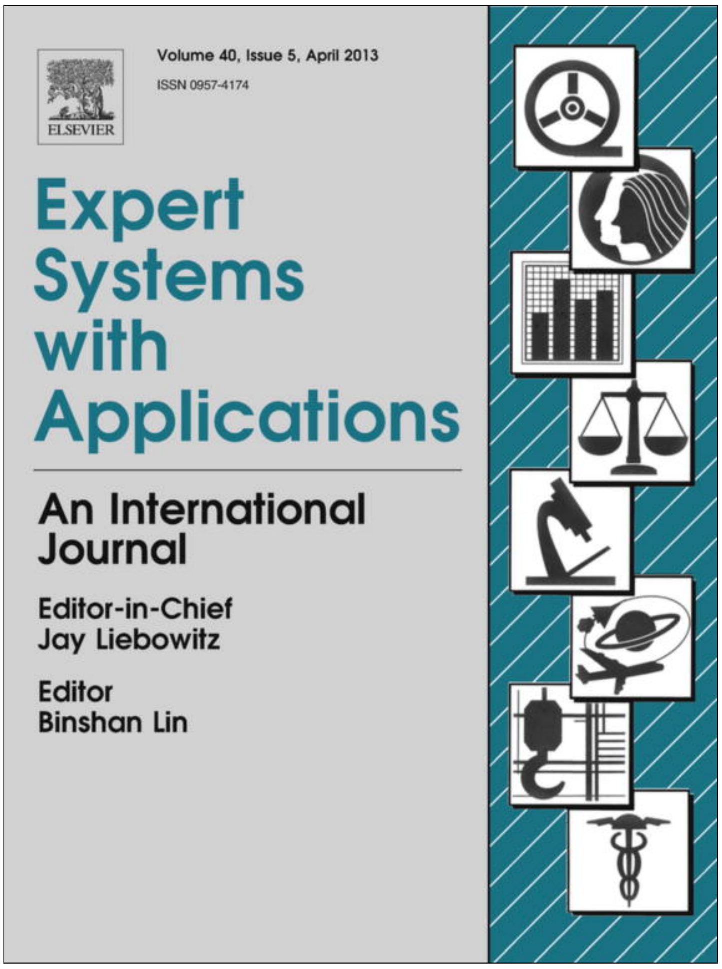

Volume 40, Issue 5, April 2013
ISSN 0957-4174
ELSEVIER
Expert Systems with Applications
An International Journal
Editor-in-Chief
Jay Liebowitz
Editor
Binshan Lin

Expert Systems with Applications 40 (2013) 1827–1838

Contents lists available at SciVerse ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

Bayesian and behavior networks for context-adaptive user interface in a ubiquitous home environment

In-Jee Song, Sung-Bae Cho *

Department of Computer Science, Yonsei University, 50 Yonsei-ro, Sudamoon-gu, Seoul 120-749, Republic of Korea

ARTICLE INFO

Keywords:
Adaptive user interface
Context-awareness
Bayesian network
Behavior network

ABSTRACT

Modern home theater systems require users to control various devices simultaneously including a TV, audio equipment, DVD and video players, and a receiver. To perform the requested user functions in this situation, the user is required to know the functions and positions of the buttons on several remote controls. Users will become more confused if a ubiquitous home environment, which contains many mobile and stationary control devices, is realized. Therefore, the user interface should be adaptable for requested user functions and to fit a specific control device. This paper presents a context-adaptive user interface for the control of devices in ubiquitous home environment. First, we modeled the ubiquitous home environment in order to implement the context-adaptive user interface. We used a Bayesian network to predict the necessary devices in each situation and used a behavior network to select the functions that constitute an adaptive user interface in several conditions. The selected functions were used to generate an adaptive interface for each controller using a presentation template. In this paper, we implemented a ubiquitous home environment and generated a controller usage log for this environment. We confirmed that the Bayesian network effectively predicted the user requirements by evaluating the inferred results of the necessary devices based on several scenarios. Finally, we compared the adaptive user interface with the fixed user interface by surveying fourteen subjects. We confirmed that the generated adaptive user interface was more comfortable for use with typical tasks than was the fixed user interface.

1. Introduction

People use remote controls in order to control appliances. As more appliances are purchased, the number of controllers also increases. For example, users have to control several devices including a cable TV receiver, DVD player, television, audio equipment, video players, and DivX players in order to use modern home theater systems. Controllers for these devices have different interfaces based on their manufacturers and models even though they look similar. Therefore, it is not easy to get accustomed to all of these controller interfaces (Nichols, Rothrock, Chau, & Myers, 2006). Even though the controllers have various functions, only a few of those functions are practical for use. If the ubiquitous home environment becomes more common, most home devices including lights, boilers and home appliances will also be controlled using a remote controller. Therefore, a context-adaptive user interface that provides users with necessary functions should be developed.

The personal universal controller (PUC), which automatically generates user interfaces in PDAs or smart phones, has recently been studied. Nichols and Myers of Carnegie Mellon University developed such a system, which can generate an optimized controller interface for smart phones using a hierarchical menu navigation system (Nichols & Myers, 2006) and introduced HUDDLE, a system that can automatically generate task-based user interface (Nichols et al., 2006). In addition, they verified through a user survey that the automatically generated interface had superior usability in terms of time efficiency and consistency than did the general interfaces for device control (Nichols, Chau, & Myers, 2007). These studies generated and provided a useful PUC, but they did not take into consideration the context for its uses. In a ubiquitous environment, it is very important to consider the context in order to provide relevant services and information (Lee, Seo, & Rhee, 2008).

This paper presents a context-adaptive user interface. In a modeled ubiquitous home environment with location and device information, the proposed method used the Bayesian network to predict the necessity of the devices given the context and the behavior network to select the necessary functions given the required devices in the current context. The adaptive user interface consists of these selected functions based on a presentation template. Modeling using Bayesian networks provides good performance by effectively incorporating the domain knowledge (Kleiter, 1996). We also used the behavior network so that the generated user interfaces could be adapted using the user input (usage log of the devices).

* Corresponding author.
E-mail address: sbcho@cs.yonsei.ac.kr (S.-B. Cho).

0957-4174/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.eswa.2012.09.019

For this experiment, we created a synthetic log for the use of the devices based on the experimental scenario, and then we inferred the necessities of these devices. We then evaluated the proposed method based on the time required to conduct the predefined tasks. Fourteen subjects were asked to perform ten tasks using both the conventional fixed home user interface and the proposed adaptive home user interface. The results indicated that the subjects dealt with the general tasks faster when using the proposed user interface than when using the conventional interface.

2. Related works

2.1. User Interface in a ubiquitous home environment

Recently, universal controllers or PDA software that can control several devices have been introduced (Nichols & Myers, 2006). In order to use these controllers and software, people have to assign the infra-red signal of the remote control to the corresponding button and then implement it. Home network technology, which allows users to access home appliances wherever they are, has been studied for the control of devices without the use of additional processes, as noted above (Filibeli, Ozkasap, & Civanlar, 2007). Technologies that are representative of home networking include Universal Plug and Play (UPnP) by Microsoft and Intel (http://www.microsoft.com/hwdev/upnp), Home Audio Video interoperability (HAVi) by Park, Kim, and Jeon (2006), and Jini by Sun Microsystems (Garud, Jain, & Kumaraswamy, 2002). Home networking technology, however, only connects devices, so the programmer must know the device information and functions and design the user interface to practically control the devices. As the number of devices increases, the interface becomes more complex. In order to solve this problem, a new interface is required whenever a new device is added. Implementing software user interface can be a solution. Umberger et al. proposed the integration of a home-automation system and its services with an IPTV system and its services at user interface level (Umberger et al., 2009). They provided user interface on TV screen. Kleindienst et al. proposed an interaction framework using speech and vision, and presented software user interface for PDA, MACI (Multi-modal appliance control interface) (Kleindienst, Macek, Seredi, & Sedivy, 2007). However, their user interfaces still depend on the corresponding controller for each device even though they did not use fixed physical remote controller. In order to solve this problem, a remote controller can provide a new interface when the appliances are added, removed or replaced in home environment. HUDDLE by Nichols and Myers proposed a task-based user interface, not device-based interface (Nichols & Myers, 2006), but they still had a problem where it did not consider the contextual information of users.

2.2. Bayesian networks for context-awareness

Context can be defined in several ways. Dey defined context as any information that can be used to characterize the situation of an entity such as a person, place or object that is considered relevant to the interaction between a user and an application, including the user and the application themselves (Kleiter, 1996). Generally, context influences the user preference for a service, but it also influences the control devices in a home environment. This is due to the fact that the desired control devices will change according to the user context.

