Implementing Personal Home Controllers on Smartphones for Service-Oriented Home Network

Keisuke Tokuda, Shinsuke Matsumoto, Masahide Nakamura
Graduate School of System Informatics, Kobe University
1-1 Rokkodai-cho, Nada, Kobe, Hyogo 657-8501, JAPAN
Email: tokuda@ws.cs.kobe-u.ac.jp, {shinsuke, masa-n}@cs.kobe-u.ac.jp

***Abstract*—The human interface devices for the home network system (HNS) should be flexible enough to reflect individual preferences and lifestyles of home users. To fill the requirement, this paper presents a novel framework that enables *Personal Home Controllers* (PHC) on smartphones. The proposed framework is designed so that a user can freely design screen layouts with buttons and pictures, and can define favorite HNS operations on the layouts. The first part of our contribution is *PHC Framework*, which dynamically implements a PHC based on a given user definition. The second part is *PHC Creater*, a GUI application which supports the user to create the user definition. The proposed framework is implemented for Android mobile devices.**

We conduct an experimental evaluation, where subjects create PHCs for operating an actual HNS. The experimental result shows that every subject creates a unique PHC with his favorite motif, and that most subjects find usefulness and pleasure in creating their own PHCs.

***Index Terms*—personalized service, mobile platform, home network system, service oriented architecture**

I. Introduction

In the next-generation smart home, house-hold appliances, equipment and sensors (e.g., TVs, lights, air-conditioners, curtains, temperature sensors) are connected with network. The *home network system* (HNS) [1] orchestrates such networked appliances to provide value-added services for home users. The service-oriented architecture (SOA) is shown to be a smart solution to achieve interoperability among heterogeneous devices in the HNS [2][3]. We have also developed a service-oriented HNS, called *CS27-HNS* [4], where the conventional appliances with infrared controllers were wrapped by Web services. The CS27-HNS is yet evolving with new applications developed, such as integrated services [4], the energy visualization service [5], the sensor service framework [6] and interactive voice controls [7].

In general, the human interface devices for operating the HNS has been developed by system vendors. Typical interfaces include remote controllers, control panels, ready-made applications on PCs or hand-held devices [4]. However, we found in our experience that such ready-made interfaces were not necessarily satisfactory for end users. The reason is that since the HNS is directly linked to the daily life, the usage varies significantly from person to person, from house to house. Thus, the interface must be flexible enough to reflect individual preferences and lifestyles of users.

The goal of this paper is to implement a framework that allows individual user to own a *personal home controller (PHC)*. The PHC is a home controller that can be highly personalized for every user. The proposed framework is designed so that a user can freely design screen layouts with buttons and pictures, and can define favorite HNS operations on the layouts. We also assume that the PHC is realized on a mobile platform (smartphone, PDA, portable game device, etc.). Thus, every member of a family will use his/her own home controller in the future.

To achieve the goal, our framework is implemented by two main components: *PHC framework* and *PHC creater*.

PHC framework: Application framework that separates a user definition of a PHC, from the control program of the HNS. The framework prescribes two languages: *PHC layout language (PHC-LL)* and *PHC action language (PHC-AL)*. They describe the user-defined layouts and actions of the PHC, respectively. When a pair of files in PHC-LL and PHC-AL are given, the framework interprets the languages and dynamically realizes a PHC on the mobile platform. The PHC framework is installed in a mobile platform.

PHC creater: GUI application that supports a user to create the definition files in PHC-LL and PHC-AL. The creater provides an intuitive visual programming environment, where a user can construct a screen layout consisting of backgrounds and buttons. The creater also provides an input form, with which a user can define an action as a sequence of operations. The action can include HNS operation, Web service invocation, forward screen, sound playback. The creater can export and import PHC-LL and PHC-AL files, in order to share and customize the PHC. The PHC creater is implemented as a PC application.

We implement the proposed framework for Android smartphones. Using the implementation, we conduct an experiment where 10 graduate students create PHCs operating a TV and a fan within the CS27-HNS. The experimental results shows that every subject creates a unique PHC with his favorite motif, and that most subjects find usefulness and pleasure in creating their own PHCs.

II. Preliminaries

A. Home Network System (HNS)

A home network system (HNS) orchestrates various household appliances, sensors and equipment via network. Each networked appliance exhibits the Application Program Interface (API) to the network. The API allows users or external agents

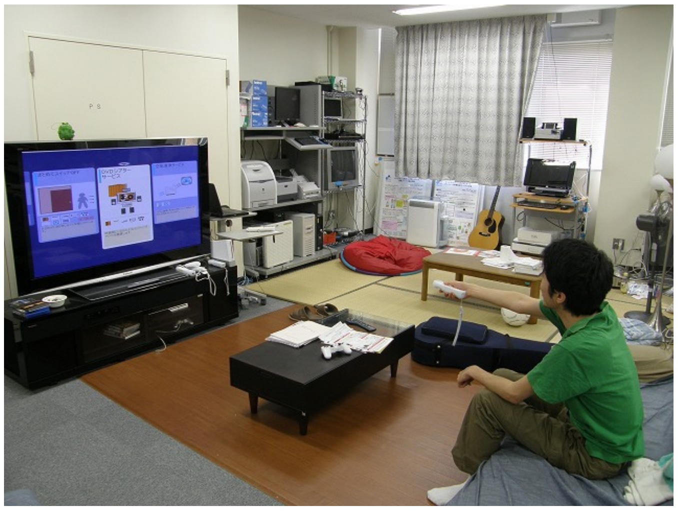


Fig. 1. CS27-HNS experimental room

to monitor and control the device. A HNS typically involves a home server, which manages all the networked devices and executes value-added services.

