<table>
<thead>
<tr>
<th><strong>Title</strong></th>
<th>A service-oriented framework for networked appliances to achieve appliance interoperability and evolution in home network system</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Author(s)</strong></td>
<td>Igaki, Hiroshi; Nakamura, Masahide; Matsumoto, Ken-ichi</td>
</tr>
<tr>
<td><strong>Citation</strong></td>
<td>WPSE’05: Eighth International Workshop on Principles of Software Evolution, 5-6 Sept. 2005, Lisbon, Portugal</td>
</tr>
<tr>
<td><strong>Issue Date</strong></td>
<td>2005</td>
</tr>
<tr>
<td><strong>Resource Version</strong></td>
<td>author</td>
</tr>
<tr>
<td><strong>Rights</strong></td>
<td>© 2005 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.</td>
</tr>
<tr>
<td><strong>DOI</strong></td>
<td>10.1109/IWPSE.2005.2</td>
</tr>
<tr>
<td><strong>URL</strong></td>
<td><a href="http://hdl.handle.net/10061/12766">http://hdl.handle.net/10061/12766</a></td>
</tr>
</tbody>
</table>
A Service-Oriented Framework for Networked Appliances to Achieve Appliance Interoperability and Evolution in Home Network System

Hiroshi Igaki, Masahide Nakamura and Ken-ichi Matsumoto
Nara Institute of Science and Technology
8916-5,Takayama-cho, Ikoma Nara 630-0192, Japan
{hiro-iga, masa-n, matumoto}@is.naist.jp

Abstract

To cope with both evolution and interoperability in multi-vendor home network systems (HNS), this paper presents a new framework for networked appliances, which extensively uses the concept of service oriented architecture (SOA).

We first propose a two-layered design of an appliance, consisting of device and service layers. The device layer corresponds to the physical device of the appliance controlled by vendor-specific interfaces. On the other hand, the service layer exhibits features of the device as self-contained services accessible via device-independent interfaces. Then, multiple appliances are loosely coupled by a service integration mechanism to provide value-added integrated services.

We evaluate the proposed framework with practical scenarios of evolution in HNS. We also implement a prototype home network system to show the feasibility in more practical setting. As a result, it is shown that the proposed framework is resilient for evolution of appliances as well as evolution of integrated services, which had been a limitation of the conventional reference-model framework.

Keywords: service-oriented architecture, interoperability, evolution, home network system, integrated services

1. Introduction

The recent ubiquitous technologies allows home electric appliances, such as TVs, lights, DVD players, refrigerators, air-conditioners and blinds, to be connected with a local area network at home. A system consisting of such networked home appliances is generally called a home network system (HNS, for short). Several companies have already released commercial networked appliances and HNS services (e.g., [14, 8]), such as remote control of the appliance outside home and group control of lights. Although networking home appliances could add value to our daily life, the current HNS services are not so sophisticated that are not far from the conventional remote control. Also, a HNS service is usually limited within a single or small number of single-vendor appliances.

The next step for the industries is to integrate and orchestrate appliances from multiple vendors flexibly, in order to provide more sophisticated value-added services for home users. We call such the next-generation HNS service as HNS integrated services in this paper. For instance, integrating a TV, a DVD player, speakers, lights and a blind would implement a HNS integrated service, say, DVD theater service. When a user turns on the DVD player, the lights become dark, the blind is closed, and the 5.1ch speakers are selected while the volume is automatically adjusted.

In order for the multi-vendor appliances to communicate with each other, a common network protocol is necessary for the HNS. For this, several protocols (e.g., DLNA [2], ECHONET [3]) are being standardized. Once a common protocol is given, the vendor of each appliance has to achieve two issues: protocol conformance and appliance interoperability. The protocol conformance is that each appliance must conform to the given HNS protocol. On the other hand, the appliance interoperability is that multiple appliances conforming to the same protocol must operate together to achieve the HNS integrated services.

In general, checking the protocol conformance of an appliance is not very hard, since the conformance testing can be performed within a (single) vendor of the appliance. However, validating the appliance interoperability is much more difficult, since it requires the actual integration of appliances under the collaboration of multiple vendors.