Bayesian network has emerged as a powerful technique for handling uncertainty in complex domains, and especially for context-awareness to infer the context and provide reliable performance (Dey, 2001). It is a model of a joint probability distribution over a set of random variables. The Bayesian network is represented as a directed acyclic graph where nodes correspond to variables and arcs correspond to probabilistic dependencies between connected nodes.

$$P\langle B,\theta_B\rangle = P(x_1,x_2,\ldots,x_n) = \prod_{i=1}^{n} P(x_i|Pa(x_i)) \tag{1}$$

Eq. (1) represents Bayesian network formally. B and θ_B mean Bayesian network structure and probabilistic variables, and $P\langle B,\theta_B\rangle$ means a joint probability distribution of this network. A structure can be represented as $B = (V,E)$ where $V = \{x_1,x_2,\ldots,x_n\}$ is a set of nodes and E is a set of arch. For each x_i, a conditional probability distribution can be represented as $P(x_i|Pa(x_i))$, and $Pa(x_i)$ represents a parent set of a variable x_i.

They can be modeled manually based on domain knowledge even when there is no data to train them. Bayesian networks have been used to classify and infer the mobile context based on these strengths. Korpipaa et al. from VTT Technical Research Center utilized the naive Bayes classifier to train and classify the user context (Korpipaa, Koskinen, Peltola, Makela, & Seppanen, 2003). Horvitz et al. from Microsoft Research proposed a system that infers user focus on at a certain point in time in an uncertain environment (Horvitz, Kadie, Paek, & Hovel, 2003). Hong et al. proposed a context-aware messenger that automatically infers the user context and allows the users to share that context (Hong, Yang, & Cho, 2010). They exploited a dynamic Bayesian networks to infer contextual information. Lin et al. proposed a systematic approach to qualitatively diagnose QoS and to quantitatively guarantee QoS using Bayesian network training (Lin, Cheng, & Chen, 2010). Hwang and Cho used a modular Bayesian network approach to identify significant events from mobile contexts (Hwang & Cho, 2009). They modeled a large number of Bayesian networks using these modular Bayesian networks. Yap et al. presented an iterative training procedure for Bayesian network that discovers the minimal set of important contexts for mobile recommendation (Yap, Tan, & Pang, 2007). As described here, Bayesian networks have been exploited as reliable models for diverse context-aware applications.

2.3. Behavior network

The idea of a behavior network was presented by Maes for robot agent control (Maes, 1989). Fig. 1 presents an example of the representative behavior network. In this network, competition between actions is the basic characteristic and is represented by a change in the activation level of each action. The behavior with

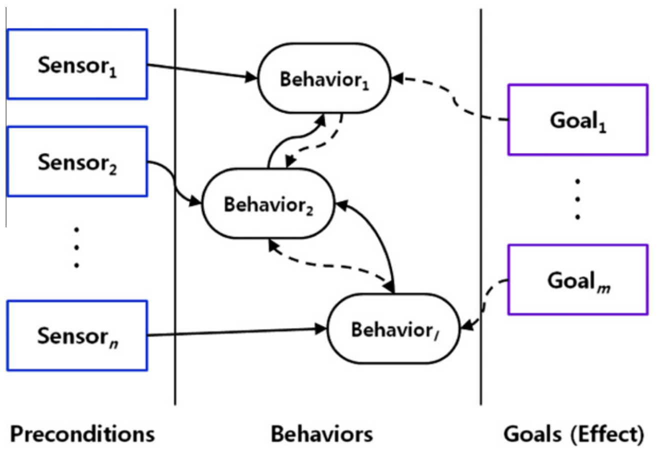


Fig. 1. An example of behavior network: the solid line represents a predecessor link, and the dashed line represents a successor link.

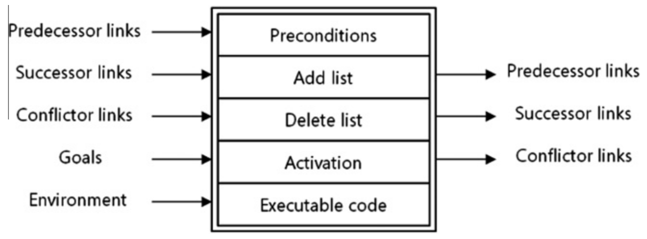


Fig. 2. Elements in a behavior node and its links to other nodes.

the highest activation level is selected after the activation spreading, and it is calculated as follows. Nodes can have different types of links that encode various relationships and stimulate one another. Each node has five components: precondition, add list, delete list, activation level, and executable code, as shown in Fig. 2. The precondition includes the conditions that must be true in order for the module to become active. The add list is a list of conditions that will be (or remain) true when the module is active. The delete list is a list of conditions that will be (or remain) false when the module is active. The executable code of each behavior is executed when all of the preconditions are true at a certain time. The links are divided into internal and external, and the internal links are subdivided into predecessor links, successor links and conflict links. The external links are connected to the sensor and goals. Assuming that there are two nodes, *x* and *y*, there is a successor link from *x* to *y* if a member of the add list of *x* is also a member of the precondition list of *y*. A predecessor link from *x* to *y* exists for every successor link from *y* to *x*. A conflictor link from *x* to *y* exists if a member of the delete list of *y* is also a member of the precondition list of *x*.

If the value of a certain sensor S_1 is true and S_1 is in the precondition list of a behavior node B, then a link from S_1 to B is activated. If goal G_1 has an activation level greater than zero and G_1 is in the add list of B, then a link from G_1 to B is activated. The procedure for selecting the behavior nodes to be executed at each step is as follows:

1. Calculate the excitation from the sensors and goals.
2. Spread the excitation along the predecessor, successor and conflictor links and normalize the activation level so that the average is equal to a preset constant.
3. Check any executable nodes, select the one with the highest activation level, execute it, and finish. A node is executable if all of the preconditions are satisfied and if its activation level is greater than the threshold. If there is no executable node, then reduce the threshold and repeat the process.

Links are set by a designer according to the problem. Based on the network designed, the system can select the proper behavior for its goal given a set of sensor states.

Since the behavior network was proposed for a behavior-based robot agent control (Maes, 1989), it has been used extensively in the robot domain (Nicolescu & Mataric, 2001; Yun & Cho, 2003). Nicolescu and Mataric used the behavior network to learn interactions between a human and a mobile robot (Nicolescu & Mataric, 2001). Yun and Cho proposed a dynamic behavior network based on a learning classifier system (LCS) (Yun & Cho, 2003). The LCS, a learning algorithm for a rule-based system, was used to train the behavior network. In Yun and Cho's study (Yun & Cho, 2003), the encoded network structure replaced the encoded rules of the typical LCS-based system. Dorer presented extended behavior networks in order to model human decision making in the sense of prospect theory (Dorer, 2010). He applied this model to RoboCup 2009 successfully. As shown in this work, behavior selection networks also can be used in non-robotic domains.

3. Context-adaptive user interface for a ubiquitous home environment

Fig. 3 summarizes the process for the proposed adaptive user interface generation for use with ubiquitous home devices. To begin with, the Bayesian network infers the necessary devices in the current context. Based on this result and the description of the devices and controllers, the behavior network is constructed in order to select the necessary functions for each device. Finally, the user interface for a given controller is generated using the user interface template.

3.1. Modeling a ubiquitous home environment

A ubiquitous home environment for adaptive user interface generation is modeled as follows.

- $E = \{Location\_List, Device\_List, Sensor\_List\}$
- $Location\_List = \{Location1, Location2, \ldots, LocationN\}$
- $Device\_List = \{Device1, Device2, \ldots, DeviceN\}$
- $Sensor\_List = \{Sensor1, Sensor2, \ldots, SensorN\}$
- $Location = \{Location\_Name, Location\_ID, Neighbors\}$
- $Device = \{Device\_Name, Device\_ID, Location\_ID, Function\_List\}$
- $Function\_List = \{Function1, Function2, \ldots, FunctionN\}$
- $Function = \{Function\_Name, Function\_ID, Function\_Type, Function\_Value, Add\_List, Delete\_List\}$
- Sensor=$\{Sensor\_Name, Sensor\_ID, Sensor\_Type, Sensor\_Value\}$

The ubiquitous home environment (E) includes the location list, which lists the locations of the users, rooms and devices; the device list, including appliances such as the TV and video player and equipment including the boiler and lights; and the sensor list, which identifies the states of the users and the environments. The device information includes the name, location, and functions of a device, and the function information includes the functions of each device and their constraints. The function list also includes the *Add_List* and *Delete_List* for the behavior network. The *Add_List* includes the functions that should be displayed with that function, while the *Delete_List* includes the functions that are not desirable for display with that function. Location information is represented as a room name.

3.2. Predicting the necessary devices using a Bayesian network

We used the Bayesian network to infer the devices that appear to be necessary for a given context in a ubiquitous home environment. Since the Bayesian network could be constructed using expert knowledge even though there was little or no data to train (Kleindienst et al., 2007), the system could result in a reliable performance in an uncertain home environment. After a sufficient user log was obtained, it was possible to train the Bayesian network.

