Development of CAMUS based Context-Awareness for Pervasive Home Environments

Aekyung Moon, **Minyoung Kim**, **Hyoungsun Kim, Kang-Woo Lee** and **Hyun Kim**
Software Robot Research Team, ETRI
e-mail : {akmoon, mykim, kimhs, kwlee, hkim}@etri.re.kr

Abstract With the advent of pervasive computing environments, the issue related to context-aware middleware is an emerging area of research. We propose proactive services on the basis of CAMUS (Context-Aware Middleware for URC Systems). First, this paper presents the system architecture of CAMUS. CAMUS is a middleware for providing context-aware applications with development and execution methodology. Accordingly, the applications developed by CAMUS respond in a timely fashion to contexts. Secondly, this paper illustrates the contents recommendation and control service agents with the property, operation, and task for the context-aware active services. Next, proposed active services are applied to the pervasive home environment for evaluation of our system. To do this, we implement the TV contents recommendation service agent, control service agent and context-aware task using CAMUS. And we evaluate the performance of Planet communication model of CAMUS.

Keyword: Ubiquitous Computing, Pervasive Computing, Context-aware, Sensor, Network-based Robots

1. Introduction

A significant amount of work has been devoted to context-aware middlewares [3,6,9,13]. These researches remained an experimental level on developing the prototype, because various contextual information and its achievement technologies were considered only for specific applications at specific platforms.

This paper presents the system architecture of CAMUS (context-aware Middleware for URC Systems). CAMUS is to provide a framework for the development and execution of context-aware applications for URC (Ubiquitous Robotic Companion) such as network-based robots and software robots [7]. To do this, CAMUS gathers contextual information from different sensors and delivering appropriate contextual information to different applications. Moreover, CAMUS provides autonomous service agents that are aware of contextual information, so that they can adapt themselves to different situations. Among other capabilities, the contents recommendation service agent provides a way to easily filter interesting items from a large of information contents on the basis of a user preference context.

We also propose CAMUS based context-aware active services and apply these services to pervasive home environments. To do this, we implement TV contents recommendation service agent, context-aware task and TV control service agent on the basis of the service framework and task development methodology of CAMUS. As a result, the proposed CAMUS services are more intelligent than previous applications because it provides context-awareness. That is to say, these services can respond in a timely fashion to contexts even without any explicitly request.

The remainder of this paper is organized as follows. Section 2 introduces the related work context-aware middlewares and services in pervasive computing environments. Section 3 introduces system architecture of CAMUS and Section 4 presents the context-aware service agent and task on the basis of CAMUS. In Section 5, we implement active services for pervasive home environments and experiment on the communication model of CAMUS. Finally, Section 6 summarizes the main conclusion of this study.

2. Related Work

The role of context has recently gained great importance in the field of pervasive computing. Pervasive computing environments are characterized by many sensors that can detect a variety of different contexts. The context is information that can be used to characterize the situation of entities that are considered relevant to the interaction between a user and various applications. This leads to the consideration that applications should take advantage of contextual information, such as user's location, to offer greater services to the user; hence, pervasive computing environments must provide middleware that supports the development of context-aware applications [9].

The study of Schilit has shown the importance of three contexts [11]. The first is a computing context, which includes the information of the available processors, devices accessible for user input and display, capacity and connectivity of the network, and computing power. The second is a user context. The user context represents the user's current location, collection of nearby people, and social situation. The third is a physical context. Examples of this are lighting and noise level. Another context should also be added that accounts for time, such as the hour, week, month, and season [4].

The representative context-aware services are call forwarding [12], conference assistant [5], tracking system [10] and ubiHome [6]. Call forwarding is developed by Olivetti Research Ltd., which utilizes user context. When the telephone is ringing, it transfers the call to the closest telephone to the user's location. The conference assistant was developed by a context toolkit. The context toolkit provides a framework for the development and execution of context-aware applications [4]. The tracking system on the basis of a smart dust sensor, which is envisioned to

"This work was supported in part by MIC & IITA through IT Leading R&D Support Project."

combine sensing, computing, and wireless communication capabilities, gathers user context. In ubiHome, several ubiSensors provides the preliminary context corresponding to user, location and time and ubiService provides context-aware movie player.

