A Context-Aware Multi-agent Service System for Assistive Home Applications[⋆]

Yong Kim, Yoonsik Uhm, Zion Hwang, Minsoo Lee, Gwanyeon Kim, Ohyoung Song, and Sehyun Park[⋆⋆]

School of Electrical and Electronics Engineering, Chung-Ang University, 221, Heukseok-dong, Dongjak-gu, Seoul 156-756, Korea
{ykim, neocharisma, zhwang, lemins, cityhero}@wm.cau.ac.kr,
{song, shpark}@cau.ac.kr

Abstract. In this paper, we present an Ontology-based Context-aware multi-Agent Service System(OCASS) architecture to provide these context-aware services in a smart home. To model various contexts, we design Ontology which supports to share context knowledge, detect and resolve inconsistency of the knowledge. In addition, we classify context-aware services into three layers - Session, Task, Subtask - to make sure definition of service conflict problems and to solve the problems easily. With our context model and the classification of service conflicts, our system supports autonomic tasks including recognizing and learning user's formal/informal activity pattern, and resolving conflicts between services in different situations of users invisibly.

Keywords: context-aware system, multi-agent architectures, Ontology, ubiquitous computing, pervasive computing, smart home.

1 Introduction

A lot of work has been done in trying to make applications in ubiquitous computing environments context-aware so that they can adapt to different situations and be more receptive to users' needs [1] [2] [3]. A number of context-aware systems, which provide users with relevant services and information based their situational conditions[1], have been developed to demonstrate the usefulness of various contexts. However, most of the systems for an interaction between context-aware services categorize them in respect of not their functions nor goals but controlling devices, so they have not been adequately considered conflict problems between services. Also, most of studies have not totally been considered the recognition of user's informal activity pattern, and the context pattern learning, including a solution of service conflict problems in respect of intelligence in a smart

[⋆] This research was supported by the MIC(Ministry of Information and Communication), Korea, under the Chung-Ang University HNRC(Home Network Research Center)-ITRC support program supervised by the IITA(Institute of Information Technology Assessment).

[⋆⋆] The corresponding author.

home. Moreover, context-aware services have not been widely available to everyday users and developing such systems is still a complex and time-consuming task.

A context-aware system is required to perform the following tasks: 1) to sense, reason and mine about the various contexts including user's formal/informal activity pattern, 2) to share the contextual information between agents or systems that are located in an open, dynamic, and distributed environment through providing the semantics of the context, 3) to manage service units to solve conflict problems between services when the system provides users with relevant multi-services by making a decision which actuators are in accordance with the current context and by scheduling a collaboration of actuators, 4)to acquire additional contexts on demand whenever to require more contexts than the first contexts.

In this paper, we propose an Ontology-based Context-aware multi-Agent Service System(OCASS) architecture that performs intelligent context-aware services to satisfy the requirements. To evaluate the effectiveness of our architecture, we also created a testbed and comprehensive scenarios that resolve conflicts between services in different situations. The rest of this paper is organized as follows. Section 2 gives related works about context-aware systems. Section 3 presents definition of conflict problems between services in a smart space. In section 4, we describe the OCASS architecture with our OWL based ontology models and the interaction of its agents. Section 5 presents our implementation testbed. Finally, we state our future work and conclusion in section 6.

2 Related Works

A number of context-aware systems have been developed to demonstrate the usefulness of context-aware technology. Recent research work of context-aware systems has focused on providing infrastructure support for context aware systems. In the Context Fabric [2] infrastructure, Hong et al. took a database-oriented approach to provide context abstraction by defining a Context Specification Language; and a set of core services. However, the design of a proprietary context specification language may lead to the lack of a common model. In the CoBrA [4] project, Chen et al. proposed an agent-oriented infrastructure for context representation, sharing knowledge and user's privacy control. They developed common ontology for developing context expression ontology to independent system and common policy language, shared context model to providing all kind of devices, services, and agents. Service-Oriented Context-Aware Middleware (SOCAM) architecture [5] provides efficient support for acquiring, discovering, interpreting and accessing various contexts to build context-aware services. SOCAM proposes a formal context model based on ontology using WebOntology Language to address issues of semantic representation, context reasoning, context classification and dependency. However, these researches use only the first contexts that are sensed by context providers[6] at first and that can be cause to provide services, so they may lead to lack of more intelligent services for recent users who request extremely various needs in a smart space.