In general, training Bayesian networks from data consists of two parts. The first is to train the structure and the other part is to train the parameter set. To train the structure can be solved by searching for the network structure which best matches the given data. The fitness can be measured by some scoring metrics. Two popular such metrics are the minimum description length (MDL) score and the Bayesian Dirichlet (BD) score. To find the best structure with score metric, greedy search algorithms can be used. A personalized model could be trained from the logs of the individual users.

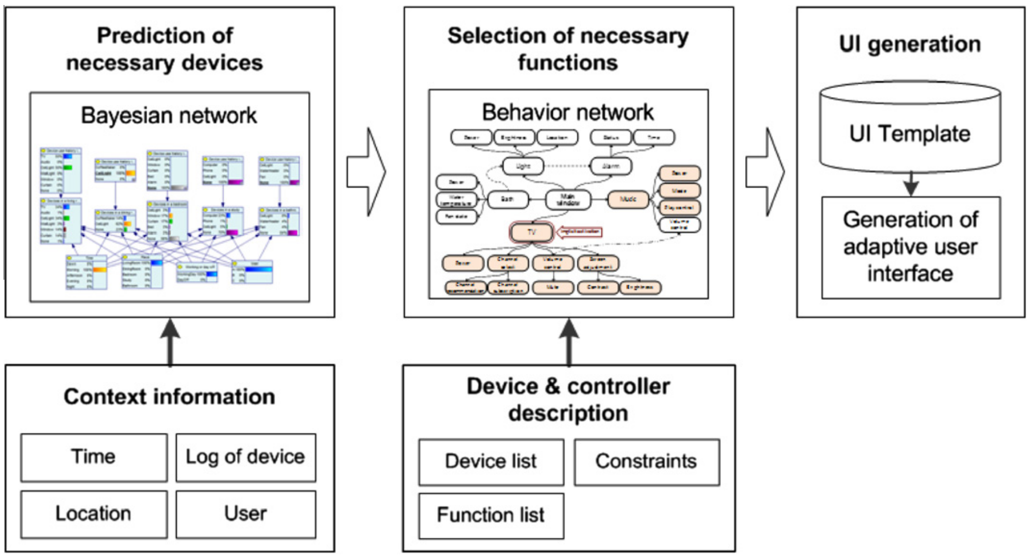


Fig. 3. The process for the generation of an adaptive user interface.

```
The K2 Algorithm
Input:  A set of $n$ nodes,
        An ordering on the nodes,
        An upper bound $u$ on the number of parents a node can have,
        A dataset $D$ including $m$ cases.
Output: A printout of each node and its parents.

{
    for ( i=1; i<n; i++ ) {
        $\Pi_i = 0$;
        $P_{old} = g(i, \Pi_i)$;
        OK2Proceed = TRUE;
        while ( OK2Proceed && $|\Pi_i| < u$ ) {
            Let Z be the node in Pred($X_i$)-$\Pi_i$ that maximizes $g(i, \Pi_i \cup \{Z\})$;
            $P_{new} = g(i, \Pi_i \cup \{Z\})$;
            if ( $P_{new} > P_{old}$ ) {
                $P_{old} = P_{new}$
                $\Pi_i = \Pi_i \cup \{Z\}$
            }
            OK2Proceed = FALSE;
        }
        print("node: ", $X_i$, "parents: ", $\Pi_i$);
    }
}
```

Fig. 4. A psuedocode for K2 algorithm to train Bayesian network.

We used the K2 algorithm proposed by Cooper and Herskovits to train our model as shown in Fig. 4 (Cooper & Herskovits, 1992).

Fig. 5 shows a simplified example of a Bayesian network used to infer the necessity of each device. The basic context including user location, current time and the date (whether the day was a holiday or not) could be considered as evidences to calculate the necessity of each device. The use history of devices that were related to each device in question was also used as evidence. The related devices are those with similar functions or those located in the same room. After all of the evidence was set, the BN inference was conducted to predict the necessity of the devices in the middle nodes in this figure. A device with higher probability is supposed to have more functions for generating a user interface than is a device with a lower probability.

3.3. Selecting necessary functions using a behavior network

Once we determined the necessity of each device, a behavior network was designed using the *Device_List* (D) and *Function_List* (F) noted in Section 3.1. If the necessities of the devices were

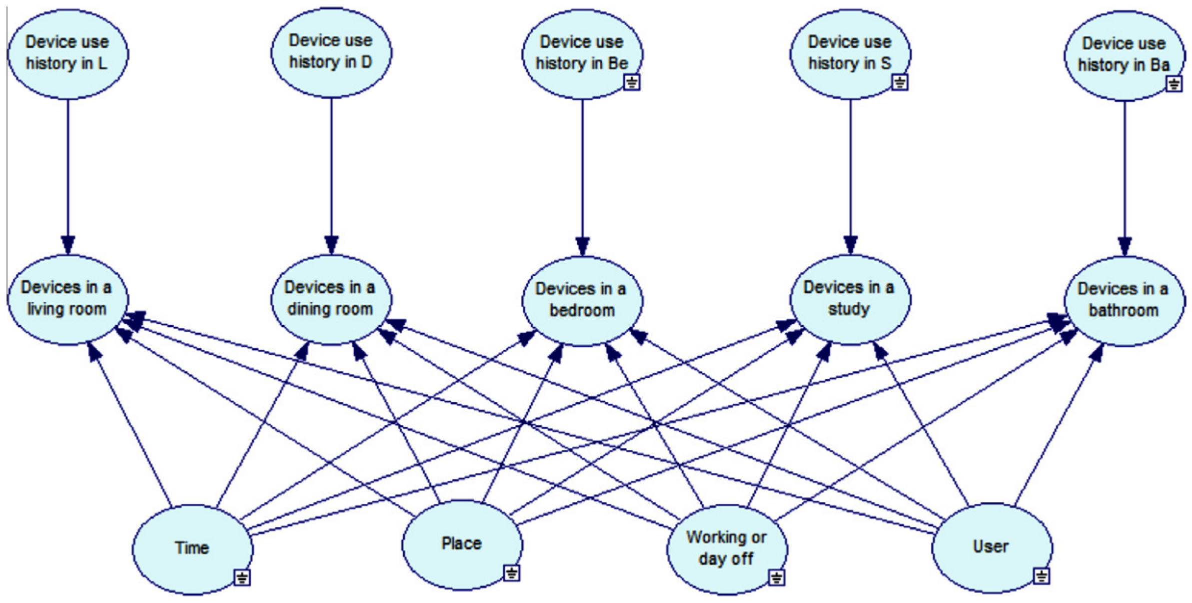


Fig. 5. A simplified example of a Bayesian network used to predict the necessary device.

grouped as a set of environmental nodes E and the states of the devices' functions were grouped as a set of functional nodes B, then E and B are defined by the following equations using the value of D and F.

$$E = \{e_i | Execute(e_i) \in D\} \tag{2}$$

$$B = \{b_i | Execute(b_i) \in F\} \tag{3}$$

After the nodes were generated, the predecessor links, successor links and conflictor links between those nodes were generated. The predecessor link was generated when Eq. (4) was satisfied. The predecessor link, which connects two functional nodes or a functional node and an environmental node, could be generated when both nodes belonged to the same device and when they were related positively to each other. A positive relationship occurs between the functions that are likely to be executed when a specific function is executed. This link was used to create a hierarchical structure of functions in one device. Successor and conflictor links were used to represent the relationship between the functions in the different devices. The successor link was determined using Eq. (5), and it was similar to the predecessor link. However, they were different in that successor link connected the functions of different devices and it only connected functional nodes. A conflictor link was generated if Eq. (6) was satisfied. It differed from the successor link in that it was generated when two functions had negative relationship (confliction).

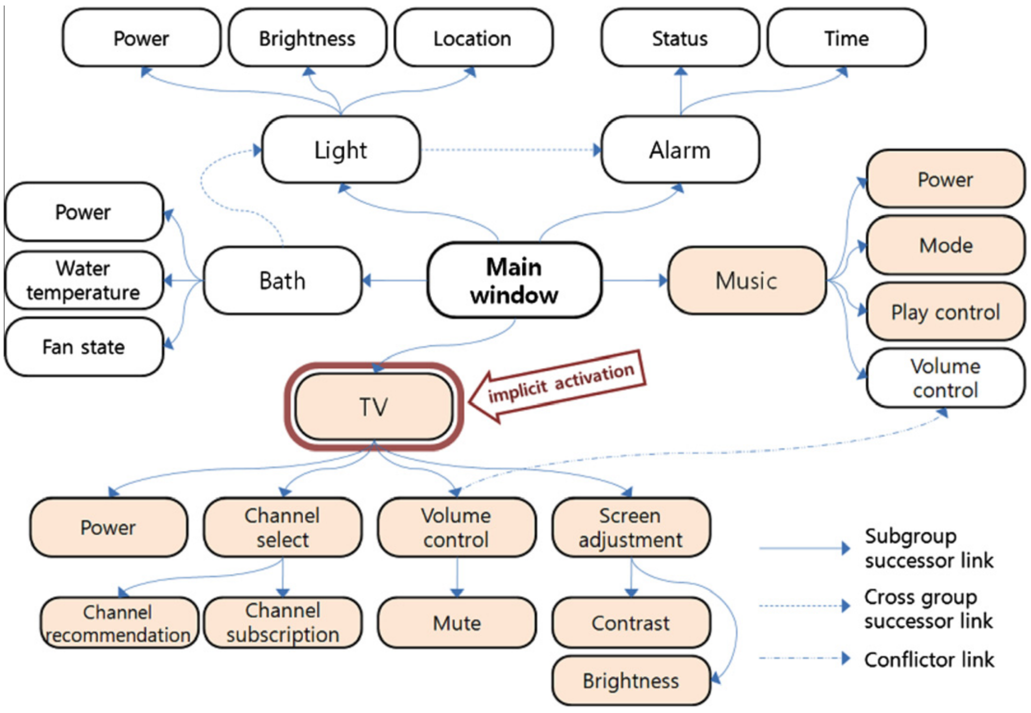


Fig. 6. Activation changes of an adapted behavior network after the user activated the TV.