3. The Context-aware Middleware for URC Systems

The System Architecture

CAMUS is to provide a framework for the development and execution of context-aware applications for network-based robots. URC is a new concept for a network-based service robot. It allows the robot to extend its functions and services by utilizing external sensor networks and remote computing servers. In the URC, CAMUS plays the important role of the software infrastructure that expands the robot functions and services, improves the context-awareness in the pervasive computing environment, and enhances the robot's intelligence. To do this, CAMUS supports following main functions [7]:

- Gathering contextual information from different sensors and delivering appropriate contextual information to different applications and/or tasks
- Inferring higher level contexts from low level sensed contexts
- Supporting a UDM(Universal data Model) to represent and manage contextual information
- Providing an ECA(Event-Condition-Action) rule-based program language extending the Java program language
- Tracking the location information of users or another entity to support location-based services
- Supporting a service framework enabling the easy integration of the various sensors and legacy applications

Figure 1 shows the system architecture of CAMUS. CAMUS is composed of four parts: CAMUS-MS (CAMUS Main Server), SAM (Service Agent Manager), SA (Service Agent) and Planet.

CAMUS-MS is a framework which collects contexts from SA and uses the contextual information. In addition, it supports a variety of functions for context-aware application development. In particular, CAMUS-MS controls all information about user context including user preference used in contents recommendations and environmental contexts, then sends events by context's changes to applications and helps applications to perform suitable actions for the context. There is another point that is important for the CAMUS-MS: it offers a service framework that can connect to the basic service agent of a robot controller and a variety of software, such as voice recognition, image recognition, and motion detection. In the next section, we describe the detailed components of CAMUS-MS.

SAM is a program that manages and controls SAs within the environment. To do this, SAM is installed within various environments in a location, obtains information from a variety of sensors in the environment, sends the information to CAMUS-MS, receives instructions from CAMUS-MS, and controls the SA in the environment. Therefore, SAM can be installed in any location, such as rooms or offices, and also in a robot platform or PDA.

SA executes the functions of legacy applications and sensors installed on physical places through communication with SAM and CAMUS-MS. Namely, SAs are the software module interfaces of devices and applications that interact with CAMUS-MS. Because SA exposures attribute and action with interfaces that are accessed by CAMUS-MS. For example, suppose that the SA, which provides the information of the user's location, has interfaces o send the id and physical location for the user and the RFID sensor exists in the environment. Then, the SA needs to implement an interface with a device driver supported by the vendor of the RFID sensor. Accordingly, this implemented program becomes SA to transfer the user location.

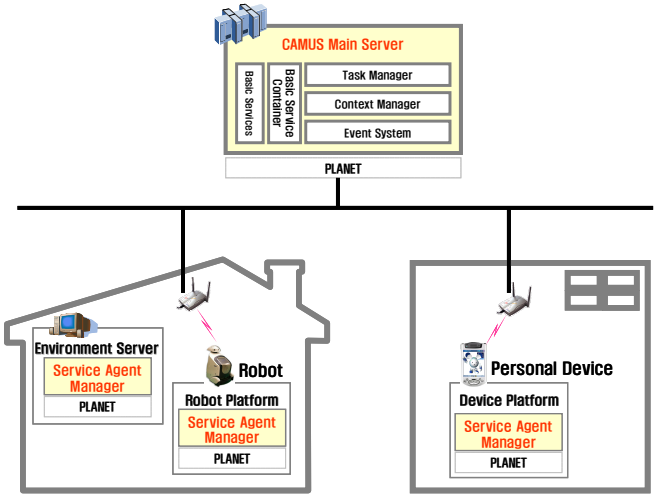

Fig. 1. System Architecture of CAMUS

Planet is the communication framework of a network-based robot system. CAMUS-MS communicates with the SAMs through Planet. A network-based robot system should consider not only the general computing environments, but also the relatively limited device environments in the resources such as embedded systems. Therefore, Planet provides a binary message encoding method to minimize the number of messages to transmit. It also considers efficient methods of transmitting large amounts of data such as voice and images; moreover, Planet handles the long term operation of robots, such as autonomous navigation and speech synthesis. Finally, Planet controls these simultaneous accesses in multi-sessions using locking mechanisms. Planet is developed to support various programming languages including Java, C/C++, and C# being run through various operating systems.