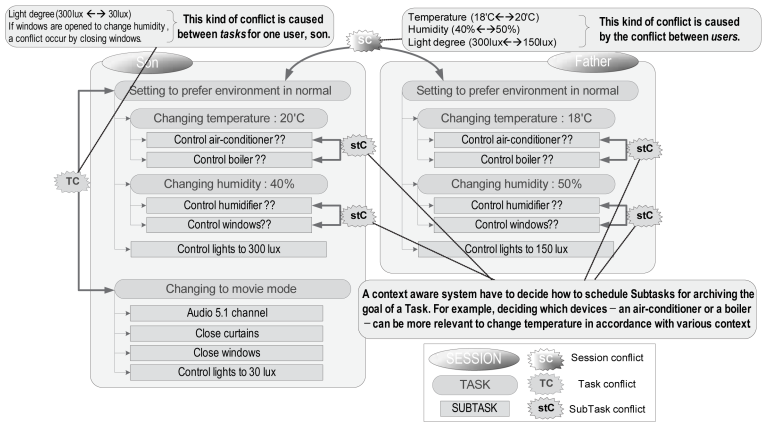

Light degree(300lux ←→ 30lux)
If windows are opened to change humidity, a conflict occur by closing windows.
This kind of conflict is caused between *tasks* for one user, son.
Temperature (18'C←→20'C)
Humidity (40%←→50%)
Light degree (300lux←→150lux)
This kind of conflict is caused by the conflict between *users*.
SC
Son
Father
Setting to prefer environment in normal
Changing temperature : 20'C
Control air-conditioner??
Control boiler??
stC
TC
Changing humidity : 40%
Control humidifier??
Control windows??
Control lights to 300 lux
Changing to movie mode
Audio 5.1 channel
Close curtains
Close windows
Control lights to 30 lux
Changing temperature : 18'C
Changing humidity : 50%
Control lights to 150 lux
A context aware system have to decide how to schedule Subtasks for archiving the goal of a Task. For example, deciding which devices – an air-conditioner or a boiler – can be more relevant to change temperature in accordance with various context
SESSION
TASK
SUBTASK
Session conflict
Task conflict
SubTask conflict

Fig. 1. A classification of context-aware services and service conflicts in an example

3 Context-Aware Services and Service Conflicts

Most of studies for interaction between services categorize the services in respect of controlling devices. Interaction between services can be more complex in especially ubiquitous environments. Context-aware services are intelligent, flexible, multiple, and dynamic services which are depending on user's situational conditions. In smart home environment, which there are various and complex requirements of home members, it is important that context-aware system manages service units to solve conflict problems between services in these aspects.

We first classify context-aware services into three layers - *Session*, *Task*, *Subtask* - to make sure the definition of conflict problems between services. *Session* is a group of Tasks. One *Session* is opened(active) when a user comes in a service-available location - including a virtual location, in which a user is located when he connects from outdoor to the home network for being provided services - and is closed(retired) when the user goes out the location if there are no remain *Task*s. The *Session* that exists per one user and one location may be able to transmit its context information to the other one. A *Task* is a service unit to contain a goal. To archive the goal, the *Task* consists of one or more *Subtask*s that are the smallest service units like controlling a device or setting up its properties. A user can be provided automatically multiple *Task*s in a *Session.* These concepts of classified service layers can provide the base for the interaction between services. Especially, to solve conflict problems between services we classified the problems with our definition of service layers as follow.

