Framework and Rule-based Language for Facilitating Context-aware Computing using Information Appliances

Kouji Nishigaki[1], Keiichi Yasumoto[1], Naoki Shibata[2], Minoru Ito[1] and Teruo Higashino[3]
[1] Graduate School of Information Science, Nara Institute of Science and Technology
Ikoma, Nara 630-0192, Japan {koji-ni, yasumoto, ito}@is.naist.jp
[2] Department of Information Processing and Management, Shiga University
Hikone, Shiga 522-8522, Japan shibata@biwako.shiga-u.ac.jp
[3] Graduate School of Information Science and Technology, Osaka University
Suita, Osaka 565-0871, Japan higashino@ist.osaka-u.ac.jp

Abstract

Recently, context-aware computing with information appliances is the topic of many research efforts. In order to realize context-aware systems, it is necessary to describe rules, each of which consists of (1) transition condition for identifying a specific context, and (2) actions to be executed when the condition holds. However, in an ordinary home without IT specialists, it would be too complicated for each user to specify feasible rules by choosing an appropriate combination of sensors and actions, since possible combinations are so many and he/she may have even no idea on functionalities of sensors or devices. Also, multiple users may want to control the same device at the same time in different ways. In this paper, we propose a framework and a rule-based language for simply and intuitively specifying feasible rules for context-aware control of information appliances. Our framework includes mechanisms for retrieving sensors/devices to specify a condition of a rule and for detecting a conflict over multiple rules. Our prototype implementation on a PC achieves practically sufficient performance for these operations.

1. Introduction

Recently, context-aware computing [1] using information appliances is one of the most important research topics. In context-aware computing systems, devices are automatically controlled based on the current context obtained from various sensors such as user's positions, room temperature and so on. In order to make context-aware systems work properly, we need to identify the current context of the environment (including users) and to retrieve the rules which can be executed on the context. As techniques to discover specific devices in an ubiquitous environment, UPnP [2] and Jini [3] have been standardized. Even with these techniques, in order to make various devices work cooperatively based on the context, a scenario of how user(s) want to control those devices, must be specified in advance. In a scenario, rules consisting of sensor conditions and device actions are described. To specify each rule correctly, it is required to carefully choose a set of sensors and devices which the user wants to control. So, users have to be familiar with their functionalities. However, it would be too difficult for users in an ordinary home to describe a feasible scenario for making the system work in their expected ways. Also, at a home environment, multiple users may want to control the same device simultaneously in different ways.

In this paper, we propose a framework for allowing everyone to easily describe scenarios for context-aware computing systems including various information appliances and sensors. Our framework facilitates (1) personalization of devices, (2) intuitive specification of rules, and (3) consistency check and conflict detection in multiple rules.

For the above purposes, first, we have defined a language called CADEL (Context-Aware rule Description Language) to specify rules. Since CADEL has similar syntax and semantics to natural languages, ordinary home users can specify rules intuitively. In CADEL, each user can define new words (e.g., hot-and-stuffy) to indicate specific contexts sensed from multiple sensors.

Secondly, our framework provides a guidance function to users during rule description, with which users can retrieve the nearby sensors and devices through GUI and obtain the information such as the allowable actions of a device and the value of a sensor. Consequently, users can easily specify a rule using the obtained information.

Thirdly, our framework provides a mechanism to automatically detect a conflict among multiple rules, which happens when the conditions of multiple rules hold at the same

time and they perform different actions to the same device. In order to avoid inconsistency due to such a conflict, we specify the priority among those conflicting rules. When a new rule is registered, our framework checks whether the rule conflicts with existing rules. If it conflicts, our framework prompts users to specify the priority among the rules. Users can attach a specific context to the priority so that the priority works only on the context.

We have implemented a prototype system of the proposed framework on a PC using UPnP, and evaluated the performance of the system in terms of response time of sensor/device retrieval and conflict detection over multiple rules. Through experiments, we have confirmed that our prototype implementation achieves practically enough performance for executing these operations.

The remainder of the paper is constructed as follows. Sect. 2 describes related work. In Sect. 3, we identify problems and give basic ideas. In Sect. 4, we present the proposed framework with its application to an ordinary home environment. In Sect. 5, we give experimental results on the performance of our prototype implementation. Finally, in Sect. 6, we conclude our paper.