The Component of CAMUS Main Server

Figure 2 shows the main component of CMAUS-MS. CAMUS-MS is composed of six parts: sensor framework, event manager, context manager, task manager, service framework and soft robot.

The sensor framework processes input data from various sources, such as physical sensors, legacy applications, and user commands, while considering the current situation. The sensor framework then transfers the input data to the context manager. For example, voice and image information from users, temperature and humidity information physical sensors and the user schedules from applications and user preference context may be included as sensor information. In particular, the sensor interpreter that is included in sensor framework for the voice

commands is voice recognizer, which translates the voices into textual commands, processes them considering the current situation, and transfers them to the context manager.

The context manager manages context information from the sensor framework, which is represented by UDM. When the contextual information in the environment is changed, the context manager transfers the events to the event manager.

The event manager delivers events to the task manager to supply the necessary contextual information required for task execution. The event notification system is implemented using publish and subscribe model.

The task manager initiates individual tasks, manages the on-going task processes, and executes the actual tasks considering the situation. To accomplish this, the task manager has an inference engine to process the facts and rules supplied by a task.

The service framework coordinates the various sensors and legacy applications. Namely, the service framework manages SAM and SA, and also executes the SA requested by task. The service framework layer also controls devices such as the robot platforms and home appliances.

The user communicates with CAMUS via a Sobot (Software Robot) avatar. Sobot is responsible for the interaction between the CAMUS and users: it transfers user commands to the system and notifies the system messages to users. It is composed of a platform, core, and plug-ins. The platform is a runtime environment for the execution of the Sobot, which may be pre-installed in the various information devices. The core is the minimal module to execute a Sobot and enables communications to/from the platform, I/O processing, and plug-in downloads. In addition, it manages plug-in profiles, knowledge, and user preferences acquired during the services sessions. A plug-in is a special program module required by clients to invoke a specific service, which is downloaded dynamically at the moment of usage.

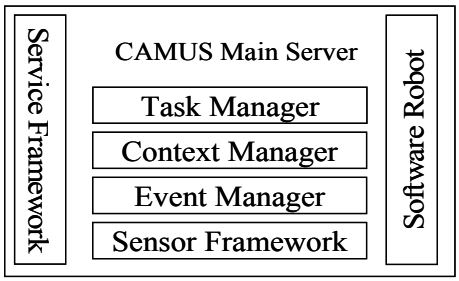

Fig. 2. Main Component of CAMUS-MS

4. Context-Awareness for Pervasive Home

To understand the proposed context-aware active services including contents recommendation services, we first illustrate the scenario that is likely to happen in pervasive computing environments. This example scenario represents a pervasive home domain.

Example 1: Suppose there are a TV in the living room and a TV in the bedroom. The TVs are controlled by voice commands. The users want to execute seamless services in their locations. The applications should take advantage of the contextual information, such as the user's current location, to offer greater services.

Pervasive Home Scenario:

1. The user enters the living room and sits on the sofa.
2. The TV in the living room is turned on and changed the high preference channel automatically.
3. The user controls the volume and channels with voice commands.
4. The user moves to the bedroom.
5. The TV in the living room is turned off automatically.
6. The TV in the bedroom is turned on if the user leaves the TV in the living room on.
7. The user moves to the child's room
8. The TV is not available in the child's room.

To complete this scenario, the context-aware active services in CAMUS consist of the contents recommendation and control service agents, and the context-aware task. The contents recommendation service agent recommends personalization items by comparing contents and user preference context. The control service agent has a responsibility for controlling properties of physical world such as a TV. On delivering event on the context, Figure 3 shows the relationship of task and service agent. Each task executes the actual operation based on the statechart transitions and the ECA rules. When these event and conditions are satisfied, tasks discovery services and executes action of service agents to control physical devices. For example, it can execute the "turning up the volume on TV" operation of TV control service agent considering user voice command and gesture events.