Session Conflict. In one location, multi-user can be provided same or similar Tasks simultaneously. Because the Tasks consist of Subtasks to target controlling the same device or changing service environment, the conflict between users can be occurred. We define this conflict as a *Session conflict.*

Task Conflict. A device can receive different commands at the same time when two or more tasks need to be performed for one user. This kind of conflict

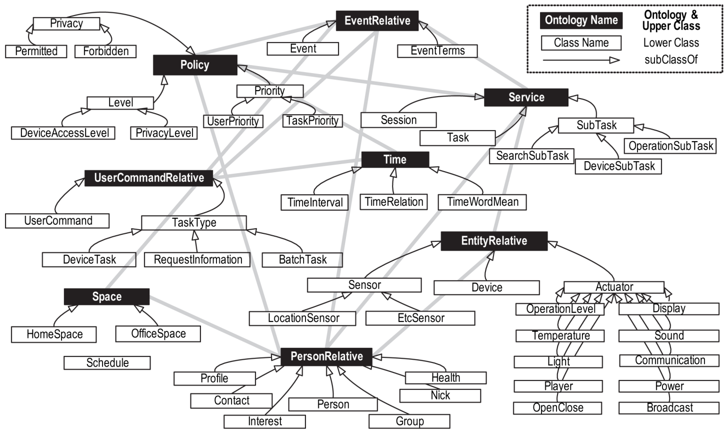


Fig. 2. Context model in a smart home (HomeNet Ontologies[7])

is not the problem between users but Tasks. We define this kind of conflict as a *Task conflict*.

Subtask Conflict. Subtasks that are organized to achieve a goal of a task have to be ordered with contextual information. At this process, it is possible to conflict between Subtasks because two Subtasks intend to control one device at the same time or to control similar devices that change same environment. We define this kind of conflict as a *Subtask conflict*. A *Session conflict* and a *Task conflict* are finally caused by one or more *Subtask conflict*s, but a *Subtask conflict* is not based on them.

Figure 1 shows context aware services newly defined by us and conflict problems with an example. We believe that this new definition will help the conflict problems between services be addressed. Context-aware systems can find or solve service conflicts which can be occurred frequently in the ubiquitous environment easily by modeling ontologies and making rules with our definition.

4 OCASS: Ontology-Based Context-Aware Multi-Agent Service System

4.1 Context Model – HomeNet Ontologies

Context-aware systems need recognizable contextual information and should be able to acquire more information or knowledge through inferencing the relations among them. Moreover, the system should support a context model to be able to share those relations with other systems which are in various connected domains. We have developed the context model(*HomeNet Ontologies*[7]) that is based on ontology expressed using OWL.

The set of ontologies consists of vocabularies to express concepts that are associated with following like person, device, space, time, service, user's activity,