2. Related Work

Among many literatures on context-aware computing, personalization of devices and context-awareness are most important concepts to be achieved.

[4] argues that device personalization will be especially important to make our daily life more convenient as various information appliances are available everywhere in the future. In order to personalize devices, those devices need to know each user's preference as well as his/her private information. However, such private information must be protected from other users. Accordingly, [4] proposes a home server which can be carried by each user to personalize nearby devices. The proposed home server is capable of personalizing various devices, controlling those devices according to users' preferences and situations. The server also has a mechanism to discover nearby devices in various locations (e.g., at home, at stations, in cars, on streets, and so on). However, this research assumes that each device is used by a single user at one time, and does not deal with the case that multiple users compete the same device for its use.

Also, there are several techniques to achieve context-awareness in ubiquitous computing systems. In [5], a location sensitive university campus guiding system has been implemented and evaluated. [6] proposes a framework for easy development of context-aware applications. In [7], a framework for developing mobile context-aware applications has been proposed. In the above existing frameworks, contexts are represented by multiple output data from different kinds of sensors. The frameworks help developers to easily implement context-awareness in the system with APIs to identify the current context by sensing required information by sensors. The frameworks also provide a mechanism to assign a relationship among sensors, actuators and application components. As described above, these frameworks focus mainly on facilitating development of context-aware systems. Our proposed framework is different from these existing ones, since our framework allows ordinary home users to easily describe scenarios for controlling the information appliances in their expected ways.

3. Basic Ideas of Proposed Framework

In this paper, we propose a framework for context-aware systems which handles both of (1) device personalization, and (2) conflict avoidance control to each device.

3.1. Problems in Typical Context-aware Control of Information Appliances at Home

Let us suppose context-aware control of information appliances in a living room at an ordinary home, consisting of myself (e.g., Tom), and parents (e.g., Alan and Emily). Also suppose that there are a stereo system, a flat-panel TV, a video recorder, a fluorescent light, floor lamps, and an air conditioner in the living room.

Typical context-aware control is as follows: When Tom comes to the living room, the floor lamps are turned on with half-lighted (e.g., indirect lighting by floor lamps), the stereo system starts to play his favorite music (e.g., jazz) with appropriate volume, the air conditioner regulates room temperature and humidity to his comfortable degrees (e.g., 25C and 60%). After a while, the TV is automatically turned on since a TV program on air includes a keyword which he is interested in. Then a pop-up menu is shown on the TV to let him decide switching off the stereo and watching the program or turning on the video recorder to record it.

It is possible to implement a system like the above with the existing personalization methods such as [4] and sensors which detect the information required to identify the current context (existence/location of the user, the current temperature and humidity, the current time, and the TV programs on air). In order to control devices appropriately, we need to describe rules indicating target devices, actions to be executed, and conditions to hold when executing those actions. It would be difficult for ordinary home users to describe such complicated rules.

On the other hand, it is also difficult to personalize devices when multiple users share the same space like in an ordinary home. For example, let us suppose that Alan (Tom's father) has got home from work while Tom is listening to jazz music in the above example. When Alan has registered a keyword "baseball game" and a game is currently on air,

the system identifies Alan and turns on the TV. In that case, the TV sound might interfere with that of the stereo, resulting in unsatisfaction of the both users. Also, when Alan has preferences for the room lighting and temperature which greatly differ from those of Tom's, the system will have a trouble in deciding whose preference should be adopted. Moreover, when Emily has got home from shopping while Alan is watching a baseball game and her favorite movie is on air, there will be a conflict with the TV between Alan and Emily. In an actual situation, conflict with the same device is likely to be solved by a negotiation among the conflicting users and by deciding a priority order with a certain policy. Existing context-aware systems, however, suppose that each device is used by one user at one time. They do not have mechanisms to detect conflicts or to support users to solve the conflicts.

If a priority order among users' preferences is defined in advance, we should be able to control the whole system appropriately. Here, we assume that Tom, Alan and Emily have the following preferences.

