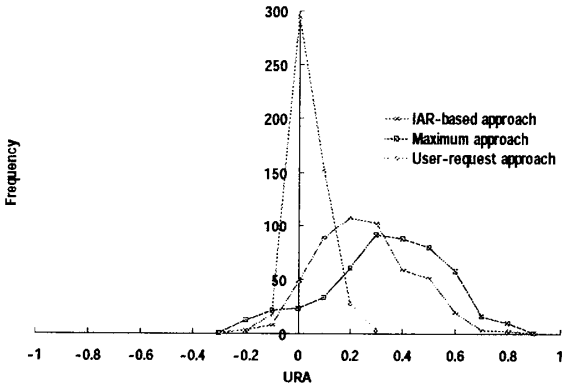

Fig.5 Results of comparison in case that *RU* is always in low

4.3 Simulation results and evaluation

Fig.4 and Fig.5 show the frequency distribution concerning *URA*. Fig.4 is a comparison for case that *RU* is always in high, and Fig.5 is a comparison in case that *RU* is always in low.

From analysis of Fig.4 in case that *RU* is always in high, our approach could achieve the user requirement much more times than User-request approaches. From this result, in case that user requirement is higher than the service environment, it can be understood that the requirement can not be fulfilled even if only user context is considered. Moreover, in case that agent considers only the user request and the maximum value that can be selected, *URA* generally is lower than our approach, when user requirement is not fulfilled. It is understood that some conflict on resource context is occurred. Also it is understood that our approach decreases bad service construction by considering IAR. This is because that IAR decreases the conflict of resource context. From these results our approach is rather effective in this kind of case.

From analysis of Fig.5 in case that *RU* is always in low, *URA* of User-request approach often closes to zero extremely more times than other approaches. In our approach and Maximum approach, the case that *URA* closes to zero is not frequent. But, our approach closes to zero much more times than Maximum approach. Moreover our approach and User-request approach reach much closer to zero than Maximum approach. Thus, we can find that the agents construct organization considering user requirement and IAR effectively. However, in this case, our approach is thought to be meddlesome service for user who does not want excessive quality.

5. Conclusion

In this paper, we proposed a multiagent-based middleware for ubiquitous computing environment, called AMUSE. We described design of AMUSE focusing on the service construction scheme for QoS-aware service provisioning considering the multiple contexts. We also evaluated our proposal with some simulation experiments and confirmed that our proposal would be remarkably useful in ubiquitous environment.

In future, we will implement prototype system using actual hardware devices such as PC and home electric appliances to confirm the feasibility and effectiveness in real-world environment. Further improvement in updating and deletion mechanism of IAR is another possible future goal in our research.

[References]

[1] C. Stefanelli, "Context-Aware Middleware for Resource Management in the Wireless Internet," *IEEE Trans. software Eng.*, vol. 29, no. 12, pp. 1086-1099, 2003.

[2] T. Itao, T. Nakamura, M. Matsuo, S. Tanaka, T. Suda, and T. Aoyama, "Relationship Mechanism for Dynamic and User Preference-Aware Service Creation," *Journal of IPSJ*, vol. 44, no. 3, pp. 812-825, 2003.

[3] K. Sugawara, N. Shiratori, and T. Kinoshita, "Toward post Ubiquitous Computing -Symbiotic Computing-," *IEICE Technical Report*, vol. 103, no. 244, pp. 43-46, 2003.

[4] S. Fujita, H. Hara, K. Sugawara, T. Kinoshita, and N. Shiratori, "Agent-based Design Model of Adaptive Distributed Systems," *The International Journal of Artificial Intelligence, Neural Networks and Complex Problem-Solving Technologies*, vol. 9, no. 1, pp. 57-70, 1998.