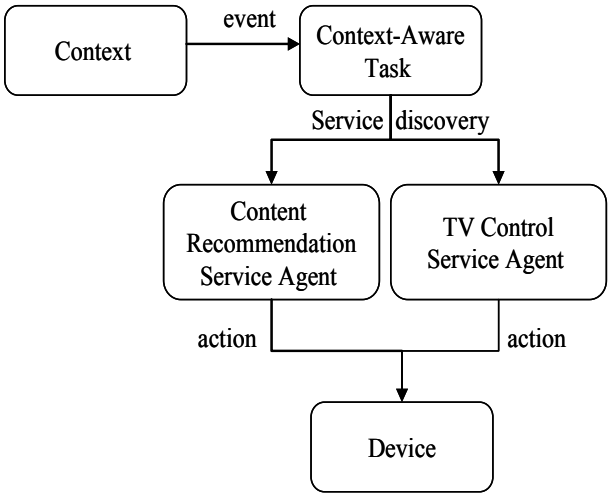

Fig. 3. Relationship between Task and Service Agents

Contents Recommendation Service Agent

The system architecture of a contents recommendation service agent is shown in Figure 4. This SA is consists of recommendation engine and SA interface. The contents recommendation engine contains three components: a information gathering manager, a recommendation manager, and a user model manager.

First, the information gathering manager acquires the items for recommendation from the Internet resource and stores the contents database. For example, the TV programs contents database can be gathered from the broadcasting stations' web servers.

Secondly, the user model manager models a user's preference context from the information recorded in user's profile and update the user preference context according to the information of his/her behavior or feedback, which is stored in a history database. The user model manager consists of two parts: the user profile and user model adaptation parts. The user profile is to generate user preference context. Initially, user model manager gathers the user preference context in an explicit way. The adaptation part is to change the user preference context. In particular, the implicit feedback stored in history database is to adapt user preference context.

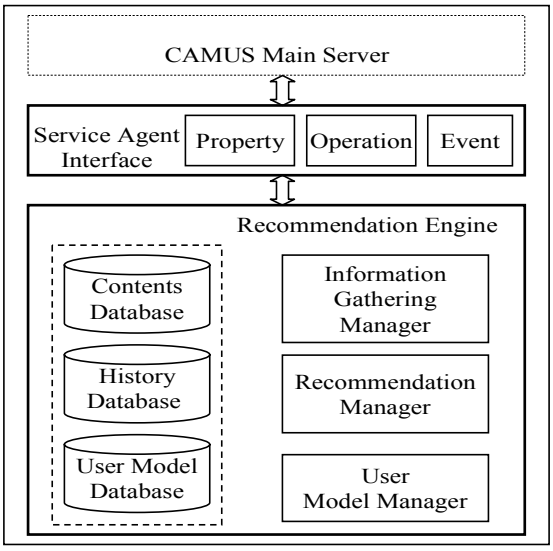

Fig. 4. Components of a Recommendation Engine and Contents Recommendation Service Agents

Finally, the recommendation manager plays an important role in contents recommendation service agent because it has a recommendation module. Recommendation engine is implemented as content-based approach [2]. This matches user's profile to contents and recommends the ones with highest user preference to generate the recommendation lists for the user. The method of computing the preference value about items each for user can be expressed by two functions of specific model and common model (equation (1)). The common model is computed by cosine similarity between keyword vectors from user model and keyword vectors extracting from the contents. The function of specific model is related to application domain.

$$P^{u,i}=\alpha\times P_{specific}^{u,i}+\beta\times P_{common}^{u,i} \quad (1)$$

u: user, i: items
$P^{u,i}$: The preference value of user u about item i
$P_{specific}^{u,i}$: The preference of the specific model
$P_{common}^{u,i}$: The preference of the common model
α: The weight of the specific model
β: The weight of the common model

Context-aware Task

The task is a set of work items that are required to be taken in a specific context or from the user's command. Each work item, which is a unit of action, is described using the ECA convention. The action part describes the operations to be executed by the SA when the incoming event meets specific conditions. SA is accessed by a task through SAM shown in Figure 1.