- Tom (myself): When I'm in the living room in evening, I want to listen to jazz music with the stereo (s1 in Fig. 1). While listening, I want to make the room half-lighting with the floor lamps. When the room becomes hot and stuffy (e.g., temperature and humidity become higher than 26 C and 65%, respectively), I want the air-conditioner to be turned on, with 25 C and 60%.
- Alan (father): When I'm in the living room and a baseball game is on air, I want to watch the game on the TV (t2 in Fig. 1). If it is impossible to use the TV, I want to record the game with the video recorder. When the room becomes hot and stuffy (e.g., temperature and humidity become higher than 25 C and 60%, respectively), I want the air-conditioner to be turned on, with 24C and 55%.
- Emily (mother) : When I'm in the living room and my favorite movie is on air, I want to watch the movie on the TV (t3 in Fig. 1). While watching a movie, I want play back the sound of the movie through the stereo (s3). While watching a movie, I want to make the room bright with the fluorescent light (l3). When the room becomes hot and stuffy (temperature and humidity become higher than 29 C and 75%, respectively), I want the air-conditioner to be turned on, with 27 C and 65%.

For the above preferences, we can control the whole system as follows. Time-cart of this control scenario is shown in Fig. 1.

In Fig.1, we assume that Alan has higher priority in the living room than Tom in the context that Alan got home from work. So, Tom has to switch off the stereo or listen to the music with the headphone. Similarly, we assume that Emily has the highest priority in the context that she got

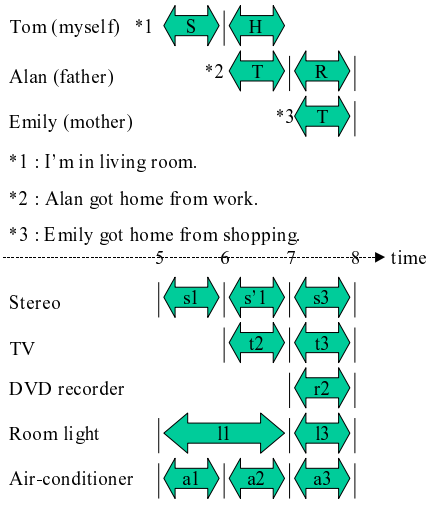

Figure 1. A Control Scenario for Conflicting Requirements

home from shopping. So, in this case, the stereo and the TV will play back the movie which Emily wants to watch. After that, the video recorder will be automatically turned on and record the baseball game.

Like the above, we can control the whole system automatically if the priority is predefined among the users in advance.

3.2. Basic Ideas of Our Framework

For the problems in the previous section, we adopt the following ideas.

Avoidance of Device Conflict In the proposed framework, we basically solve the device conflict by defining a priority among conflicting users. However, users might want to change the priority depending on situations. So, we adopt a policy that users can define multiple different priorities for the same device and attach a context to each of them. For example, to the TV, our framework can let Alan have a higher priority than Tom in the context that Alan got home from work, and at the same time it can give a higher priority to Tom in the context that today is Tom's birthday.

Intuitive Rule Description In order to allow ordinary home users to easily and intuitively describe rules for context-aware systems, our proposed framework provides (i) a lookup service of sensors and devices, and (ii) personalization of contexts.

In existing context-aware systems, users must be familiar with functionalities of sensors and devices as well as their locations, and specify precise values (or ranges) for

sensors and action names for devices in each rule. Unlike the existing systems, our framework provides a lookup service for sensors and devices, which allows users to browse them (e.g., visually or by voice) and to see their functionalities, current values of sensors and so on. Each user can reach the target sensors and devices quickly by giving conditions such as a retrieval range of device locations (e.g., within the current room, current floor, or so on), category of sensors and devices (e.g., concerning with room temperature), and so on.

Also, it is complicated to specify a condition in each rule as a compound context which is typically represented by a logical conjunction of inequalities for the values derived from multiple different sensors. Our framework provides a facility to define each compound context as a simple word. For example, as in the example of Sect. 3.1, users can define new words such as "half-lighting" and "hot and stuffy" for representing a value to be set to a device and a value range(s) of a sensor(s) for representing (part of) a context, respectively. The advantages of this facility are as follows: (a) each user can easily describe rules for other devices with the predefined words, (b) each user can define and reproduce a favorite environment with a sensory word, and (c) system designers can adopt various user interfaces such as voice recognition since rules can be specified with simple words.