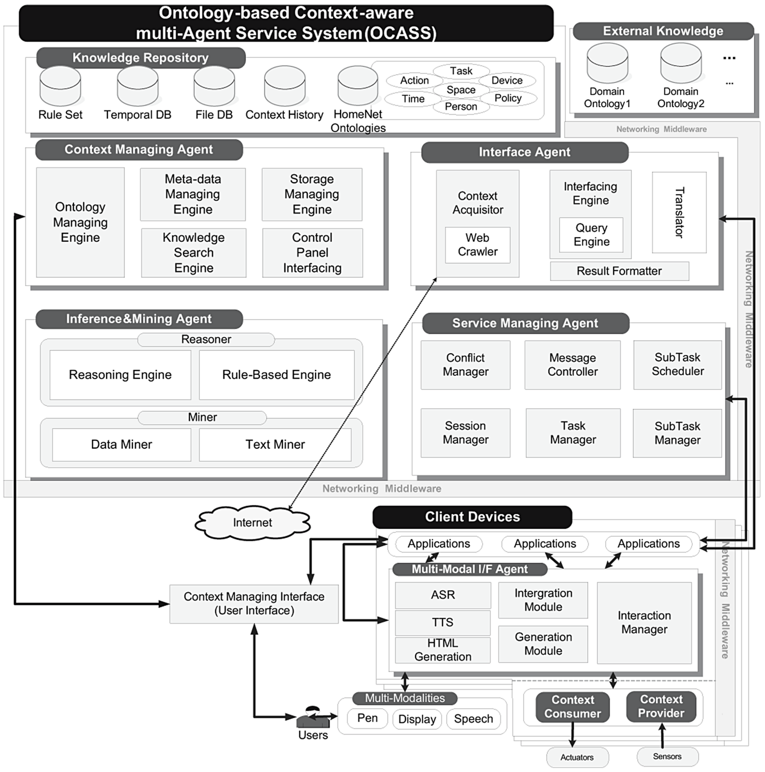


Fig. 3. OCASS architecture and client devices

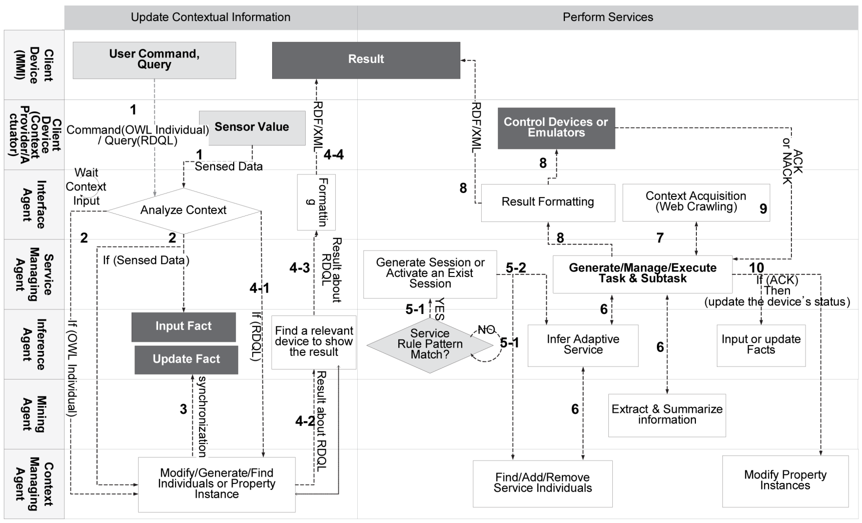
event, and policy. The ontologies are grouped into eight distinctive ontology documents. Figure 2 shows the ontologies, their properties, and their associated relations.

4.2 OCASS Architecture

Our OCASS(an Ontology-based Context-aware multi-Agent Service System) architecture is designed to satisfy the requirements that are specified in Section 1 above for a smart home. Figure 3 shows the system diagram of OCASS design. OCASS consists of several big components - *Knowledge Repository*, *Context Managing Agent*, *Interface Agent*, *Inference & Mining Agent* and *Service Managing Agent*.

* **Knowledge Repository.** The *Knowledge Repository* provides various contexts and information, structured and unstructured data, rules. Also, it stores static and dynamic patterns as knowledge. Using RuleML and JESS[8] script language, *Inference Engine* accesses and stores, deletes its rules.

* **Context Managing Agent.** Context-aware system can efficiently manages number of contexts for supporting context-aware services well. That is, it has


Update Contextual Information
Perform Services
Client Device (MMI)
Client Device (Context Provider/Actuator)
Interface Agent
Service Managing Agent
Inference Agent
Mining Agent
Context Managing Agent
User Command, Query
Result
Command(OWL Individual) / Query(RDQL)
Sensor Value
Sensed Data
Wait Context Input
Analyze Context
If (Sensed Data)
Input Fact
Update Fact
synchronization
If (OWL Individual)
If (RDQL)
Modify/Generate/Find Individuals or Property Instance
Find a relevant device to show the result
Result about RDQL
Formatting
RDF/XML
Control Devices or Emulators
ACK or NACK
Result Formatting
Context Acquisition (Web Crawling)
Generate Session or Activate an Exist Session
Generate/Manage/Execute Task & Subtask
If (ACK) Then (update the device's status)
Service Rule Pattern Match?
YES
NO
Infer Adaptive Service
Input or update Facts
Extract & Summarize information
Find/Add/Remove Service Individuals
Modify Property Instances
1
2
3
4-1
4-2
4-3
4-4
5-1
5-2
6
7
8
9
10