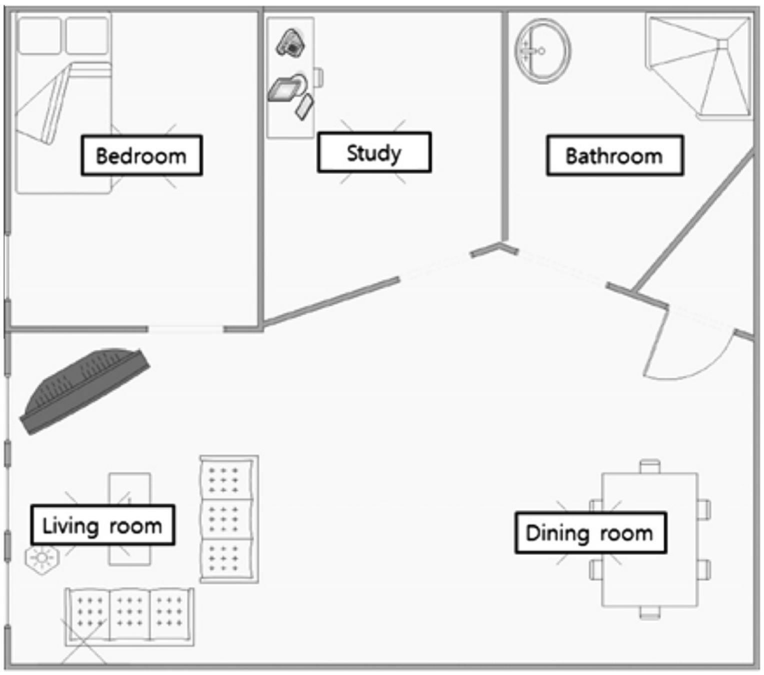


Fig. 7. A plane figure of the ubiquitous home environment.

$$predecessor(n_p, n_s) = \begin{cases} 1 & \text{if} \begin{cases} ((n_p \in B) \vee (n_p \in E)) \wedge (n_s \in B) \\ Device(n_p) = Device(n_s) \\ relation(Excute(n_p), Excute(n_s)) \end{cases} \\ 0 & otherwise \end{cases} \tag{4}$$

$$succesor(n_p, n_s) = \begin{cases} 1 & \text{if} \begin{cases} ((n_p \in B) \wedge (n_s \in B)) \\ Device(n_p) \neq Device(n_s) \\ relation(Excute(n_p), Excute(n_s)) \end{cases} \\ 0 & otherwise \end{cases} \tag{5}$$

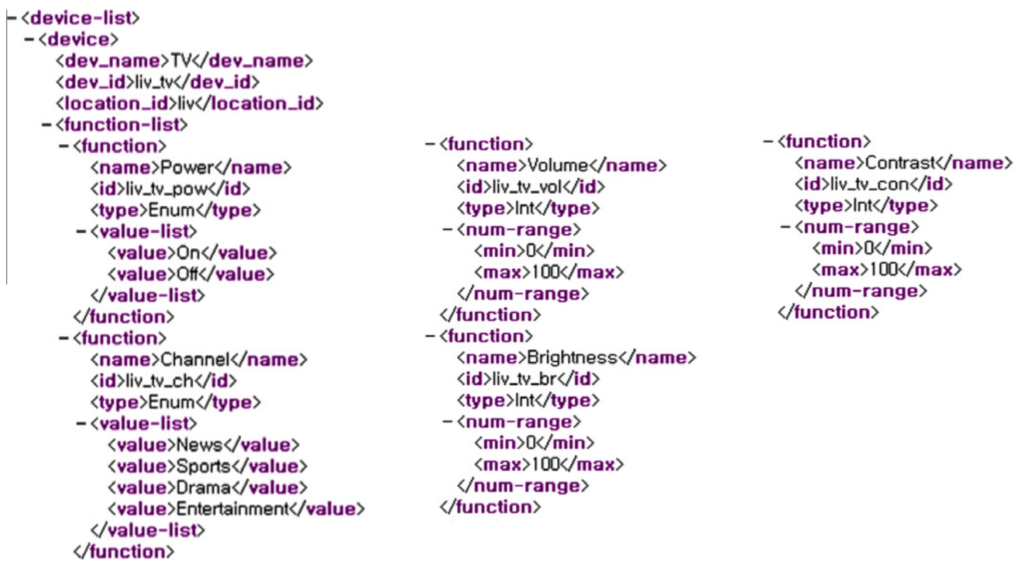
$$conflictor(n_p, n_s) = \begin{cases} 1 & \text{if} \begin{cases} ((n_p \in B) \wedge (n_s \in B)) \\ Device(n_p) = Device(n_s) \\ confliction(Excute(n_p), Excute(n_s)) \end{cases} \\ 0 & otherwise \end{cases} \tag{6}$$

A constructed behavior network is basically represented by a tree, which has device nodes as parents and their functions as children, and each link is added to that tree based on its functional relationships. In this network, the activation functions were evaluated as F: $E \times B \to [0\ldots1]$. Using the inferred necessities of the devices, a set of environmental nodes E was developed. After that, the procedure used to select a necessary function in the behavior network was followed as noted in Section 2.3. After the procedure was completed, we determined an activation level for each functional node. The active functional node $b_i(t)$ with the current state t was selected using Eq. (7). We allowed several functions to be selected at once, different from the conventional behavior network, and the user interface could display these selected functions during the next step.

$$b_i(t) = \begin{cases} 1 & \text{if} \begin{cases} \alpha_i(t) \geqslant \theta \\ executable(b_i, t) = 1 \end{cases} \\ 0 & otherwise \end{cases} \tag{7}$$

Table 1
Devices and functions in each room within the home environment.

Place	Device	Function	
		Name	Type
Living room	TV	Power	Enum
		Channel	Enum
		Volume	Range
		Brightness	Range
		Light intensity	Range
	Audio equipment	Power	Enum
		Mode	Enum
		Play	Enum
		Volume	Range
	Ceiling light	Power	Enum
		Light intensity	Range
	Wall light	Power	Enum
		Light intensity	Range
	Window	Open/Close	Enum
	Curtain	Open/Close	Enum
Dining room	Coffee maker	Power	Enum
		Status	Enum
	Ceiling light	Power	Enum
		Light intensity	Range
Bedroom	Ceiling light	Power	Enum
		Light intensity	Range
	Window	Open/Close	Enum
	Curtain	Open/Close	Enum
	Bed	Fold/Unfold	Enum
	Alarm	Status	Enum
		Set (am/pm)	Enum
		Set (hour)	Range
		Set (minute)	Range
Study	Computer	Power	Enum
		Status	Enum
	Phone	Status	Enum
	Ceiling light	Power	Enum
		Light intensity	Range
Bathroom	Ceiling light	Power	Enum
		Light intensity	Range
	Instantaneous water heater	Power	Enum
		Temperature	Range
	Fan	Power	Enum
		Mode	Enum