4. Proposed Framework

In order to achieve the facilities explained in Sect. 3.2, we define a language called CADEL (Context-Aware rule DEfinition Language), and propose a framework with user interfaces to easily produce rules in CADEL, a mechanism to automatically detect inconsistencies and conflicts of rules, and a mechanism for controlling devices based on rules.

4.1. Overview

Target environment As shown in Fig. 2, our framework facilitates development of context-aware computing systems consisting of users with IDs (e.g., RFID tags), interface devices, a home server(s), sensors, and information appliances with actuators. We assume that these components are connected with each other through a wired or wireless physical network.

In the system, users behave in two ways. First, users describe rules for personalizing devices through interface devices. Secondly, users generate contexts such as IDs and locations and receive context-aware services according to the contexts and rules they registered. As interface devices, we suppose input devices such as microphones, touch panels,

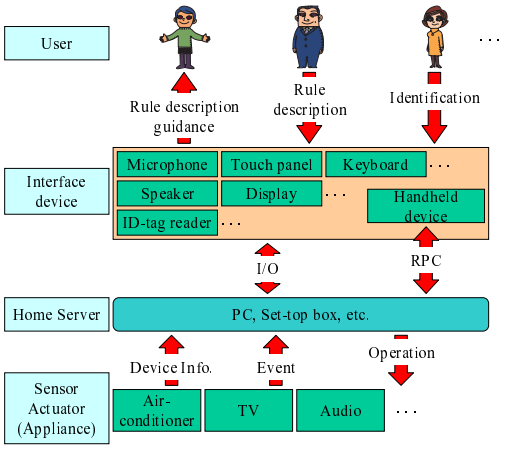

Figure 2. Environment of Our Framework

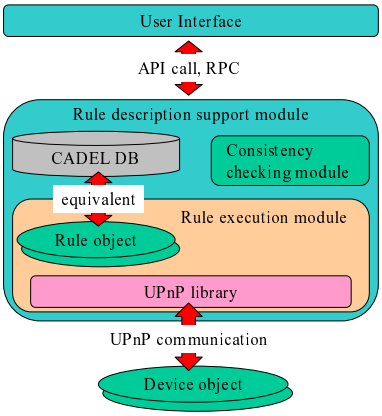

Figure 3. Structure of Our Framework

keyboards and mice, and output devices such as speakers and displays. Portable computing devices such as PDAs and cell phones can also be used as interface devices. We suppose that information appliances have capability of communication with a home server with standard protocols such as UPnP. Finally, we suppose that most functionalities of the proposed framework are implemented in a home server(s). Any PC or set-top box can be a home server.

Structure of our framework As shown in Fig. 3, we compose our framework of the following modules: (1) rule execution module, (2) rule database, (3) rule description support module, (4) consistency checking module, and (5) communication interface module.

The rule description support module allows users to easily describe rules by communicating with interface devices and collecting information from sensors and devices through the communication interface module. Here, we suppose that the UPnP library is used as the communication in-

terface module. Rules described by the rule description support module are represented as CADEL descriptions and stored in the rule database.

The consistency check module checks whether a new rule is consistent or not and whether the new rule conflicts with other rules registered by other users. The rule execution module executes CADEL descriptions. The rule execution module does not executes rules by interpreting CADEL descriptions, but in the rule execution module, a CADEL description is expressed as equivalent a "rule object", by executing rule objects. It receives events from external components in Fig. 2 and issues commands to devices through the communication interface module. Here, we use the UPnP library to retrieve sensors and actuators, to obtain data from the sensors, and to interact with actuators.

4.2. CADEL

We have defined a rule description language called CADEL (Context-Aware rule Definition Language). In CADEL, each rule is described by a tuple of a condition and an action to a device. In CADEL, we have adopted syntax similar to natural languages, aiming at intuitive rule description by ordinary home users. Although we only describe English-based version of CADEL in this paper, different versions of CADEL based on any other languages can be defined. Users can use their mother language based CADEL to describe rules. The syntax of CADEL in BNF notation is shown in Table 1.

In CADEL, we can describe the following rules, for example.