Figure 4 shows the state transition diagram of a context-aware TV task designed to implement the scenario represented in Example 1. The proposed task uses the two events of UserEntered and SpeechReceived provided by the user to CAMUS. UserEntered is generated by the RFID sensor or image recognition application and is related to the user's location; the SpeechReceived is related to the user's voice command. This task consists of four states: Starting, TVNotAvailable, TVON, and TVOFF. As the user moves, the task checks current user's location, checks the status of the TV, and executes the related operations of the service agents.

- **Starting state**: This is initial state. It branches to the TVON, TVOFF and TVNotAvailable states according to statues of TV and the user's current location.
- **TVNotAvailable state**: This state indicates that TV is not available at the user's current location. If the user's location is changed, it transitions to the Starting state.
- **TVON state**: This state indicates that the TV at the user's current location is turned on. On entering this state, the task executes the recommendation operation of TV contents recommendation service agent. And it changes high preference TV program channel using control service agent. Also the user can control the TV with voice recommend. For example, if user says "turn off TV" and then TV is turned off and transitions to the TVOFF state. If the user location changes, the state transitions to the Starting state and turns off TV. In this state, when the user moves another location with an available TV, the Starting state executes operations of the control service agent and then turns on TV in the location.
- TVOFF state: This state indicates that the existing TV at the user's current location is turned off. If the user location changes, this state transitions to the Starting state. If user says "turn on TV", and then it turns on the TV through control service agent and transitions to the TVON state.

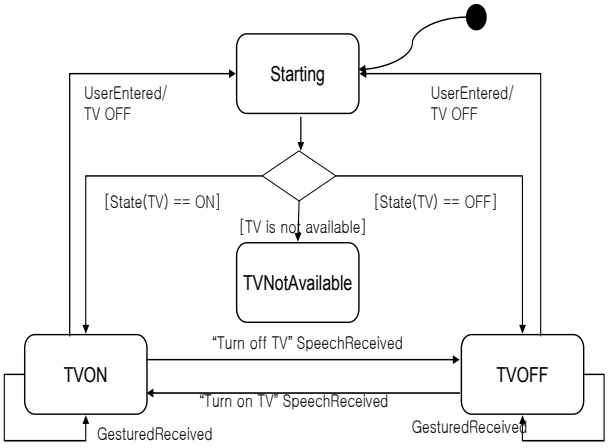

Fig. 5. StateChart Diagram of Context-aware Task

5. Implementation and Experiments

To evaluate the proposed active services, we apply them to the pervasive home environments. Therefore, we implement TV contents recommendation and control service agents. Additionally, a context-aware TV task is implemented. Finally, we experiment the example scenario mentioned above and evaluate the performance of the Planet that is communication framework of CAMUS.

5.1 TV Contents Recommendation and TV Control Service Agents

The rapid expansion of TV broadcasting channels has made it hard to find desired broadcasting programs. Over the past few years a considerable number of studies have been conducted on TV contents recommendation. These TV contents recommendation applications should match a user's desired TV programs and recommend programs with high user preference [1,8].

To implement a context-aware service agent based on CAMUS, the service agent can define property. The property is represented by name and value pair for itself. CAMUS-MS shown in Figure 1 can access the property of the service agent through the service framework to check on the state of a variety of sensors and devices in pervasive computing environments. Furthermore, by changing the value of a property, CAMUS-MS controls devices and software applications. In the design of the TV contents recommendation service agent, we considered the following properties shown in Table 1.

The content of TV programs can be represented in these items: identification information (ID, title), category information (genres/subgenres), broadcast information (channel, the starting time and ending time of the program), content ratings, and keywords. Content ratings and keywords are particularly important to stress. The main reason is that content ratings prevent teenagers from being recommended to view TV programs with violent or sexual contents, and keywords affect the user model of another recommendation service agent. Suppose that user views the related program of the World Cup every so often. Then, the preference value of the World Cup keyword goes up. As a result, the probability of recommendation news related to the World Cup increases in a web news recommendation service agent.