1. First of all, MMI(Multi-Modal Interaction framework) or Context Provider recognizes and collects some contextual data from sensors or users.
2. After Interface Agent translates, analyzes and formats the contexts from Client Deivces into RDF/XML form, Inference Agent updates facts - a collection of information nuggets - to reason the situational conditions.
3. During Inference Agent updates the facts, Context Managing Agent also finds, modifies and generates OWL individuals or property instances for synchronization between the facts and the OWL individuals.
4. If the format of the contextual data from Client Devices is RDQL, Interface Agent forwards the RDQL to Context Managing Agent and Context Managing Agent gives a result of the query. After the result is formatted to RDF/XML, Interface Agent transmits the information to a relevant Client Device that is found by rule-based reasoning with the last facts.
5. If some sensed information matches service rule patterns that a user has written in RuleSet or that is automatically made by learning mechanisms, Inference Agent orders Service Managing Agent to generate a Session or to activate an exist Session. At the same time Inference Agent infers adaptive services based on several facts (for example, user id, user's current location, default tasks, proper actuators, etc.).
6. Then, Service Managing Agent manages multi service layers. At this time, it tries to solve several conflict problems between services that we defined in the above. For instance, it performs scheduling Subtasks so that it prevents service conflicts like turning on a boiler and an air-condition simultaneously if a user didn't order purposely two devices to turn on. Services provided by OCASS can be both controlling devices and extrating or summarizing useful information.
7. If there are more contexts needed for supporting intelligent services, Service Managing Agent requests Context Acquisitor to acquire needed contexts from Context Providers or internet.
8. Finally, Interface Agent makes a proper format of actuator control commands that the actuator can understand and can be operated.
9. In addition, our system estimates whether it executes right services or not for self-learning and for determining to support other available services. For example, if main light in a room is out of order when a service includes the operation of it, the system tries to turn on an alternate device through reasoning of Inference Agent.
10. If the services correspond to an user's intend, OCASS updates the current status that is changed by the services.

Fig. 4. Interaction of OCASS agents

to perform several functions: 1) to find the contexts that devices require and to support them in some available format, 2) to validate and to design the context ontology class, and 3) to store and to manage information related with users, devices, space, location, and etc. *Context Managing Agent* is an agent to perform these three functions in connection with *Knowledge Repository*. We implemented a part of this *Context Managing Agent* related OWL using Jena[9].

* **Interface Agent.** *Interface Agent* performs representing of contextual data from MMI(Multimodal Interaction framework)[10] or sensor agents, creating queries, transforming and transmitting the result about the queries. *Context Acquisitor* collects and selects contexts needed for adaptive service from web or *Context Providers* (MMI or Sensor Agents).

* **Inference & Mining Agent.** *Reasoner* is responsible for checking class consistency and implied relationship and for asserting inter-ontology relations when the system needs to integrate domain specific ontologies. It is implemented using a rule-based reasoning engine. This agent deduces current situation from relations between classes and individuals of context model. We implemented the core of *Inference Engine* using modified JESS[8].

Miner generates contexts and patterns to fit a user's interest and design. *Data Miner* generates or modifies OWL individuals from a useful tendency with statistic information in *Context History*. These individuals are used for *Reasoner* to decide proper services with environmental information for users. We implemented this learning mechanism of *Data Miner* using HHMM(Hierarchical Hidden Markov Model)[11]. *Text Miner* has a function of summarizing and drawing, clustering with unstructured data in a text form. As *Text Miner*, we use the core modules of IN2 Platform [12].