Our group has been developing a practical HNS, called CS27-HNS [8], based on the service-oriented architecture (SOA). Figure 1 shows the experimental room of the CS27-HNS. As shown in the figure, various appliances like TV, fan, air-conditioner, curtain, lights are installed. Every appliance is abstracted as a vendor-neutral service, encapsulating a device proprietary protocol (e.g., infra-red, serial COM, ZigBee, etc.) under the service layer. Since APIs are deployed as Web services, the appliances can be controlled by REST or SOAP. For example, accessing to the following URIs turns on a TV and sets the channel to no.8.

```
http://cs27-hns/TVservice/on
http://cs27-hns/TVservice/channel?num=8
```

B. Home Controller as Human Interface of HNS

To allow human users to operate appliances and services in the HNS, we need a human interface. The interface is often called *home controller*. As for the legacy appliance, a remote controller (often with infra-red communication) was bundled to every appliance. In the HNS, multiple appliances can be aggregated within a single home controller, with which a user can control all the appliances in the HNS. The home controller is basically developed by HNS vendors.

Figure 2 enumerates home controllers that we have so far developed for the CS27-HNS. The figure contains the flash interface on a wide-screen TV, the mobile phone interface, the wii-mote interface, the PC widget, the touch panel interface, the Nintendo DS interface.

In [7], we also developed an interactive voice control system for the CS27-HNS. In the system, a user just speaks a command like “Turn on a fan” to operate the appliances. The system also accepts context words like “hot” or “dark” to reason implicit user requirements.

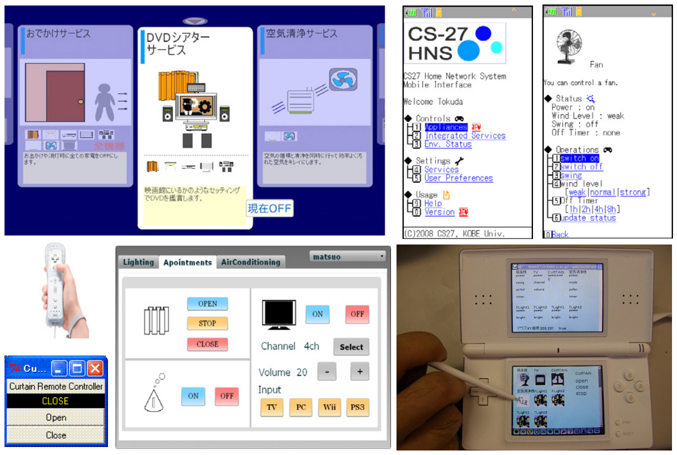


Fig. 2. Conventional home controllers for CS27-HNS

C. Personalizing Home Controllers

Although we have developed various home controllers, we fount it quite difficult to satisfy all the users. In the context of the HNS, there is a fact that installed appliances are different from house to house. Also, depending on the lifestyle, frequency in use of appliances varies from person to person. Therefore, the ready-made interface does not necessarily satisfy such varieties of users. The above observation motivated us to *personalize* the home controller.

We define the term *personal home controller* (PHC, for short) to refer a home controller whose user interface can be freely defined based on user’s preference. Due to its nature, the PHC would be realized on smartphones belonging to individual users.

D. Requirements of Personal Home Controllers

The goal of this paper is to provide a smart solution to implement the PHC. To clarify the scope of this paper, we here address the following two requirements.

Requirement R1: A user should be able to define user interface independently of control programs of the HNS. The home controller is generally composed of a user interface and control programs. The user interface defines screen layouts and callback actions when events occur (e.g., button pressed). While, the control program invokes designated appliance API of the HNS on receiving a request from the interface. In our previous implementations, the interface and the control program were tightly coupled. Therefore, the user could not define the user interface only. To achieve the PHC, we need a mechanism to decouple the user interface definition from the control program.

Requirement R2: A user should be able to create the PHC as easily as possible.

To satisfy a variety of users, the effort for creating the PHC should be as small as possible. For this, we need an intuitive and productive support tool for creating the PHC.

We also clarify the target users assumed in this paper.

Assumption on target users: We assume that users who just want to *use* the PHCs are able to use smartphones, and