(1) If humidity is higher than 80 percent and temperature is higher than 28 degrees, turn on the air conditioner with 25 degrees of temperature setting.

(2) After evening, if someone returns home and the hall is dark, turn on the light at the hall.

(3) At night, if entrance door is unlocked for 1 hour, turn on the alarm.

In CADEL, users can define new words that can be used in conditions and/or in device configurations of rules by `<CondDef>` and `<ConfDef>` of Table 1.

For example, we can define a new adjective *hot and stuffy* as follows.

"Let's call the condition that humidity is higher than 60 % and temperature is higher than 28 degrees *hot and stuffy*"

4.3. Rule Description Support Module

This module navigates users to describe rules in CADEL. When rules are described, they are compiled into rule objects to make the system easily process them.

```
<Command> ::=          <RuleDef> | <CondDef> | <ConfDef>
<RuleDef> ::=          [<PreCondition>] <Verb> <Object>
                       [<Configuration>] [<PostCondition>]
<Verb> ::=             "Turn on" | "Turn off" | "Record" |
                       ...
<Object> ::=           [<Article>] <DeviceName> [<Modifier>]
<Article> ::=          "a" | "an" | "the"
<PreCondition> ::=     [<TimeSpec>] "if" <CondExpr> ["then"]
                       | [<TimeSpec>] "when" <CondExpr>
                       | <TimeSpec>
<PostCondition>::=     "if" <CondExpr>
                       | "when" <CondExpr> | <TimeSpec>
<CondExpr> ::=         <Cond> [<PeriodSpec>] [<TimeSpec>]
                       | <Cond> <TimeSpec> <PeriodSpec>
                       | <CondExpr> "and" <CondExpr>
                       | <CondExpr> "or" <CondExpr>
                       | "(" <CondExpr> ")"
<Cond> ::=             <Sensor> [<Modifier>] <State> |
                       <UserDefinedCond>
<Sensor> ::=           <DeviceName> | <Person> | <Place> |
                       <Event> | "nobody" | ...
<Event> ::=            "baseball game" | ...
<State> ::=            [<Be>] "turned on" | [<Be>] "dark" |
                       [<Be>] "is higher than" Temperature |
                       [<Be>] "over" Percent
                       | [<Be>] "hotter than"
                       | [<Be>] "at" <Place>
                       | "comes back" | "returns home" | ...
<Be> ::=               "is" | "are"
<Temperature> ::=      <Figures> "degrees" | <Figures>
                       "degrees Celsius" | <Figures>
                       "degrees Fahrenheit"
<Configuration>::=     "with" <RowOfConfs>
<RowOfConfs> ::=       <Setting> "of" <Parameter> "setting"
                       | <RowOfConfs> "and" <RowOfConfs>
<Parameter> ::=        "temperature" | "channel"
<Setting> ::=          <Temperature> | <Channel> | ...
<Modifier> ::=         "at the second floor" | "at the
                       living room" | ...
<TimeSpec> ::=         "after" <Time> | "at" <Time>
                       | "until" <Time> | ...
<PeriodSpec> ::=       Period | "from" <Time> "to" <Time> |
                       Period "after" <Time>
<Time> ::=             [DateSpec] <TimeOfTheDay>
<DateSpec> ::=         <Date> | "every" <DayOfTheWeek>
<Period> ::=           "for" <Figures> "seconds" | "for"
                       <Figures> "minutes" | ...
<CondDef> ::=          "Let's call the condition that"
                       <CondExpr> <UserDefinedCond>
<ConfDef> ::=          "Let's call the configuration that"
                       <RowOfConfs> <UserDefinedConf>
```

Table 1. Syntax of CADEL

This provides the navigation functions which is provided through GUI. Rules are described in the following steps.

Rule description The interface for rule description is shown in Fig. 4. Basically, we use a GUI-based dialog box as shown in the figure. We also support a voice recognition system to specify rules. The interface consists of two sub-interfaces: condition description I/F and action configuration I/F.