Table 1. Properties of the TV content recommendation service agent

Property	Definition
no_recommend	Number of programs that the user wants to be recommended.
tv_obtain_interval	Number of days between information updates: the interval between when the broadcasting information manager acquires the TV broadcasting information from the broadcasting station's web server.
start_limit	Starting time limitations of a recommended TV program.
end_limit	Ending time limitations of a recommended TV program.
tv_channel_code	Local TV channel code

TV control service agent defines property of IP and port to communicate digital TV or STB (Set Top Box). And the operation has the three types shown in Table 2.

Table 2. Operation of TV Control Service Agent

Control Items	Interface
TV on, off	boolean getPower() void setPower(boolean power)

Channel	void setChannel(int channel);
	void channelUp();
	void channelDown();

Volume	void volumeUp(); void volumeDown();

5.2 Context-aware TV Task

CAMUS supports the PLUE (Programming Language for Ubiquitous Environments) that allows the programmer to develop context-aware tasks. PLUE is basically an extension of Java programming language, and in fact, its compiler is a pre-processor of the Java compiler. This program code is an example of the TV task in the case of TVON state that is described by PLUE. ECA rules in PLUE can be augmented with an event expression so that they are fired only when the expected events are received. That is, they are intensively used in the domain where applications are needed to react to environmental changes. On entering this state, task execute the recommendation operation of TV contents recommendation service agent. And it changes high preference TV program channel with TV control service agent.

Figure 6 indicates the example code of Starting state. $place indicates the service agents existing in the user's current location and $owner identifies the specific user. Also, the user can control the TV using voice commands. For example, if the user says "turn off TV", the TV is turned off and moves to the TVOFF state.

As shown in the example code, the expression "on (SpeechReceived e)" describes the event that starts this rule and the "if (e.symbol == volume up)" describes the condition of the rule; finally, the remainder of the rule explains the action to be taken. The rule in the example is read as, "Whenever the user says volume up, turn up the volume of the TV." Conventional Java method calls are allowed in the rule expression: volumeUp() is an operation of the TV control service agent. If the location of user changes, the state transitions the Starting state and turns off the TV.

```
State TVON {
  entry {
    TVProgram info =
      $place.tv. getRecommend($owner);
    $place.tv.setChannel(info.channel);
  }
  Exit {$place.tv.setPower(false); }
  on event SpeechReceived(e)
  condition(e.symbol == 'volume up')
        $place.tv.volumeUp();

  on event SpeechReceived(e)
  condition(e.symbol == 'volume down')
        $place.tv.volumeDown();
}

Transition StateTVON to StateTVOFF {
    on event SpeechReceived(e)
    condition(e.symbol == "tv off")
        $place.tv.setPower(false);
}
```

```
Transition StateTVON to Starting {
    on event UserEnetred(e)
    condition($owner.lastLocation()!=
    $owner.currentLocation())    {
        $owner.tv = true;
        $place.tv.setPower(false);
    }
}
```

Fig. 6 Example Code of the Starting State

The Figure 7 shows the results of experiments based on the scenario of example 1. The user's current location is recognized by the RFID sensor and camera sensor. When the user enters a living room, the TV is turned on and set the channel with high preference automatically (Fig. 7-(a)). Also, when the user moves to the bedroom, the TV is turned off in the living room and then the TV is turned on in the bedroom automatically (Fig. 7-(b) and (c)). However, if the user moves the kid's room, the TV is turned off in the bed room only because the TV is not available in the kid's room (Fig. 7-(d) and (e)).

(a) The user is in living room

(b) The user moved to bed room

(c) The user is in bed room

(d) The user moved to kid's room

(e) The user is in kid's room

Fig. 7. Snapshot of Experiments

5.3 Experiments

We evaluate the performance of CAMUS using a communication model. Because CAMUS is a network-based distributed system, we evaluated the performance with Planet communications module of CAMUS shown in Figure 1. Figure 8 shows the communication model for the performance evaluation. We model the three components, which consists of Planet server, Planet client and Evaluator synchronizer.