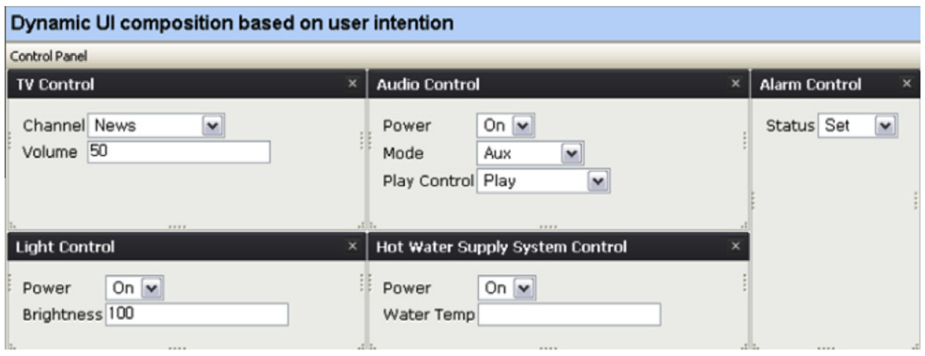


```
- <device-list>
  - <device>
      <dev_name>TV</dev_name>
      <dev_id>liv_tv</dev_id>
      <location_id>liv</location_id>
    - <function-list>
      - <function>
          <name>Power</name>
          <id>liv_tv_pow</id>
          <type>Enum</type>
        - <value-list>
            <value>On</value>
            <value>Off</value>
          </value-list>
        </function>
      - <function>
          <name>Channel</name>
          <id>liv_tv_ch</id>
          <type>Enum</type>
        - <value-list>
            <value>News</value>
            <value>Sports</value>
            <value>Drama</value>
            <value>Entertainment</value>
          </value-list>
        </function>
```

```
- <function>
    <name>Volume</name>
    <id>liv_tv_vol</id>
    <type>Int</type>
  - <num-range>
      <min>0</min>
      <max>100</max>
    </num-range>
  </function>
- <function>
    <name>Brightness</name>
    <id>liv_tv_br</id>
    <type>Int</type>
  - <num-range>
      <min>0</min>
      <max>100</max>
    </num-range>
  </function>
```

```
- <function>
    <name>Contrast</name>
    <id>liv_tv_con</id>
    <type>Int</type>
  - <num-range>
      <min>0</min>
      <max>100</max>
    </num-range>
  </function>
```

Fig. 8. Part of the XML code that represents the devices and functions.



Dynamic UI composition based on user intention

Control Panel

TV Control
Channel: News
Volume: 50

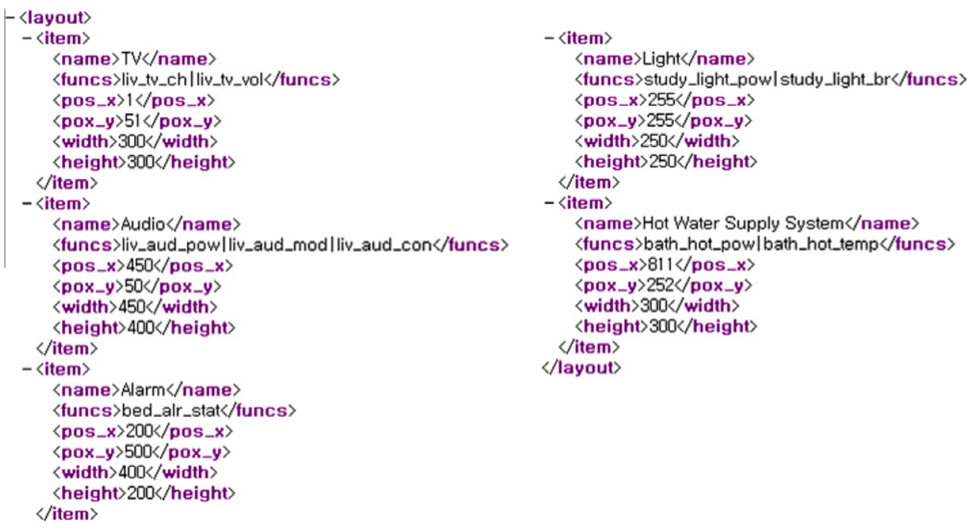
Audio Control
Power: On
Mode: Aux
Play Control: Play

Alarm Control
Status: Set

Light Control
Power: On
Brightness: 100

Hot Water Supply System Control
Power: On
Water Temp:

Fig. 9. Initial user interface (a holiday morning in a bedroom).

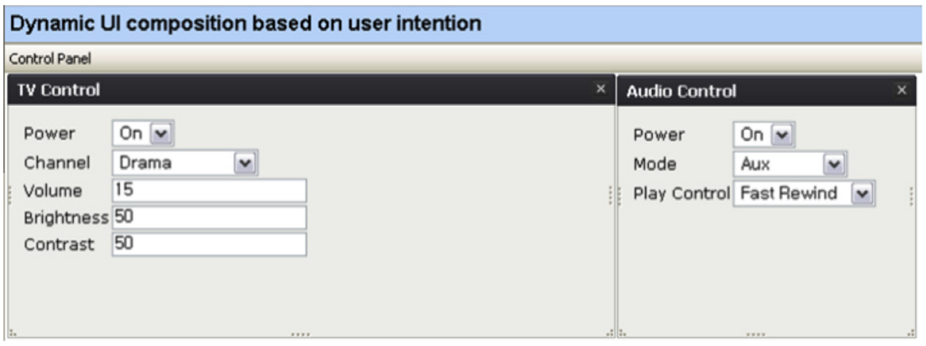


```
- <layout>
  - <item>
      <name>TV</name>
      <funcs>liv_tv_ch|liv_tv_vol</funcs>
      <pos_x>1</pos_x>
      <pox_y>51</pox_y>
      <width>300</width>
      <height>300</height>
    </item>
  - <item>
      <name>Audio</name>
      <funcs>liv_aud_pow|liv_aud_mod|liv_aud_con</funcs>
      <pos_x>450</pos_x>
      <pox_y>50</pox_y>
      <width>450</width>
      <height>400</height>
    </item>
  - <item>
      <name>Alarm</name>
      <funcs>bed_alr_stat</funcs>
      <pos_x>200</pos_x>
      <pox_y>500</pox_y>
      <width>400</width>
      <height>200</height>
    </item>
```

```
- <item>
    <name>Light</name>
    <funcs>study_light_pow|study_light_br</funcs>
    <pos_x>255</pos_x>
    <pox_y>255</pox_y>
    <width>250</width>
    <height>250</height>
  </item>
- <item>
    <name>Hot Water Supply System</name>
    <funcs>bath_hot_pow|bath_hot_temp</funcs>
    <pos_x>811</pos_x>
    <pox_y>252</pox_y>
    <width>300</width>
    <height>300</height>
  </item>
</layout>
```

Fig. 10. An example of the XML code that represents the UI layout.

The constructed network was evaluated again when the necessary devices in the device list changed frequently or when the user requested an evaluation. Fig. 6 depicts the behavior network used in this paper. We used two kinds of successor links with different semantics, but that were functionally the same. We connected a cross group successor link from the node "Bath" to the node "Light,"



Dynamic UI composition based on user intention

Control Panel

TV Control

Power: On
Channel: Drama
Volume: 15
Brightness: 50
Contrast: 50

Audio Control

Power: On
Mode: Aux
Play Control: Fast Rewind

Fig. 11. Changed user interface according to the user control of the TV.

Table 2
Contextual information used to generate data.

Day	Time to wake up	Time to return home	Time to go to bed
1 (Mon)	Dawn	Evening	Night
2 (Tue)	Dawn	Evening	Night
3 (Wed)	Dawn	Evening	Night
4 (Tur)	Dawn	Night	Night
5 (Fri)	Dawn	Evening	Night
6 (Sat)	Morning	Night	Dawn
7 (Sun)	Afternoon	No going out	Night
8 (Mon)	Dawn	Evening	Night
9 (Tue)	Dawn	Night	Night
10 (Wed)	Dawn	Evening	Night
11 (Tur)	Dawn	Evening	Night
12 (Fri)	Dawn	Evening	Night
13 (Sat)	Morning	Night	Dawn
14 (Sun)	Morning	Afternoon	Night

because most baths are taken with the lights turned on. In this manner, we made a cross group successor link. Subgroup successor links, successor links according to the menu hierarchy, were obvious, so we omitted those details. In Fig. 5, if the user watched TV, the necessity of the TV increased, and the network evaluation was updated. In this figure, the filled behavior nodes were activated nodes. It was natural that many relative functions were included in the user interface in this case because the user was watching TV. If the Bayesian network selected parts of several devices, for example, a TV, a music player and a light, then the behavior nodes corresponding to those devices that were not selected would not be considered as candidate functions to be selected.

4. Experiments

We conducted experiments in a simulated ubiquitous home environment for evaluation. Comparing the proposed adaptive UI with the conventional fixed UI, we confirmed that the adaptive UI had a superior performance.