(i) Retrieval of Context and Condition Description

In the condition description I/F, users can retrieve contexts and related sensors as shown in Fig. 5. The retrieval of contexts and sensors can be done by specifying combination of the following items: (1) keyword, (2) action, (3) sensor type, (4) sensor name, and (5) location. For example, the air-conditioner, the temperature meter and so on can be re-

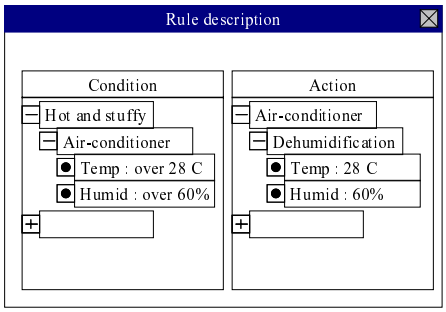

Figure 4. GUI for Rule Description

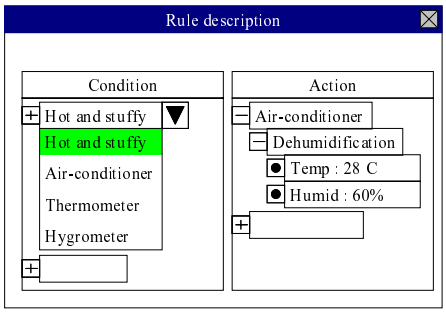

Figure 5. Condition Description by Retrieving Sensors

trieved by specifying temperature as the sensor type. Moreover, sensors can be retrieved by the user defined word. For example, sensors which can measure temperature and humidity can be retrieved by the word ”hot and stuffy”. Contrarily, information about sensor types and the user defined words can be retrieved by specifying sensors.

(ii) Definition of Compound Context as New Word

In the condition description I/F, users can define a new word to represent a compound context through GUI. Fig. 4 shows an example to define a new word *hot and stuffy* for representing the context with temperature more than 28 degrees and humidity more than 60 %.

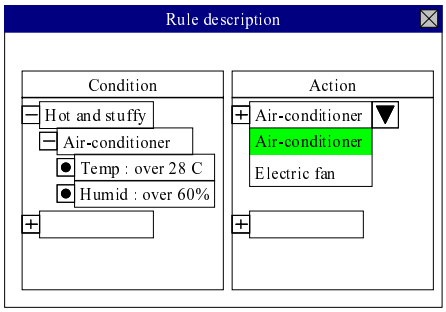

Figure 6. Action Selection by Retrieving Devices

(iii) Retrieval of Device Actions and Configuration

In the action configuration I/F (Fig. 6), devices can be retrieved by specifying combination of the following items: (1) keyword, (2) context, (3) action, and (4) location. By selecting a specific device in the retrieved device list, the I/F shows what actions are allowed in the device.

(iv) Import and Export of Rules

The navigation function of our framework greatly simplifies rule description. However, some users may still find it complicated to describe rules for various devices at different locations. So, our framework provides an import/export mechanism for rules. Users can import a rule registered in the database, and customize it to suit their preferences.

4.4. Consistency and Conflict Check Module

In CADEL, since condition in each rule is described as a logical conjunction of inequalities, it can be calculated efficiently to check whether the condition can hold or not.

Inconsistency Check of Rules

Whenever a new rule is described and registered in the system, the module evaluates the condition in the new rule to check whether it can hold. If the condition cannot hold, the module warns the user to modify the condition in the rule.

Conflict Check among Multiple Rules

If a new rule r_{new} is consistent and registered in the system, then the module checks whether r_{new} can conflict with other rules in the database in the following steps.

First, the module extracts from the database the set of rules R which control the same device as r_{new}. Then, for each rule r in R, the module checks if there is a value (a combination of values) satisfying both conditions of r_{new} and r simultaneously. If there is a value satisfying both conditions, r_{new} conflicts with r.

When the module detects a conflict, it warns the user to modify the new rule or to specify the priority order among the conflicting rules.

Specifying Priority Order When the conflict check module detects a conflict, it shows conflicting rules in a dialog box as shown in Fig. 7. In the dialog box, users can specify the priority order among those rules. Here, rules are arranged in the decreasing order of the priority degree [1] (i.e., the first rule (Alan) has the highest priority). If the priority order is already specified among some of them, users can modify the order.

1 In general, the partial order should be defined among those conflicting rules. Here, we use the total order to simplify the interface.