The Planet server registers the objects to be used in these experiments. The resource monitor is to provide the utilization of memory and CPU of a server-side. The client makes evaluators to communicate with Planet server. We evaluate with varying the number of evaluators. The evaluator synchronizer controls the remote method calls made by evaluators.

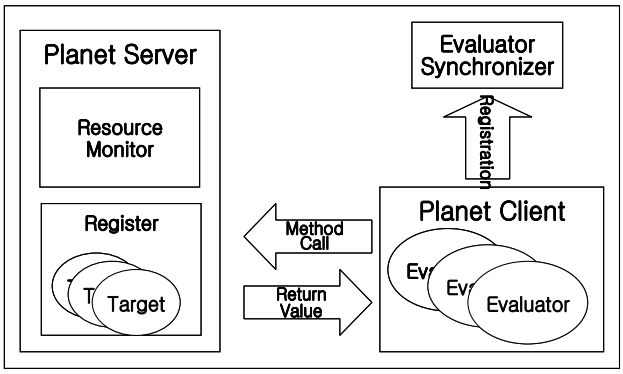

Fig 8. Structure of the Communication Model

Table 3 shows the experiment parameters for specifying the settings under a wide variety of system configuration. The performance metric used in the experiments is the response time in millisecond. The response time is measured as the difference between when remote object is requested and when the result value is returned.

Table 3. Experiments Parameters

Parameters	Description
Evaluator Number	Number of evaluators
Test Type	Argument type of remote object
Frequency	Number of remote method calls
Array length	Array length when the argument of method of remote object is array type.

We made the Plant client-side GUI (Graphic User Interface) for performance evaluation with ease. Figure 9 shows the result of performance evaluation with client-side GUI. This figure 9 indicates that 100 evaluators call the remote object with the byte type argument. The mean value of response time with 100 frequencies is used for the statistical analysis of experiment results.

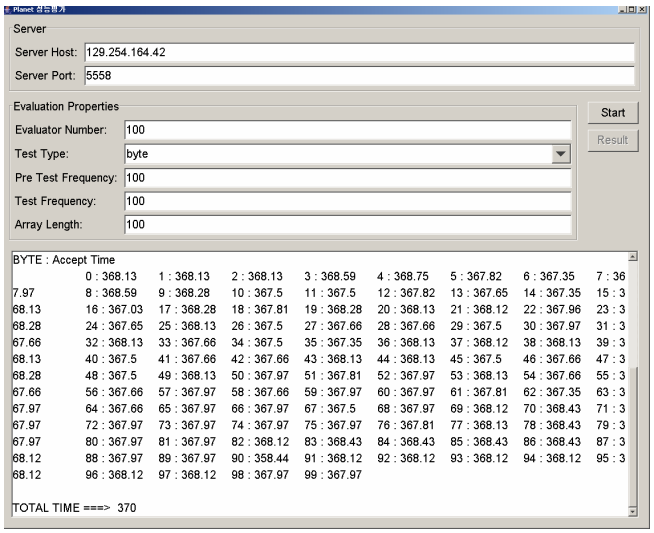

Fig. 9. Performance Evaluation using Client-side GUI

6. Conclusions

We introduce CAMUS which is a context-aware server framework for URC. CAMUS provides a substructure which allows a robot to be connected with a server for utilizing contextual information and ultimately giving proactive services to users. This paper also proposes contents recommendation service agent and context-aware task on the basis of the service framework and task development methodology of CAMUS.

We apply proposed CAMUS based service agents to pervasive home environments for the functional evaluation. As a result, implemented context-aware TV task respond in a timely fashion to contexts such as user's current location and voice command. In addition, we evaluate the performance with Planet communications module of CAMUS.

References

[1] L.Ardissono, et al., "Personalized recommendation of TV Programs," *LNCS* 2829, pp. 474-486, 2003.

[2] M.Balabnovic and Y. Shoham, "Content-Based, Collaborative Recommendation," *CACM* 40(3), pp. 66-72, 1997.

[3] G. Biegel and V. Cahill, "A Framework for Developing Mobile, Context-aware Applications," *Proc. Pervasive Computing and Communications (PerCom)*, 2004.