* **Service Managing Agent.** *Service Managing Agent* is an agent that administrates various services. It has three modules to manage three service layers(Session, Task and Subtask), to find service conflicts, and to solve the conflict problems. Especially, *Subtask Scheduler* schedules multiple Subtasks so that solves Subtask conflict problems.

Figure 4 shows how the agents in OCASS architecture interact with each other included MMI(Multi-Modal Interaction framework)[10], Context Providers, and Actuators. In fact, the agents parallelly make their operations caused by ondemand services.

5 Implematation Testbed

5.1 Testbed Setup

In order to test our OCASS architecture for home networks, we created the testbed shown in figure 5. For OCASS gateway host, we used a standard PC (Intel Pentium4 2 GHz) with a JVM from the Java 2 Enterprise Edition. Agents open a remote-method-invocation (RMI) connection to send requests to the appropriate service agents.

For location tracking, we deploy a location tracking system based on IR sensors, pressure sensors and image sensors. The image based location tracking system consists of CCD cameras and a pan-tilt-zoom (PTZ) camera with feature fusion-based people tracking algorithm [13]. Installing the sensors, labeling the coordinates, and manually adjusting sensitivity of sensors, entering this data into our software took about 168 person-hours.

5.2 Service Scenario

We demonstrated our approach with comprehensive scenarios. We trained our system recognizing and learning user's formal/informal activity patterns on *Knowledge Repository* in several weeks.

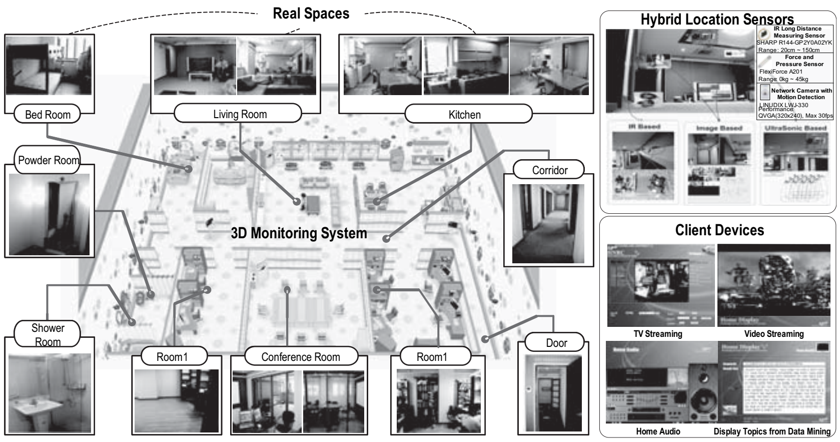


Fig. 5. Our Implementation Testbed in Home Network Research Center

* **Service Scenario 1.** *The context-aware service based on the learned user profiles* (fig. 6(a)): The father enters the living room at 7 PM and sits on the sofa. OCASS reasons the father is at home after his work. OCASS evaluates the dynamic context to provide the father with more adaptive and optimal service environment. OCASS turn on the TV on the display 1 and tunes the preferred channel with the program guide of the channel on the display 2. OCASS also performs the web crawling and show the stock prices of his interests on the display 3.

* **Service Scenario 2.** *The intelligent service resolving conflicts between two user services in different situations* (fig. 6(b)): The son laid himself on the couch in the living room at 7 PM. OCASS configures the living room condition as *the flu mode* by warming up the air and turning on the humidifier (Scene 1). When the father enters the living room as the scenario 1, and then the *session conflict* occurs. As the service priority has given to the son, OCASS do not change the living room environment for the father (Scene 2). When the father walks through the corridor and enters his room, location sensors indicate the father's new location to OCASS. OCASS reasons that the father is now capable of receiving the preferred services. OCASS adjusts the room condition according to his preferences by turning on the lights and configuring the temperature of the air conditioner (Scene 3).