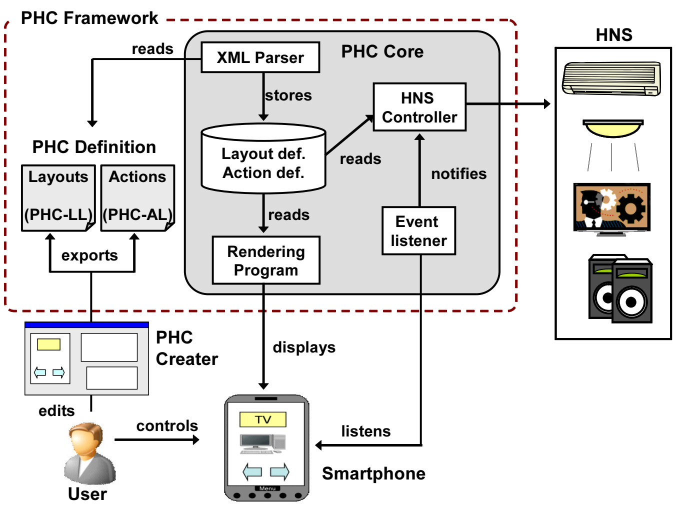


Fig. 3. Architecture of proposed framework

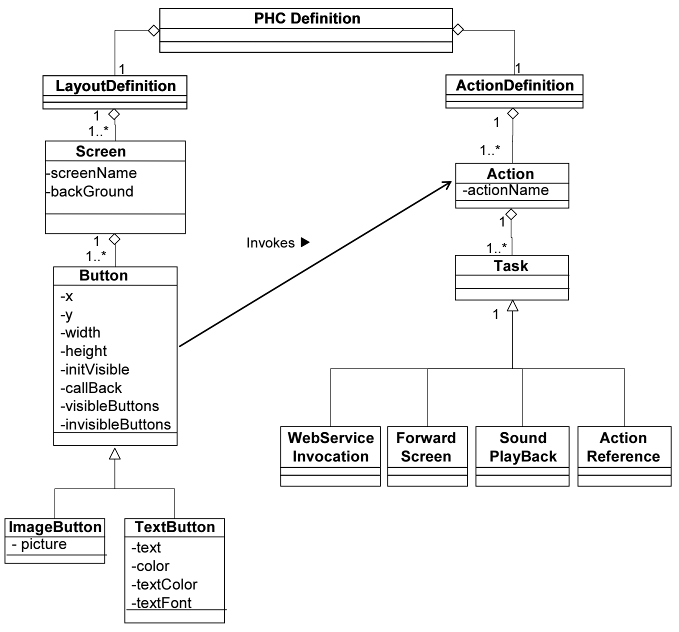


Fig. 4. Data structure of PHC definition

no other special knowledge is required. We also assume that users who want to *create* own PHCs should be familiar with basic PC operations like using editors and drawing images. However, no special knowledge on the HNS or programming is required.

III. Personal Home Controller Framework

A. Overview

In order to satisfy Requirement R1, we propose the *Personal Home Controller Framework (PHC framework)*. Figure 3 shows the overall architecture of the proposed framework. In the figure, the PHC framework is shown as a dotted rectangle. The PHC framework separates the definition of the PHC (*PHC definition*) from other programs of the home controller (described as *PHC core*).

The PHC definition is given as a pair of XML files: a *layout file* and a *action file*. The layout file defines screen layouts of the PHC, whereas the action file specifies actions of the PHC. These files are respectively described in a domain specific languages called PHC-LL and PHC-AL, as explained later.

The PHC core parses the PHC definition and stores the layouts and actions in a DB. Based on the layout definition, the rendering program displays a PHC on the smartphone. When a user press a button within the PHC, the event listener notifies the HNS controller of the button pressed. Finally, based on the action definition, the HNS controller identifies an appropriate HNS operation, and invoke it.

B. Description Languages for PHC Definition

We here introduce two kinds of description languages for the PHC definition. The one is the PHC layout language (PHC-LL) and another is the PHC action language (PHC-AL). The PHC-LL defines the *screen layouts* of a PHC, specifying screens, background, buttons and callbacks. On the other hands, the PHC-AL specifies *actions* of a PHC to be executed when the buttons are pressed. Both languages are represented in XML. Figure 4 the data structure of the PHC Definition in a form of UML class diagram, where LayoutDefinition and ActionDefinition are respectively represented by the PHC-LL and the PHC-AL.

PHC Layout Language (PHC-LL): As shown in the left side of Figure 4, the PHC-LL defines the screen layouts as a collection of *screens*. A screen is identified by a screen name and a background color. A screen has multiple *buttons*. Each button is characterized by coordinates (x, y), size (w $\times$ h) and visibility. Callback specifies an action that is triggered when the button is pressed. The attribute visibleButton (or invisibleButton) specifies buttons to appear (or disappear, respectively) when the button is pressed. These attributes are used to put or remove buttons dynamically in the screen. A *image button* is a button decorated with a picture. A *text button* is a button depicted by colored text.

Figure 5(a) shows an example of PHC-LL. In the figure, a screen named TOP is defined with a background color `#FFFFFF`. In the screen, two buttons TV and DVD-T are defined. The button TV is placed on the coordinates (65, 32) with the size 148x50. The button is initially visible and represented as a picture `tv_ms.png`. When the button is pressed, the action `goTV` is called back. Similarly, the button DVD-T is defined. This button appears as text `DVD Theater` with color `#000000`. The call callback is `DVDTheater`.

PHC Action Language (PHC-AL): As shown in the right side of Figure 4, the PHC-AL defines a set of *actions*. An action is defined by a set of *tasks*. A task can be a Web service invocation, forward screen, sound playback or action reference. The Web service invocation is a task that directly triggers a Web-API in the service-oriented HNS. The forward screen is a task that forwards the current screen to a designated screen. The sound playback is a task that plays

```
<screen>
  <screenName>TOP</screenName>
  <backGround>#FFFFFF</backGround>
    <buttons>

    <button>
      <buttonName>TV</buttonName>
      <x>65</x>       <y>32</y>
      <width>148</width>
      <height>50</height>
      <Visibility>true</visibility>
      <picture>tv_ms.png</picture>
      <callBack>goTV</callBack>
    </button>

    <button>
       <buttonName>DVD-T</buttonName>
       <x>72</x>  <y>151</y>
       <width>165</width>
       <height>60</height>
       <visibility>true</visibility>
       <color>#ffcccc</color>
       <text>DVD Theater</text>
       <textFont>30</textFont>
       <textColor>#000000</textColor>
       <callBack>DVDTheater</callBack>
    </button>
 </screen>
     :
```