4.1. Experimental environment

The simulated home environment is illustrated in Fig. 7, which was implemented with Python and Javascript. It has five rooms including a living room, a dining room, a bedroom, a study, and a bathroom, and each room contains the devices summarized in

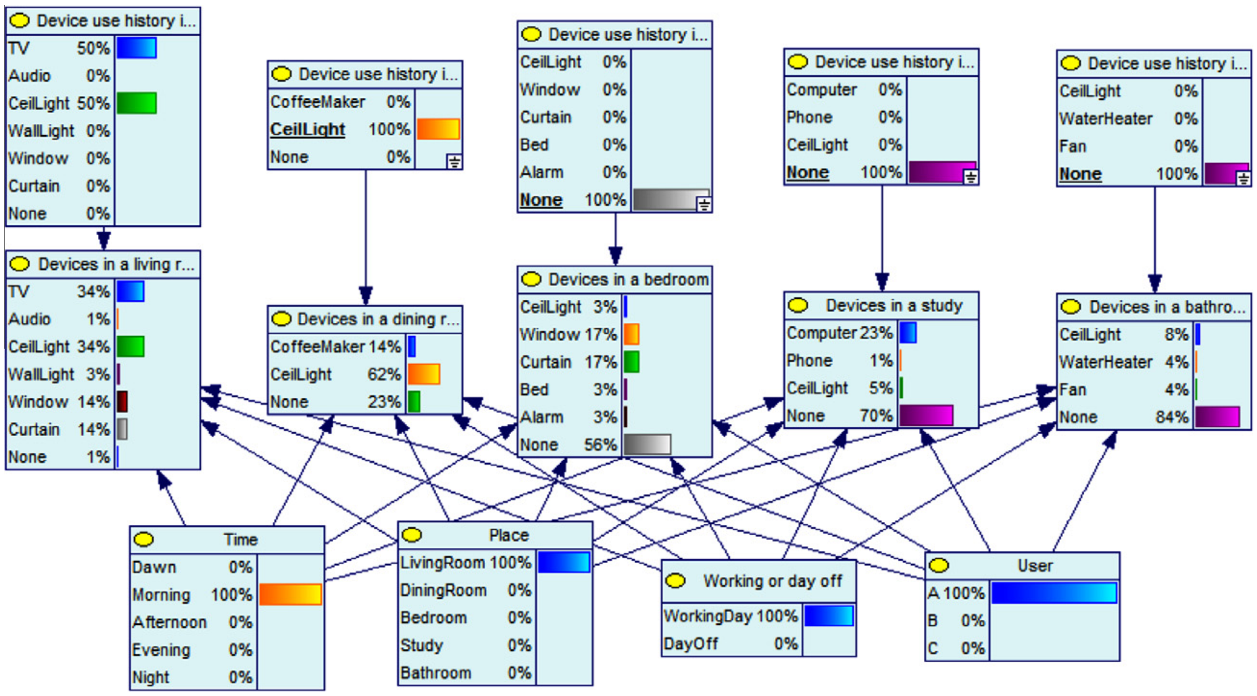


Fig. 12. Changed user interface according to the user's control of the TV.

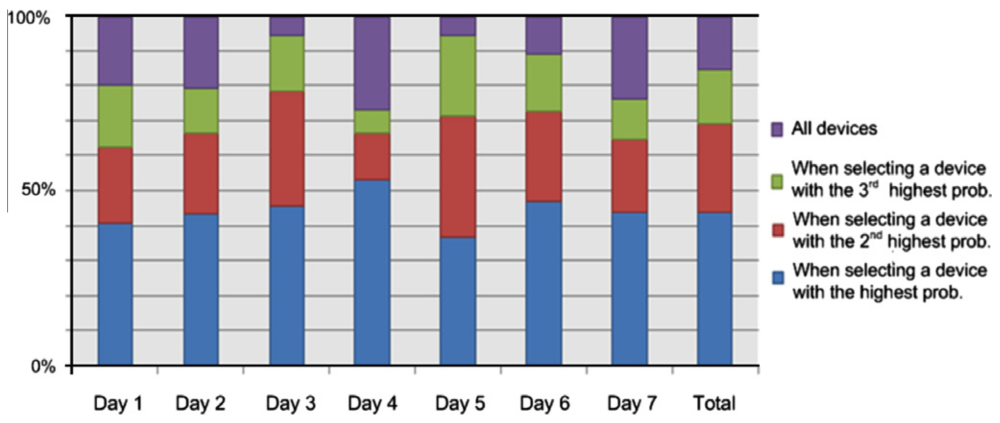


Fig. 13. Accuracy of the Bayesian network model by day.

Table 3
Ten situations and detailed tasks for usability test.

No.	Situation	Detailed task for situation
1	Having breakfast while listening to music	Turn on the ceiling light in the dining room → Turn on the coffee maker → Turn on the audio equipment in the living room → Set the frequency → Set the volume → Have the breakfast → Turn off the ceiling light in the dining room
2	Watching TV in the evening	Turn on the ceiling light in the living room → Turn on the TV in the living room → Set TV channel in the living room → Set TV volume in the living room → Turn off the TV in the living room → Turn off the ceiling light in the living room
3	Playing computer games	Turn on the ceiling light in a study → Turn on the computer → Play computer games → Turn off the computer → Turn off the ceiling light in the study
4	Getting out of bed in the morning	Turn off the alarm → Open the curtain → Open the window in the bedroom
5	Taking a shower	Set the temperature for the bath water → Turn on the light in the bathroom → Turn on the fan in the bathroom → Take the shower → Turn off the fan in the bathroom → Turn off the light in the bathroom → Set the temperature for the water
6	Closing all of the windows and curtains	Close the window in the bedroom → Close the curtain in the bedroom → Close the window in the living room → Close the curtain in the living room
7	Turning off all of the lights	Turn off the ceiling light in the bedroom → Turn off the ceiling light in the living room → Turn off the ceiling light in the study → Turn off the ceiling light in the bathroom
8	Watching TV while eating dinner	Turn on the ceiling light in the dining room → Turn on the ceiling light in the living room → Turn on the TV in the living room → Set the TV channel → Set the TV volume → Have a dinner → Turn off the ceiling light in the dining room → Turn off the ceiling light in the living room
9	Having a phone conversation in bed at night	Turn on the ceiling light in the bedroom → Have a phone call → Set the TV volume in the living room → Hang up the phone → Turn off the ceiling light in the bedroom
10	Going to bed at night	Turn on the ceiling light in the bedroom → Set the alarm → Close the curtain in the bedroom → Close the window in the bedroom → Turn off the ceiling light in the bedroom

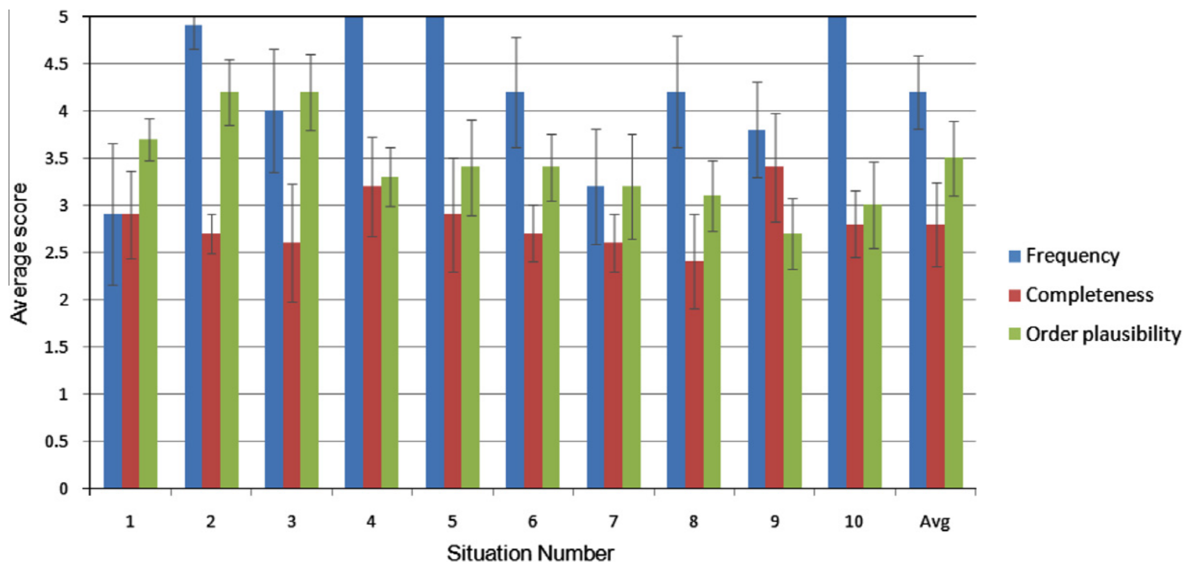
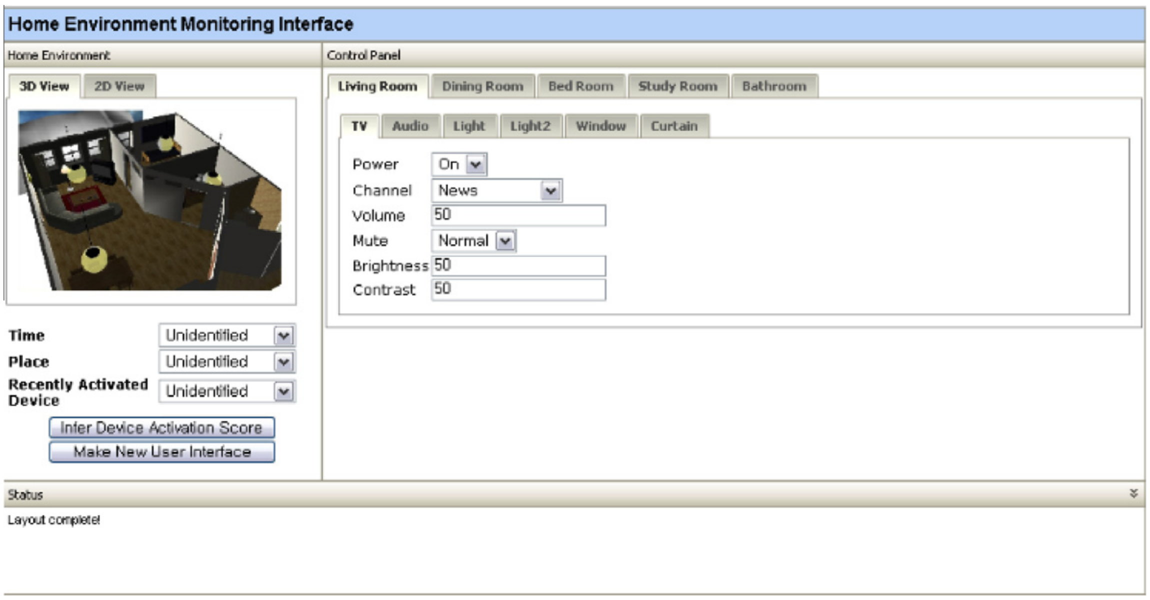