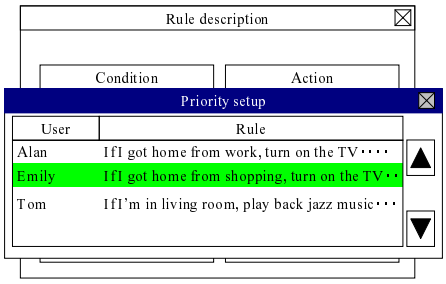

Figure 7. Interface to specify Priority Order

For flexible conflict avoidance, our framework provides a mechanism to change the priority order when the conflict occurs. When the system detects the conflict, it shows the conflicting rules with the current priority order among them and lets users to follow or modify the current priority order.

5. Experimental Results

We have implemented a prototype system based on the proposed framework and evaluated its performance in retrieving sensors/devices and in detecting conflicts over many rules. In our experiments, we used a PC with Athlon2200+ and 512M memory and JDK1.5.0 on Windows XP as the home server. We used CyberLink IPv6 for Java as the UPnP library. To detect conflicting rules, we implemented a C library for solving the satisfiability of given linear expressions using the Simplex Method. The experimental results are shown below.

Time for Retrieving Devices We invoked 50 instances of virtual UPnP devices on the PCs connected to the home server, and measured the time for retrieving a specified device by its device name. The execution time was 10ms or less. When we retrieved devices by their service names, the execution time was also 10ms or less. From the above result, we see that the retrieval time will not be a problem even when many devices are in a user's home.

Time for Detecting Conflicting Rules When a user registers a new rule in the home server, the server searches the existing rules conflicting with the new rule. The server (1) extracts existing rules which specify the same device as the new rule, (2) for each extracted rule, constructs a logical conjunction of linear inequalities by concatenating it and the new rule, and (3) checks if this expression has feasible solutions or not.

In the experiment, we assumed the case that the server retains 10,000 registered rules, and that among them 100 rules specify the same device in their action parts. We also assume that the condition part of each rule contains a logical product of two inequalities. Thus, a logical product of four inequalities must be evaluated for each extracted rule.

The time for extracting the rules with the same device was 10ms or less, and the time for evaluating a logical product of four inequalities 100 times was about 0.2ms.

The above result is good enough for practical use.

6. Conclusions

In this paper, we proposed a framework for context-aware computing systems with information appliances, which facilitates rule description and device configuration for ordinary home users. The novelty and contribution of this paper reside in that the proposed framework provides (1) an easy and intuitive way in configuring the whole system to personalize devices, and (2) a support for device conflicts among multiple users.

Through experiments, we confirmed that performance of our prototype implementation is practically good enough to retrieve sensors and devices and to detect conflicts over many rules.

Currently, we are implementing the whole framework including user interfaces. More detailed evaluation of the proposed framework will be part of future work. Also, we are going to implement in our framework some security mechanisms, e.g., for limiting access or allowable operations to each device depending on users' privileges.

References

[1] Guanling Chen and David Kotz : A Survey of Context-Aware Mobile Computing Research, *Technical Report TR2000-381, Dept. of Computer Science, Dartmouth College*, November (2000).

[2] http://www.upnp.org/

[3] http://wwws.sun.com/software/jini/

[4] Tatsuo Nakajima and Ichiro Satoh : Personal Home Server: Enabling Personalized and Seamless Ubiquitous Computing Environments, *Proceedings of 2nd IEEE International Conference on Pervasive Computing and Communications (PerCom2004)*, pp.341-345, March (2004).

[5] Jenna Burrell and Geri K. Gay and Kiyo Kubo and Nick Farina : Context-Aware Computing: A Test Case, *Proceedings of UbiComp 2002: Ubiquitous Computing: 4th International Conference*, pp.1-15, September (2002).

[6] Jakob E. Bardram : Applications of context-aware computing in hospital work: examples and design principles, *Proceedings of the 2004 ACM symposium on Applied computing (SAC2004)*, pp. 1574–1579 (2004).

[7] Gregory Biegel and Vinny Cahill : A Framework for Developing Mobile, Context-aware Applications, *Proceedings of 2nd IEEE International Conference on Pervasive Computing and Communications (PerCom2004)*, pp.361-365, March (2004).

[8] http://www.cybergarage.org/net/upnp/java/index.html