(a) PHC-LL

```
<action>
 <actionName>goTV</actionName>
 <forwardScreen>TV</forwardScreen>
</action>
<action>
 <actionName>TVON</actionName>
 <webServiceInvocation>
  http:/hns/TVService/on
   </webServiceInvocation>
</action>
<action>
  <actionName>DVDTheater</actionName>
 <actionReference>
  TVON
  </actionReference>
 <webServiceInvocation>
  http://hns/LightService/off
  </webServiceInvocation>
   <webServiceInvocation>
  http://hns/CurtainService/close
   </webServiceInvocation>
<actionReference>
  DVD
  </actionReference>
   <soundPlayBack>
  playDVDTheater.mp3
   </soundPlayBack>
</action>
     :
```

(b) PHC-AL

Fig. 5. Example of PHC definition

back a sound within the PHC when the action is executed. The action reference is a task that calls the another existing action just like a sub-routine call.

Figure 5(b) shows an example of PHC-AL, where three actions are defined. The first action `goTV` forwards the current screen to `TV`. The second action `TVON` invokes Web service `http://hns/TVService/on/`, which turns on a TV in the HNS. The third action `DVDTheater` contains multiple tasks that turns on TV by `TVON`, turns off a light by Web service, and plays back a sound `playDVDTheater.mp3`.

C. PHC Core

The *PHC core* (see Figure 3) dynamically implements a PHC on a smartphone, based on given PHC definition. We explain sub-components of the PHC core as follows.

XML Parser: It parses given PHC definition files in PHC-LL and PHC-AL, and stores the layouts and the actions in a local data store.

Rendering Program: Interpreting the layouts definition, the program displays screen images on a smartphone. In the following, we explain how the screen rendering is performed. In the explanation, we use `<tag>` of PHC-LL.

1) Identify a screen (let it be *scr*) with `<screenName>`. The initial (default) screen is `TOP`.
2) For each button b in *scr*, set the coordinates of b from `<x>` and `<y>` attributes, where the origin $(0, 0)$ is at top-left of the screen.
3) Determine the design of the button b. set the size of b from `<width>` and `<height>`. If b is a TextButton, b is labeled by a string in `<text>`. The color and font of the text is set from values of `<color>` and `<font>`. If b is a ImageButton, load the image specified in `<picture>`.
4) If the `<visible>` is true, draw b on the screen.

Event Listener: It captures events that occur in the PHC. When a user press a button in the PHC, the event listener notifies which button is pressed of the HNS controller.

HNS Controller: On receiving an event notification from the event listener, the HNS controller performs the corresponding action. It works as follows.

1) Receive a notification that a button b is pressed from the event listener.
2) Identify an action act bound to b from the `<callback>` attribute of PHC-LL.
3) Derive a set of tasks $t_1, t_2, ..., t_n$ of act based on `<action>` of PHC-AL.
4) Execute $t_i (1 \leq i \leq n)$ one by one. If t_i is Web service invocation, invoke the corresponding Web-API of the HNS. If t_i is forward screen, update the value of the current screen and tell the rendering program to refresh the screen. If t_i is sound playback, play back the designated sound file from the smartphone. If t_i is action reference, call another action as a sub-routine.

We have implemented the PHC core for Android mobile devices. Many parts of the implementation were inspired by the Struts Web application framework [9]. The code was written in Java with Android SDK, comprising 2,267 lines of code. The development effort was about 3 man-months.

D. Example of PHC

Figure 6 shows an example of a PHC, which is implemented based on the PHC definition in Figure 5. Figure 6(a) represents the `TOP` screen. It contains two buttons as defined in Figure 5(a). If a user presses the TV button, the action `goTV` is called. In Figure 5(b), we can see that the task of `goTV` is a forward screen, and the screen should be forwarded to `TV`. Hence, the screen is refreshed to the one in Figure 6(b), which represents a TV controller.

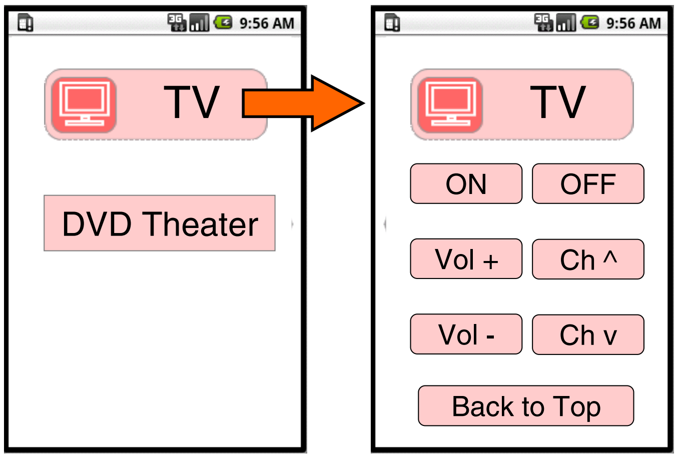
Although this example is quite simple, the PHC definition comprises many lines of XML as shown in Figure 5. Thus, it would be a tedious and unreliable task for a user to write the PHC definition from scratch. For practical settings with more appliances and services, it would be almost impossible to write the definition without fault. Therefore, a certain tool support is inevitable, as introduced in the next section.

IV. Personal Home Controller Creater

A. Overview

As seen in the previous section, writing the PHC definition from scratch imposes a huge amount of effort on a user, which cannot achieve Requirement R2. To cope with the problem, we present the *Personal Home Controller Creater (PHC creater)*. The PHC creater is a GUI application intended to support users to create PHC definitions.