5.3 Lessons Learned

In this section, we discuss several findings from the OCASS deployments. In a similar fashion to [14] we share the participants' general feedback on the home network environment with OCASS, where they used it in their homes, and how they interacted with it. We also discuss the OCASS's impact on the lives of the home network service members. The results of our deployments suggest that

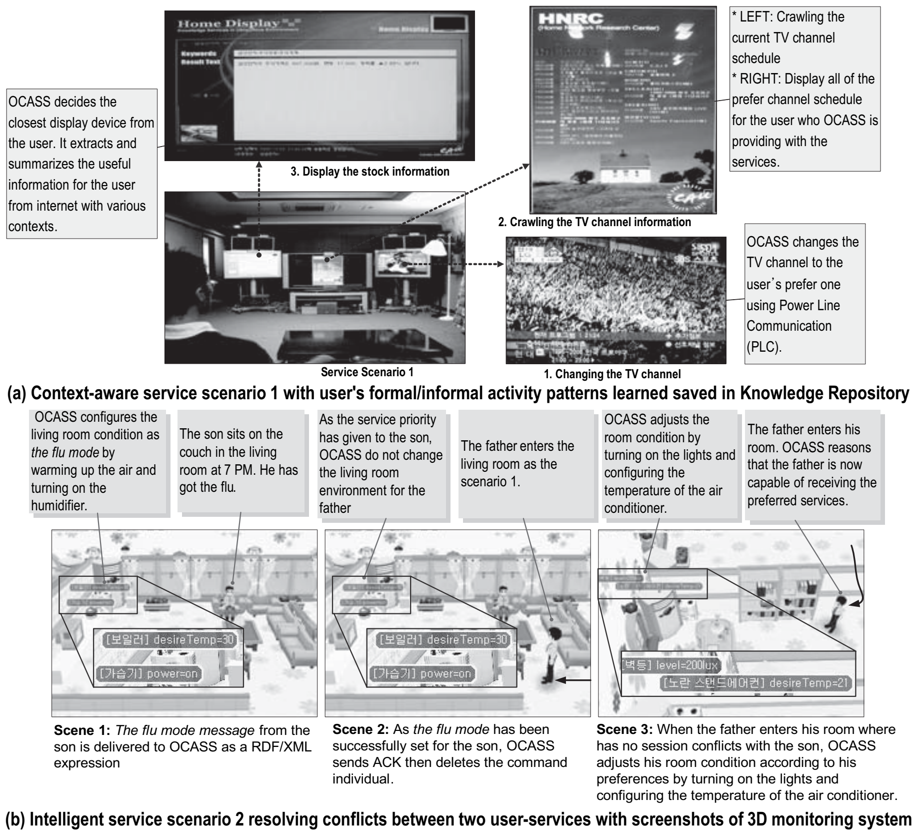

OCASS decides the closest display device from the user. It extracts and summarizes the useful information for the user from internet with various contexts.
Home Display
Keywords
Result Text
3. Display the stock information
HNRC
(Home Network Research Center)
* LEFT: Crawling the current TV channel schedule
* RIGHT: Display all of the prefer channel schedule for the user who OCASS is providing with the services.
2. Crawling the TV channel information
OCASS changes the TV channel to the user's prefer one using Power Line Communication (PLC).
Service Scenario 1
1. Changing the TV channel
(a) Context-aware service scenario 1 with user's formal/informal activity patterns learned saved in Knowledge Repository
OCASS configures the living room condition as the flu mode by warming up the air and turning on the humidifier.
The son sits on the couch in the living room at 7 PM. He has got the flu.
As the service priority has given to the son, OCASS do not change the living room environment for the father
The father enters the living room as the scenario 1.
OCASS adjusts the room condition by turning on the lights and configuring the temperature of the air conditioner.
The father enters his room. OCASS reasons that the father is now capable of receiving the preferred services.
[보일러] desireTemp=30
[가습기] power=on
[보일러] desireTemp=30
[가습기] power=on
[벽등] level=200lux
[노란 스탠드에어컨] desireTemp=21
Scene 1: The flu mode message from the son is delivered to OCASS as a RDF/XML expression
Scene 2: As the flu mode has been successfully set for the son, OCASS sends ACK then deletes the command individual.
Scene 3: When the father enters his room where has no session conflicts with the son, OCASS adjusts his room condition according to his preferences by turning on the lights and configuring the temperature of the air conditioner.
(b) Intelligent service scenario 2 resolving conflicts between two user-services with screenshots of 3D monitoring system