Fig. 14. Frequencies, completeness and order correctness for ten situations.


Home Environment Monitoring Interface

Home Environment | Control Panel

3D View | 2D View

Living Room | Dining Room | Bed Room | Study Room | Bathroom

TV | Audio | Light | Light2 | Window | Curtain

Power	On
Channel	News
Volume	50
Mute	Normal
Brightness	50
Contrast	50

Time	Unidentified
Place	Unidentified
Recently Activated Device	Unidentified

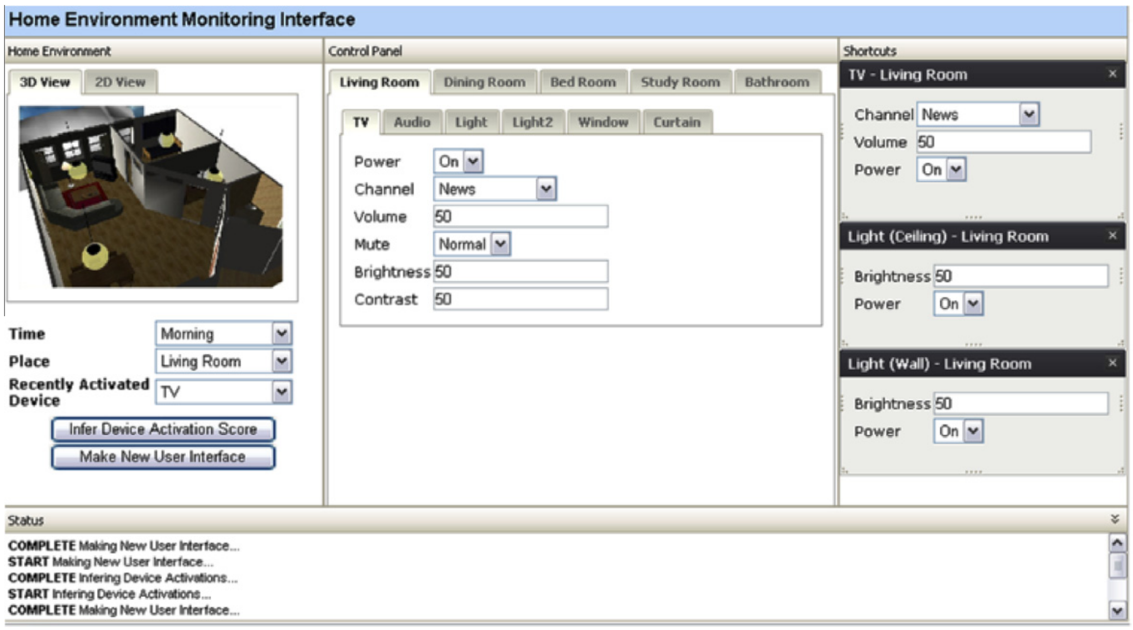
Infer Device Activation Score

Make New User Interface

Status

Layout complete!

Fig. 15. A user interface used in a modern home network environment (a fixed user interface only).


Home Environment Monitoring Interface

Home Environment | Control Panel | Shortcuts

3D View | 2D View

Living Room | Dining Room | Bed Room | Study Room | Bathroom

TV | Audio | Light | Light2 | Window | Curtain

Power	On
Channel	News
Volume	50
Mute	Normal
Brightness	50
Contrast	50

Time	Morning
Place	Living Room
Recently Activated Device	TV

Infer Device Activation Score

Make New User Interface

TV - Living Room

Channel	News
Volume	50
Power	On

Light (Ceiling) - Living Room

Brightness	50
Power	On

Light (Wall) - Living Room

Brightness	50
Power	On

Status

```
COMPLETE Making New User Interface...
START Making New User Interface...
COMPLETE Infering Device Activations...
START Infering Device Activations...
COMPLETE Making New User Interface...
```

Fig. 16. A user interface including the context-adaptive interface generated by the proposed method (an adaptive user interface).

Table 1. The lists of devices and functions are represented as XML in Fig. 8, which shows functions of TV as a device.

Figs. 9 and 11 show examples of the adaptive user interface when changing the functions and layout are changed according to the user's control of the device. Fig. 9 presents the interface generated by the proposed method in a situation where the date is a holiday morning and the place is a bedroom. The layout of the user interface is also represented as XML, and Fig. 10 provides the XML code that represents the user interface in Fig. 9. The layout represented by the XML is shown in a web browser processed by Javascript as shown in Fig. 9. After a user activated the TV with this interface, it changed as shown in Fig. 11 with more control buttons for TV because the necessity of the TV functions increased.

4.2. Evaluation of the necessity of the inferring device using the Bayesian network

In order to train the Bayesian network to infer device necessity, we collected data for two weeks on a middle-class man based on the home environment scenario presented in Fig. 7. The contextual information including the user, time, place and date (whether the day is a holiday or not) influences the device necessity and was an important factor in this scenario. The scenario used to generate the data was based on the contextual information in Table 2. The data provided included the times to wake up, return home and go to bed, and the weekends corresponded to holiday. The data for the first week was used for training, and that of the second week was used to test the model.

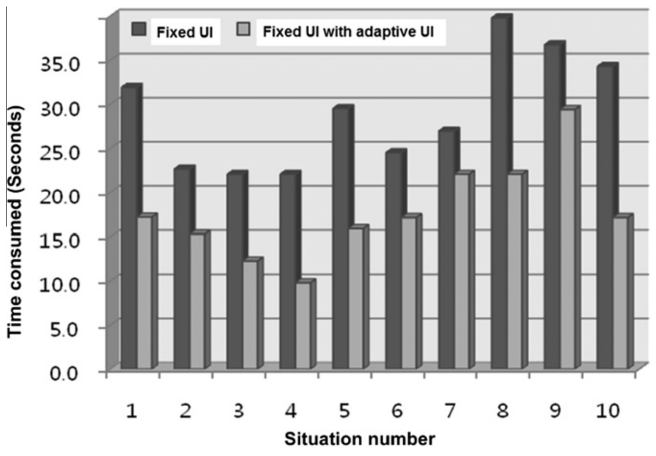


Fig. 17. Times to perform the tasks in ten situations.

Fig. 12 illustrates the initial Bayesian network structure designed and inferred probabilities of devices being in each place of home environment. Based on the contextual information observed or stored, the model could infer the necessities of the devices. Before using the probabilities of the device necessity, normalization was required because each node (room) contained a different number of devices, which could produce the wrong result. The parameters of the network were trained using the user logs that were collected periodically. Fig. 13 shows the accuracy of the Bayesian network model. We considered the accuracy of the models when the number of selected devices was one, two or three since it usually selects several devices. If we selected three devices, the average accuracy would be more than 80%. The correct answer in each situation was given by human and was based on common sense.

4.3. Usability test for the adaptive user interface

To evaluate the usability of the proposed method, we assumed ten situations that could happen in the home environment and evaluated with GOMS the generated adaptive interface based on time required to complete the predefined tasks in those situations (Card, Moran, & Newell, 1983). The situations and detailed tasks are provided in Table 3. We selected situations that could happen in the home environment in Section 4.1.

For each situation and its associated tasks, we conducted a questionnaire survey to measure the completeness (the number of associated tasks that they completed for each situation) and frequency (the number of times when they were in the situation) of the tasks, and the order plausibility (the correctness of the order of tasks completed for each situation). Fourteen subjects (seven males and seven females) in their 20s and 30s were participated in the survey, and answers were integers between 0 and 5. Fig. 14 shows the result of this survey. As shown, the tasks with high frequencies showed minimal standard deviations, indicating that tasks with high frequencies are typical.