As shown in Figure 3, a user uses the PHC creater to create and modify his PHC definition. Thanking to the GUI, the user can edit screen layouts and actions in a much more intuitive manner, rather than editing the XML files, directly. The layouts and actions edited on the PHC creater are exported in PHC-LL and PHC-AL files, which can be used in the PHC framework.



(a) Screen TOP (b) Screen TV

Fig. 6. example of Personal Home Controller

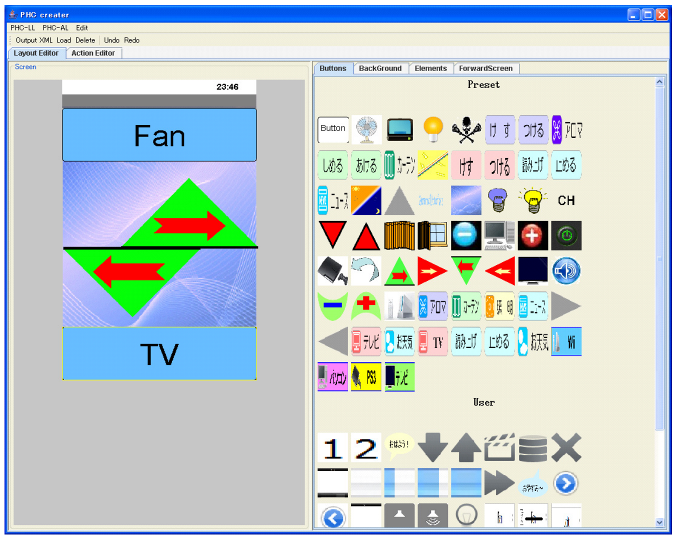
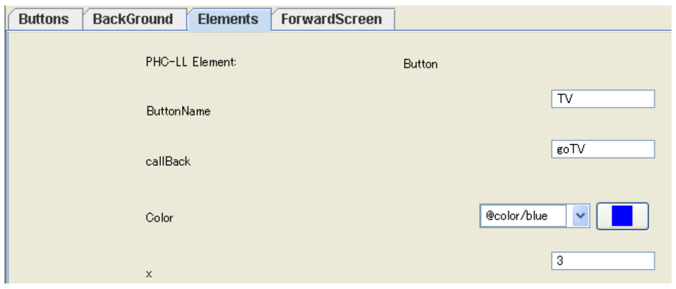


Fig. 7. Layout editor of PHC creater



PHC-LL Element:	Button
ButtonName	TV
callBack	goTV
Color	@color/blue
x	3

Fig. 8. Element pane of layout editor

The PHC creater is composed of two editors: *layout editor* and *action editor*. They are supposed to be used in PC. We explain these editors in the following subsections.

B. Layout Editor

The layout editor supports users to create the screen layouts of the PHC. Figure 7 depicts a screenshot of the layout editor. A user first creates a new screen, puts buttons on the screen, finally sets the parameters on each button.

The right side of Figure 7 displays a lot of button images. A user basically drags a button, and drops to a screen in left side. The size and position of every button can be changed by mouse operations. There are many button images preset in the layout editor, which cover most of typical household appliances. A user can also register custom button images to the editor.

Once a user places a button on the screen, the user can configure parameters to the button. Figure 8 shows the element pane, where a user can specify the parameters of the button. The input field corresponds to the tags of the PHC-LL. In this figure, parameters of a button `TV` are specified. The created layout is exported as a PHC-LL file.

C. Action Editor

The action editor supports users to define actions. Figure 9 shows a screenshot of the action editor.

A user first inputs an action name and description in Action Description section. Then, the user adds tasks to Home Appliance Operation section. A user inputs parameters of a task in Add Tasks section. The type of each task is either `ref` or `web service`. The type `ref` corresponds to the reference to the existing action, and the type `web service` corresponds to the Web service invocation. In the figure, a user is about to add a pre-defined action `LightOff`. All the tasks added so far are enumerated in TaskList section.

Tasks of forward action and sound playback can be specified in Device Action section. These actions are performed on the target smartphone. Each action can contain at least one forward action and at least one sound playback. In this example, a user defines an action `GoodNight`, which turns off TV and light and close curtain in the living room.

Once an action is created, the editor stores the action in a persistence storage for future reuse. The action editor can also import a file of pre-defined tasks, so that the user can reuse the existing HNS operations easily.

D. Implementation of PHC creater

We have implemented the PHC creater based on an open-source software DroidDraw [10]. The code was written in Java with Android SDK, comprising about 11,000 lines of code. The development effort was about 2.5 man-months.