[4] A. Dey, D. Abowd and D. Salber, "Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications," *Human-Computer Interaction (HCI) Journal* (16), 2001.

[5] A. Dey, M. Futakawa, D. Salber, and G. Abowd, "The Conference Assistant: Combining Context-Awareness with Wearable Computing," *Proc. Symp. Wearable Computers*, pp. 21-28, 1999.

[6] S. Jang and W. Woo, "Ubi-UCAM: A Unified Context-Aware Application Model, " *Context, LNAI* 2680, pp. 178-189, 2003.

[7] H. Kim, Y. Cho and S. Oh, "CAMUS: A Middleware Supporting Context-aware Services for Network-based Robots," *IEEE Workshop on Advanced Robotics and Social Impacts*, 2005.

[8] W. Lee and T. Yang, "Personalizing Information Appliances: a Multi-agent Framework for TV Programme Recommendations," *Expert Systems with Applications* 25, pp. 331-341, 2003.

[9] A. Ranganathan and R. H. Campbell, "A Middleware for Context-Aware Agents in Ubiquitous Computing Environments, " *LNCS* 2672, pp. 143-161, 2003.

[10] K. Romer, "Tracking Real-World Phenomena with Smart Dust, " *LNCS* 2920, pp. 28-43, 2004.

[11] B. Schilit, N. Adams and R. Want, "Context-aware Computing Applications," *Proc. IEEE Workshop on Mobile Computing Systems and Applications*, pp. 85-90, 1994.

[12] R. Want, A. Hopper, V. Falcao and J. Gibbons , "The Active Badge Location System," *ACM Transactions on Information Systems*, 10(1) pp. 91-102, 1992.

[13] S. Yau, F. Karim, Y. Wang, B. Wang and S. Gupta, "Reconfigurable Context-Sensitive Middleware for Pervasive Computing," *IEEE Pervasive Computing*, 1(3), 2002.

Authors

Aekyung Moon

Received a M.S. and Ph D. degree in computer engineering from Yeungnam University, Korea, 1992, and 2000, respectively. In 2000 she joined Electronics and Telecommunication Research Institute, Korea. Currently, she is senior researcher in software robot research Team. Her research interests include distributed systems, context-aware middlewares, sensor networks and recommendation systems.

Minyoung Kim

Received the B.S. and M.S. degrees in the department of computer sicence from Chungbuk University, Cheongju, Korea, in 2004, 2006 respectively. He joined ETRI in 2006. Currently, he is a member in Intelligent Robot Research Division, ETRI. His research interests include networked robot software, the context-awareness, ubiquitous computing, and distributed computing.

Hyoungsun Kim

Received the M.S. degree in Computer Engineering from Kwangwoon University, Seoul, Korea, in 1991 and PhD. degree in Computer Engineering of Daejeon University, Daejeon, Korea 2003. He had worked for Systems Engineering Research Institute (SERI) from 1986 to 1998. He joined Electronics and Telecommunications Research Institute (ETRI) in 1998 and has worked for the software development related to Intelligent Robot. Currently, he is senior researcher in Software Robot Research Team, ETRI. His research interests include context-aware computing, networked robots, software robot, distributed computing, Information security, and distributed database.

Kang-Woo Lee

Received the B.S., M.S. and Ph.D. degrees in the department of Computer Science from Seoul University, Seoul, Korea, in 1991, 1993 and 2000, respectively. He joined ETRI in 2000 and has worked for the system software development related to distributed systems. Currently, he is a member in Intelligent Robot Research Division, ETRI. His research interests include networked robot software, the context-awareness, ubiquitous computing, and distributed computing.

Hyun Kim

Received the B.S., M.S. and Ph.D. degrees in the department of mechanical design and manufacturing from Hanyang University, Seoul, Korea, in 1984, 1987 and 1997, respectively. He had worked for Systems Engineering Research Institute (SERI) from 1990 to 1998. He joined ETRI in 1998 and has worked for the software development related to intelligent systems. Currently, he is a project leader in Intelligent Robot Research Division, ETRI. His research interests include networked robots, the context-awareness and ubiquitous computing, distributed computing and virtual engineering