Using these situations and tasks, we evaluated the user interfaces in Figs. 15 and 16 using the time required to conduct the same tasks. Fig. 15 had the interface with a tap menu for selecting the place and device. In order to control a certain device, two steps of tap selection were required. This interface was based on a controller design for a widely used home network. The interface in Fig. 16 adds a context-adaptive interface on the right side that changes when the user initiates device control or when the context changes. We compared the time required for fourteen users to perform the tasks in the ten situations listed in Table 3 using these two interfaces, and the results were summarized in Fig. 17. Table 4 shows the reduction rate of the time required when using the proposed method with the adaptive user interface. When using an adaptive interface, we achieved 38.63% reduction rate in the time required compared to the one with a fixed user interface. We also analyzed the results in the survey by frequency. If we classified ten situations into three groups according to frequency, then situations #2, #4, #5 and #10 were the most frequent, and situations #1, #7 and #9 were the least frequent. The situations which occurred more frequently could be considered more typical situations. Table 5 shows that the reduction rate of the time required by the more typical (frequent) situations was more than that of the less typical situations.

We have conducted a psychological test by Shéffe's method (David, 1969) in order to see the user's satisfaction. This method is a subjective test that requests subjects give a score to the difference between a given pair of interfaces. It makes a psychological distance measure from the score. The significance of difference among evaluated interfaces is tested by the analysis of variance. For this test, subjects were shown the two interfaces created by the proposed method and the conventional method, and then we request the subjects to evaluate the interfaces with a more appropriate look into 7 degrees from −3 to 3. The score about a difference between the interfaces has 7-point scale such as −3, −2, −1, 0, 1, 2, 3. We can observe relatively good satisfaction to the proposed interface since we get a mean score of 1.8 at 95% and at 99% of reliability.

5. Conclusions and future works

In this paper, we proposed an adaptive user interface generation method using Bayesian and behavior networks. The Bayesian network was used to select the necessary devices in a home environment, and the behavior network was used to select the functions that would be included in the adaptive user interface. For evaluation, we modeled a general home environment with devices and functions, and evaluated the user interface generated by the proposed method. Comparing to the fixed user interface, we achieved a 38% reduction rate in the time required to complete the tasks

Table 4
Reduction rate of the time required to complete the tasks in 10 situations (R. rate: reduction rate of the time required).

No.	1	2	3	4	5	6	7	8	9	10	Avg.
R. rate	46.07	32.45	44.44	55.56	46.14	30.00	18.18	44.60	20.22	50.00	**38.63**

Table 5
Reduction rate of the time required to complete the tasks classified by frequency (S: situations, R. rate: reduction rate of the time required).

Situations classified	Most frequent S. (#2, #4, #5, #10)	Second most frequent S. (#3, #6, #8)	Least frequent S. (#1, #7, #9)
Average R. rate	46.04	39.68	28.08

when using the adaptive user interface, and found that the proposed interface is more satisfactory than the fixed one.

Although adaptive user interface is a good idea in a complex environment, it may also make the learning curve of a user interface become flat since it is more difficult to find a specific function or control. Instead of focusing on user interface presentation, some other work focuses on user preference on control settings, which could be more desirable. Chen's work is an example of Bayesian network used in this application of smart home environment (Chen et al., 2006). More critiques or comparisons might be desirable to get to the sound conclusion.

There are many things remained for the future work. First, we need to improve the model with better training. A log of the user's activities was used to specify the Bayesian network. We will see how this network can adapt to changes in user's behaviors, and develop a deeper model of contextual information such as evolving user habits or weather conditions. Second, the accuracy of the user interface needs to be better defined instead of being judged by a human based on common sense. In order to improve the system evaluation we will include some qualitative measures on user experience and satisfaction from verbal user feedback. Finally, we will apply the proposed method to mobile devices so that it will be more practical. Experiments with data collected in a real home environment are also planned to produce more true-to-life results.

Acknowledgements

The authors would like to thank Dr. Han-Saem Park for preparing for this manuscripts. This research is supported by Ministry of Culture, Sports and Tourism (MCST) and Korea Creative Content Agency (KOCCA) in the Culture Technology (CT) Research & Development Program, and by Korea Communications Commission (KCC) as a project, "Development of UX based Smart TV Environmental Status Recognition Technology."

References

Card, S., Moran, T. P., & Newell, A. (1983). *The psychology of human computer interaction*. Lawrence Erlbaum Associates.

Chen, Z. -Y. et al.(2006). Using semi-supervised learning to build Bayesian network for personal preference modeling in home environment. In *IEEE international conference on systems, man and cybernetics*. Taipei, Taiwan.

Cooper, G., & Herskovits, E. A. (1992). A Bayesian method for the induction of probabilistic networks from data. *Machine Learning, 9*(4), 109–347.

David, H. A. (1969). *The method of paired comparison*. Charles Griffin and Co. Ltd.

Dey, A. K. (2001). Understanding and using context. *Personal and Ubiquitous Computing, 5*, 20–24.

Dorer, K. (2010). Modeling human decision making using extended behavior networks. *Lecture Notes in Artificial Intelligence, 5949*, 81–91.

Filibeli, M. C., Ozkasap, O., & Civanlar, M. R. (2007). Embedded web server-based home appliance networks. *Journal of Network and Computer Applications, 30*(2), 499–514.

Garud, R., Jain, S., & Kumaraswamy, A. (2002). Institutional entrepreneurship in the sponsorship of common technological standards: The case of sun microsystems and java. *The Academy of Management Journal, 45*(1), 196–214.

Hong, J.-H., Yang, S.-I., & Cho, S.-B. (2010). ConaMSN: A context-aware messenger using dynamic Bayesian networks with wearable sensors. *Expert Systems with Applications, 37*, 4680–4686.

Horvitz, E., Kadie, C. M., Paek, T., & Hovel, D. (2003). Models of attention in computing and communications: From principles to applications. *Communications of the ACM, 46*(3), 52–59.

Hwang, K.-S., & Cho, S.-B. (2009). Landmark detection from mobile life log using a modular Bayesian network model. *Expert Systems with Applications, 36*, 12065–12076.

Kleindienst, J., Macek, T., Seredi, L., & Sedivy, J. (2007). Interaction framework for home environment using speech and vision. *Image and Vision Computing, 25*, 1836–1847.

Kleiter, G. D. (1996). Propagating imprecise probabilities in Bayesian networks. *Artificial Intelligence, 88*(1–2), 143–161.

Korpipaa, P., Koskinen, M., Peltola, J., Makela, S.-M., & Seppanen, T. (2003). Bayesian approach to sensor-based context awareness. *Personal and Ubiquitous Computing, 7*(2), 113–124.

Lee, J. Y., Seo, D. W., & Rhee, G. (2008). Visualization and interaction of pervasive services using context-aware augmented reality. *Expert Systems with Applications, 35*, 1873–1882.

Lin, X., Cheng, B., & Chen, J. (2010). Context-aware end-to-end QoS qualitative diagnosis and quantitative guarantee based on Bayesian network. *Computer Commucations, 33*(17), 2132–2144.

Maes, P. (1989). How to do the right thing. *Connection Science Journal, 1*(3), 291–323.

Nichols, J., Rothrock, B., Chau, D. H., & Myers, B. A. (2006). Huddle: Automatically generating interfaces for systems of multiple connected appliances. In *ACM symposium on user interface software and technology* (pp. 279–288).

Nichols, J., Chau, D. H., & Myers, B. A. (2007). Demonstrating the viability of automatically generated user interfaces. In *ACM conference on human factors in computing systems 2007* (pp. 1283–1292).

Nichols, J., & Myers, B. A. (2006). Controlling home and office appliances with smart phones. *IEEE Pervasive Computing, 5*(3), 60–70.

Nicolescu, M., & Mataric, M. J. (2001). Learning and interacting in human-robot domains. *IEEE Transaction on Systems, Man, and Cybernetics Part A: Systems and Humans, 31*(5), 419–430.

Park, J. H., Kim, K. O., & Jeon, J. W. (2006). PDA based user interface management system for remote control robots. In *SICE-ICASE international joint conference* (pp. 417–420).

Umberger, M., Humar, I., Kos, A., Guna, J., Zemva, A., & Bester, J. (2009). The integration of home-automation and IPTV system and services. *Computer Standards & Interface, 31*, 675–684.

Yap, G.-E., Tan, A.-H., & Pang, H.-H. (2007). Dicovering and exploiting causal dependencies for robust mobile context-aware recommenders. *IEEE Transactions on Knowledge and Data Engineering, 19*(7), 977–992.

Yun, E.-K., & Cho, S.-B. (2003). Learning action selection network of intelligent agent. *Lecture Notes in Artificial Intelligence, 2903*, 578–589.