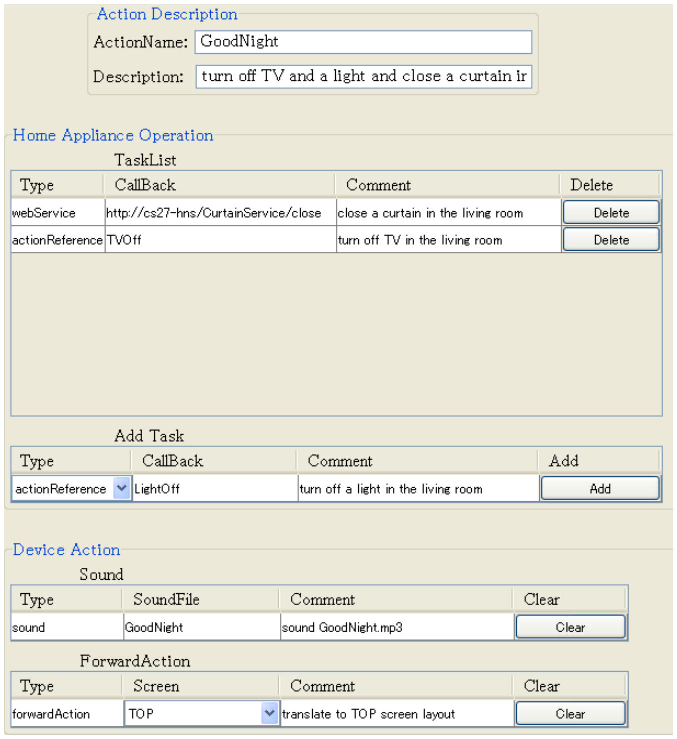
V. Experimantal Evaluation

A. Design of Experiment

We have conducted an experiment, where subjects create their own PHCs. The objective of the experiment is to evaluate the proposed framework from the following three viewpoints.

1) validity of personalization of home controller.
2) capability of the PHC framework
3) usability of the PHC creater

Total 10 subjects participated the experiment, consisting of 7 graduate students and 3 undergraduates. They were all males



Action Description

ActionName: GoodNight

Description: turn off TV and a light and close a curtain ir

Home Appliance Operation

TaskList

Type	CallBack	Comment	Delete
webService	http://cs27-hns/CurtainService/close	close a curtain in the living room	Delete
actionReference	TVOff	turn off TV in the living room	Delete

Add Task

Type	CallBack	Comment	Add
actionReference	LightOff	turn off a light in the living room	Add

Device Action

Sound

Type	SoundFile	Comment	Clear
sound	GoodNight	sound GoodNight.mp3	Clear

ForwardAction

Type	Screen	Comment	Clear
forwardAction	TOP	translate to TOP screen layout	Clear

Fig. 9. Action editor of PHC creater

in their 20's. The experiment was conducted in our CS27-HNS experimental room (see Section II-A). As for a smartphone for the PHC, we prepared Android Dev-Phone One (Android 1.6), ZiiO7 (Android 2.1), and GALAXY Tab (Android 2.2). For the appliances controlled, we used a plasma TV (Panasonic VIERA TH-58PZ800), a fan (Doshisha GIR-350), a ceiling light (Panasonic HHFZ5810) and a curtain (NAVIO power track) installed in the CS27-HNS. The API references for operating these appliances were registered in the PHC creater.

In the experiment, we told the subjects a certain functional requirement of a home controller. Then, each subject individually created a PHC that conformed to the requirement. We measured the development effort (i.e., time) taken for each subject. After the development, we interviewed each subject to ask the satisfaction of his created PHC. Also, we asked questions about the above three viewpoints.

B. Requirement and Tasks

The functional requirement of the PHC was as follows:

- For the TV, the PHC must be able to power on/off, change the input mode TV and DVD, set sound volume and change channels.
- For the ceiling light, the PHC must be able to turn on/off.
- For the fan, the PHC must be able to switch on/off, swing, change wind volume.

We did not impose any non-functional requirements such as layout designs, button images, the number of screens, etc. These were up to the subjects. Some of them prepared favorite pictures by themselves. Some used the preset materials. For fine-grained effort measurement, the subjects were instructed to create the PHC, in accordance with four tasks.

Task 1 (Layout Design): Each subject sketches screen layouts of the PHC on a paper or a PowerPoint. Also the subject collects materials (icons, pictures, sounds) necessary for the design.

Task 2 (Layout Creation): Based on the design, each subject creates his own PHC, using the PHC creater. Since all the necessary actions were already set in the PHC creater, the subject just refers the preset one.

Task 3 (Layout Modification): To see the maintainability of the PHC, we change the functional requirement, and ask the subject to modify the PHC. The changes are:

- Change the TV/DVD input modes to game/PC modes.
- Add one more ceiling light.
- Delete the fan control.

Task 4 (Action Creation): Each subject creates the following two actions.

1) `GoodNight`, consisting of (a) turn off the TV, (b) switch off the ceiling lights and (c) close the curtain.
2) `GoodMorning`, consisting of (d) open the curtain, (e) turn on the ceiling lights and (f) turn on the TV.

After finishing these tasks, each subject operated the appliances using the PHC he created.

C. Result

Figure 10 shows screenshots of the PHCs created by the ten subjects (Subjects A, B, ..., I). Figure 11 shows that the user is controlling the TV by PHC in Galaxy Tab. We confirmed that all the PHCs satisfied the functional requirement. We can see that every subject created a unique PHC with his favorite motif, reflecting personal preferences on the home controller.

TableI summarizes the time taken for PHC creation, which is shown in minutes. The average time taken for Tasks 1 to 4 were 126 mins, 76 mins, 23 mins and 22 mins, respectively. Roughly speaking, more than half of the time was spent to consider the layout design and to collect the materials in Task 1. We can see that the layout modification (Task 3) and the action creation (Task 4) were smoothly performed within relatively short time. The effort spent for Tasks 1 and 2 varied

TABLE I
TIME TAKEN FOR PHC CREATION (IN MINUTES)

	Layouts			Actions
	Task 1	Task 2	Task 3	Task 4
Subject	Design	Create	Modify	Create
A	57	22	8	23
B	57	5	9	26
C	77	40	24	8
D	57	19	16	33
E	210	12	30	10
F	330	30	30	20
G	132	150	45	19
H	144	137	19	22
I	83	227	24	23
J	114	116	29	35
Average	126	76	23	22