The conventional approach to achieve the appliance interoperability is that the alliance of the vendors determines a reference model with a rigorous specification, for each kind of appliances [9]. The rigorous specification minimizes vendor-specific implementation. Thus, each vendor can perform the interoperability testing in isolation, by checking if the developed appliance can work with the reference model. However, this approach lacks flexibility in changing
the reference model, which significantly limits the evolution of home network systems. Specifically, the reference model may prevent appliance vendors from implementing vendor-specific features exceeding the model. Also, the variety of integrated services is also limited within the reference model.

To enable the evolution of HNS with sustaining the appliance interoperability, this paper proposes a new framework for HNS appliances. The proposed framework extensively utilizes the concept of service oriented architectures (SOA) [17].

In order to apply SOA to the networked appliance, we first propose a two-layered design of an appliance, where each appliance consists of two layers: device layer and service layer. The device layer corresponds to the physical device of the networked appliance. On the other hand, the service layer is a software application on top of the former two layers, which is a core of the proposed framework.

The service layer exposes the features of the underlying device as services, based on the SOA principle. Specifically, the service layer aggregates the features of the appliance as self-contained services that are independent of other appliances. Also, for each service, the service layer exposes a platform-independent and strictly-typed interface, by wrapping proprietary device APIs with a certain SOA framework (e.g., Web Services [4, 23]). Thus, each appliance can conform to a platform(i.e., implementation)-independent protocol, which achieves conformance.

Next, we construct HNS integrated services by combining the (existing) services of appliances, specifically executing exported methods in a certain order. The execution of the integrated service is performed with an external service integration mechanism. Thus, scenarios of the integrated services are separated from the appliances.

We evaluate the proposed framework from several viewpoints of evolution in HNS. To show the feasibility to practical setting, we have also implemented a prototype home network system with the proposed framework. The evaluation demonstrates that the proposed method can significantly achieve high interoperability among appliances, which is quite resilient to addition or modification of the appliance devices and the integrated service scenarios.

2. Preliminaries

2.1 Networked Appliances and Integrated Services

A home network system (HNS) consists of one or more networked appliances, which are connected to a local area network. In general, each networked appliance has device control interfaces (i.e., APIs) by which users or external software agents can control the appliance via network. Thus, the appliance is supposed to own a processor, a storage (to store device applications or middleware), and a network interface to handle the API calls.

Controlling only a single networked appliance does not offer much added value compared to the traditional appliances [12]. The main advantage of the HNS lays in integrating the control of multiple appliances together. This yields value-added and more powerful services, which we call HNS integrated services in this paper.

The communication among networked appliances is performed by an underlying protocol. Assuming that multiple vendors participate in developing networked appliances, a number of protocols for the appliances have been proposed and standardized. Typical protocols include X-10 [24], HAVi [6], Jini [10], UPnP [22], ECHONET [3] and DLNA [2], although their purposes and operating layers may vary. In order for two or more appliances to collaborate with each other, basically the appliances have to conform to the same protocol, which we call protocol conformance.

Checking the protocol conformance is not very hard, since it can be performed by individual vendor of the appliance, independently. Tools for the conformance testing could be given. For instance, DLNA provides CTT [9](Conformance Test Tool) for digital audio/video appliances.

2.2 Example of HNS Integrated Services

For a more comprehensive discussion, here we introduce a simple example of HNS integrated services. In the example, we suppose a HNS consisting of three appliances (a DVD player, a TV, a speaker). We prepare the following two service scenarios of the HNS integrated services.

Auto-TV Service: When the user turns on the TV, the speaker’s channel is set to 2ch, and the volume of the speaker is automatically adjusted for the TV mode.

Auto-DVD Service: When a user switches on the DVD player, the TV is turned on in DVD mode, the 5.1ch speakers are selected, and the volume of the speaker is automatically adjusted.