Fig. 6. Service Scenario

OCASS deployments can be an effective tool in helping home network members with the tasks of service and information sharing. The intelligent services with OCASS was well received both by the home network members and the elders. In all cases, the home network members who participated said that they would use such a service if it were given to them, and in most cases, they would deploy the system if it were commercially available and affordable. Participants thought that the display was pleasing and blended in nicely with their home appliances, though some complained that the services were "somewhat frightening." They tended to receive the services in often used, common areas of their homes. For example, the services were provided in the family/TV room, living room, home office, or kitchen. When asked about what the elders thought of these deployments of the services, most thought them to be acceptable. Based on our OCASS experience, we think a certain set of service strategies might be generally applicable to the design of smart home environment.

6 Conclusion

In this paper, we describe an Ontology-based Context-aware multi-Agent Service System architecture to provide context-aware services in a smart home. The development of the Ontology-based context model - HomeNet Ontologies - and OCASS architecture are still at an early stage of research. We believe that our system architecture, with the context model, should support tasks including recognizing user's informal activity pattern, learning context patterns, and resolving conflict problems between services. We will continue to develop our system to be able to interact with other agents or systems in an extended smart space. We will also work to develop for supporting a detail resolution of service conflict problems including potential services in the space. In addition, we plan to prototype adjusted context model considering user activity and emotion, activity pattern.

References

1. Dey, A.K.: Providing architectural support for building context-aware applications. Human-Computer Interaction (HCI) Journal **16** (2001)
2. Hong, I., J.: An infrastructure approach to context-aware computing. HCI (**16**)
3. Pascoe, J., Ryan, N., Morse, D.R.: Issues in developing context-aware computing. In: HUC. (1999) 208–221
4. Chen H and Finin T: An ontology for a context aware pervasive computing environment ijcai workshop on ontologies and distributed systems. (In: Acapulco MX 2003)
5. Tao Gu: A service-oriented middleware for building contex-aware services. (In: Journal of Network and Computer Applications 28)
6. Biegel, G., Cahill, V.: A framework for developing mobile, context-aware applications. In: PerCom. (2004) 361–365
7. Yong Kim, EunYoung Hwang, and Sehyun Park: A context aware multiagent system architecture for a smart home. (In: Net-Con'2005)
8. JESS, the Rule Engine for Java Platform: (http://herzberg.ca.sandia.gov/jess/)
9. Jena2, A Semantic Web Framework: (http://www.hpl.hp.com/semweb/jena2.htm)
10. Multimodal Interaction Framework: (http://www.w3.org/tr/mmi-framework)
11. Fine, S., Singer, Y., Tishby, N.: The hierarchical hidden markov model: Analysis and applications. Machine Learning **32** (1998) 41–62
12. IN2 Platform,Saltlux: (http://www.in2web.co.kr)
13. Lee, J., Kim, S., Kim, D., Shin, J., Paik, J.K.: Feature fusion-based multiple people tracking. In: PCM (1). (2005) 843–853
14. Consolvo, S., Roessler, P., Shelton, B.E.: The carenet display: Lessons learned from an in home evaluation of an ambient display. In Davies, N., Mynatt, E.D., Siio, I., eds.: Ubicomp. Volume 3205 of Lecture Notes in Computer Science., Springer (2004) 1–17