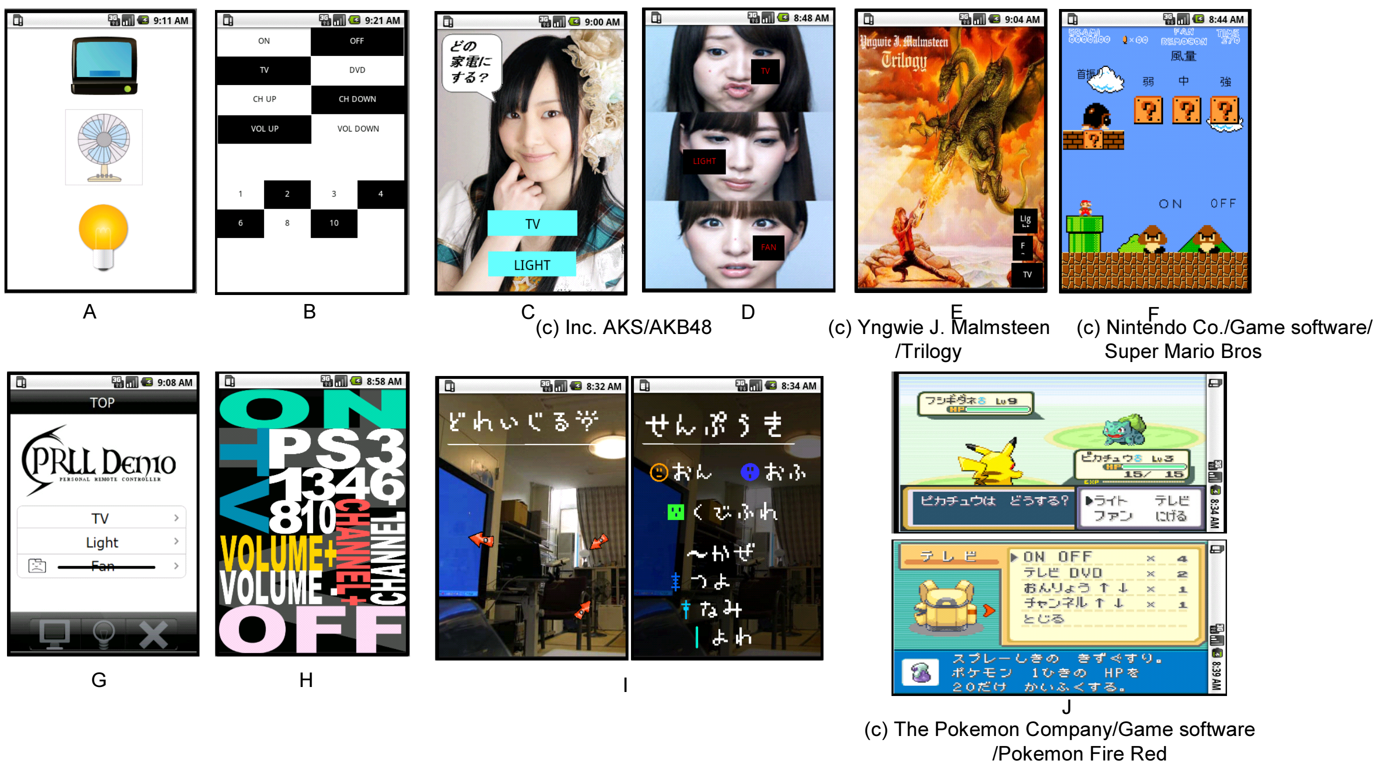


Fig. 10. PHC layouts which created by the subjects

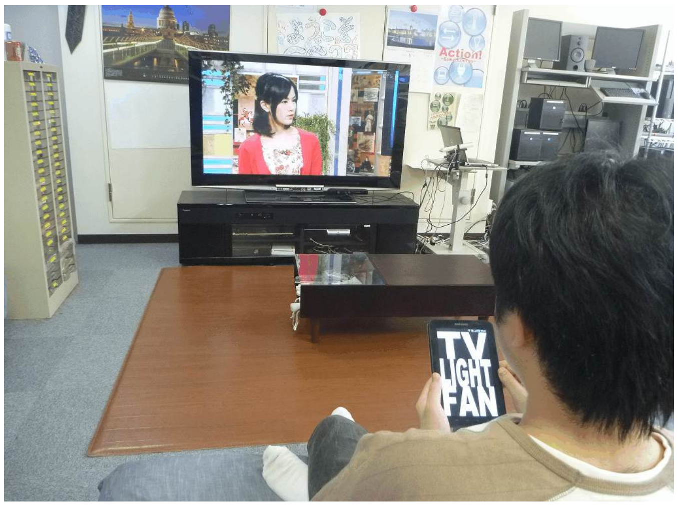


Fig. 11. Controlling the TV by PHC in Galaxy Tab

from one to another, depending on the *devotedness* of the subject to his PHC.

D. Evaluation

Based on the result and comments gathered in the subsequent interviews, we discuss the three viewpoints mentioned in Section V-A. As for the validity of the personalization of home controllers, we got a confidence because such a variety of PHCs were created in the experiment. Also in the interviews, many positive comments were obtained. For instance, "I enjoyed the process of PHC creation", "I love my PHC I created", "My favorite appliance can be arranged in the best position", "I can easily adapt the controller to changes of my preference and lifestyle".

Next, we discuss the capability of the PHC framework. We believe that the framework supports basic features to implement the home controllers. However, there were requests for improvement. Some subjects said that they want to show the status of the appliances and sensor values on the PHC. Another subject wanted to use *gesture* to operate the appliances. These features are not supported in our current framework. We will consider them in the future work.

Finally, we evaluate the usability of the PHC creater. In Table I, we can see that subjects G, H, I and J spent long time in Task 2. Hence, we suspect a usability problem. However, there was no negative comment from the subjects. They devoted themselves to create original and fancy products, which resulted in the long creation time. In the experiment, we found that the subjects tends to make mistakes in specifying URIs of web service invocation. We need to cope with this to improve the usability.

VI. Related Work

Crespo and Bier [11] proposed *WebWriter* that supports non-experts to create Web pages and applications. Since WebWriter covers general Web applications, it may be applied to the PHC creation. However, our framework is specialized to the home controller. Hence, we expect that the effort for the PHC creation will be significantly smaller than that of WebWriter.

Many studies have been conducted to implement user interface adapted to user's individual preference. Studies of Liu [12] and Lassila [13] automatically generate better user interface based on user profiles, status, and operation history. Kryzystof [14] proposed a system which automatically switches to preferable widgets, according to user's preference and used devices. For example, when a user operates a printer, if he has a mouse, then the system displays a drop-down menu. If he uses a touch panel, the system displays a list menu. These studies basically deal with interfaces automatically adapted to individual users. Our approach differs in that individual users create interfaces by themselves. However, the adaptive interface does not conflict with the PHC, and should be complementary to provide more convenient personal interface.

VII. Conclusion

In this paper, we have proposed a novel framework that creates the personal home controllers (PHC) for the home network system. Our contribution consists of the PHC framework and the PHC creater. The PHC framework is an application framework that dynamically implements a PHC on a smartphone from user-defined layouts and actions. The PHC creater is a GUI application that helps users create the PHC definition. We have conducted an experiment with ten subjects. The result showed that every subject was able to create a PHC reflecting individual preference, and that most subjects were satisfied with created PHCs.

In our future work, we plan to extend the PHC framework so as to include appliance status and sensor values within the PHC. Moreover, we refine the features of PHC creater as requested in the experiment. Introducing the adaptive interface to the PHC is also an interesting topic.

Acknowledgment

This research was partially supported by the Japan Ministry of Education, Science, Sports, and Culture, Grant-in-Aid for Young Scientists (B) (No.21700077) and Research Activity Start-up (No.22800042).

References

[1] M. Nakamura, A. Tanaka, H. Igaki, H. Tamada, and K. Matsumoto, "Adapting legacy home appliances to home network systems using web services," in *International Conference on Web Services (ICWS2006)*, September 2006, pp. 849–858.

[2] C. L. Wu, C. F. Liao, and L. C. Fu, "Service-oriented smart home architecture based on osgi and mobile agent technology," in *IEEE Trans. on Systems, Man, and Cybernetics (SMC), Part C*, vol. 37, no. 2, 2007, pp. 193–205.

[3] J. Bourcier, A. Chazalet, M. Desertot, C. Escoffier, and C. Marin, "A dynamic-soa home control gateway," in *Proc. the 3rd IEEE International Conference on Services Computing (SCC)*, 2006, pp. 463–470.

[4] M. Nakamura, A. Tanaka, H. Igaki, H. Tamada, and K. Matsumoto, "Constructing home network systems and integrated services using legacy home appliances and web services," *Int'l J. of Web Services Research*, vol. 5, no. 1, pp. 82–98, 2008.

[5] H. Igaki, H. Seto, M. Fukuda, and M. Nakamura, "Mashing up multiple logs in home network system for promoting energy-saving behavior," in *Proc. of 8th Asia-Pacific Symposium on Information and Telecommunication Technologies (APSITT2010)*, vol. CDROM, June 2010.

[6] M. Nakamura, S. Matsuo, S. Matsumoto, H. Sakamoto, and H. Igaki, "Application framework for efficient development of sensor as a service for home network system," in *the 8th IEEE 2011 International Conference on Services Computing (SCC)*, 2011, pp. 576–583.

[7] N. Matsubara, S. Matsumoto, and M. Nakamura, "Characterizing user habituation in interactive voice interface experience study on home network system," in *Int'l Conf. on Information Integration and web-based Applications & Services (iiWAS2011)*, 2011, pp. 375–378.

[8] M. P. Papazoglou and D. Georgakopoulos, "Service-oriented computing," *Communication of the ACM*, vol. 46, no. 10, pp. 25–28, 2003.

[9] "Struts," http://www.struts.org/.

[10] "Droiddraw," http://droiddraw.org/.

[11] A. Crespo and E. A. Bier, "Webwriter: A browser-based editor for constructing web applications," *Int'l J. Computer and Telecommunications Networking*, vol. 28, no. 7, pp. 1291–1306, 1996.

[12] J. Liu, C. K. Wong, and K. K. Hui, "An adaptive user interface based on personalized learning," *IEEE Intelligent Systems*, vol. 18, no. 2, pp. 52–57, 2005.

[13] D. Khushraj and O. Lassila, "Ontological approach to generating personalized user interfaces for web services," in *Proc. Int'l Conf. The Semantic Web*, vol. 3729, 2005, pp. 916–927.

[14] K. Z. Gajos, D. S. Weld, and J. O. Wobbrock, "Automatically generating personalized user interfaces with supple," *Artificial Interlligence*, vol. 174, no. 12-13, pp. 910–950, 2010.