2.3 Appliance Interoperability

Even if multiple appliances achieve the protocol conformance, it does not mean that the appliances can successfully work together to provide the HNS integrated services. Major reason is that the protocol conformance is validated by each vendor in isolation, which does not cover the combined behavior of multiple appliances. This issue is known as appliance interoperability.

A straight-forward way for validating the interoperability is to run and test the HNS integrated services thoroughly.
on the actual appliance implementations. However, it is difficult for the vendors to predict exactly which appliances are combined in a HNS by the individual home users. Worse in the multi-vendor environment, the number of possible combinations grows in exponential as the number of appliances. Thus, achieving complete interoperability is a quite difficult problem.

The conventional approach to circumvent this problem is to introduce a reference model for each class of appliances [2, 3] with rigorous requirements and specifications. For instance, let us consider a case with the networked TV and the networked DVD player with Figure 1.

Suppose that each of vendors A, B and C wants to develop a networked TV (TV_A, TV_B or TV_C) to achieve Auto-DVD service (see Section 2.2). While, each of D, E and F tries to develop a networked DVD player (DVD_D, DVD_E or DVD_F). Then, the alliance of the vendors determines a reference model for each class of TV (TV_ref) and DVD player (DVD_ref). Each vendor develops the product so that it conforms to the reference model. Thus, TV_A, TV_B and TV_C are supposed to conform to TV_ref. Therefore, if DVD_D or DVD_E, DVD_F is developed so that it can work with TV_ref, then DVD_D achieves interoperability with all of TV_A, TV_B and TV_C. Note that this interoperability validation can be performed by D without having TV_A, TV_B and TV_C.

The method of interoperation (e.g., how to turn on the appliance and how to change the inputs, etc.) is strictly predefined among the reference models. Hence, the HNS integrated services are implemented in details according to the reference models.

2.4 Problem on Conventional Approach

The conventional approach with the reference model currently gives a realistic solution. However, in the near future it will place serious obstacles in evolution of HNS, which are summarized as follows:

E1: Evolution of reference model

Since the reference model is tightly coupled with all the appliances, it is quite difficult to change or update the reference model. Any changes in the reference model force the vendors to update all the appliance products, which would corrupt the interoperability among the new and the existing appliances.

E2: Evolution of networked appliances

The reference model rigorously prescribes over the multiple vendors how the appliances should behave. Each appliance has to be strongly aware of how the appliance should be interoperated with other appliances. Also, all appliances conforming to the same model are regarded to be equivalent. These facts would significantly limit the vendor-specific features, which prevents the evolution of the appliances.

E3: Evolution of integrated services

The HNS integrated services are implemented in a ready-made form by appliance vendors or service providers, according to the reference model. Hence, developing integrated services requires vendor’s expert knowledge, and customizing the integrated services is limited within the framework of the reference model. Thus, the reference model would limit the possibility of future programmable services, where home users integrate arbitrarily appliances as they like and create their own integrated services.

3. Service-Oriented Framework of Home Networked Appliances

To achieve the appliance interoperability in a more flexible way, we introduce the concept of the service oriented architecture in the HNS.

3.1 Service Oriented Architecture

Service oriented architecture (SOA) [17] is a system architecture to integrate different systems distributed over a network. Each system exports own features to the network as a unit of service (a set of tasks, which is coarser than an object). Each service is self-contained, that is, a service operates independent of the context or states of other services. The internal logic and implementation are encapsulated in the system. The system exposes only interfaces of the service in form of strictly-typed exported methods.

A service user executes the remote exported method and gets desired results. This remote procedure call is performed by a standardized platform-independent framework. Once an exported method is deployed, its interface definition is not allowed to change. Therefore, the change in the internal service logic or service implementation platform do not influence the service user. Thus, the loose coupling between the user and the service is achieved. Web Services [4, 23] are widely known as a major SOA framework.
Figure 2 shows an example of SOA. A service user with the client application calls an exported method of Service A. Service A is implemented by tightly-coupled objects, which internally invoke another exported method of Service B. In this example, the service user uses an integrated service consisting of services A and B, which is depicted by a bold oval.

Since SOA facilitates the integration of distributed heterogeneous systems, much research and discussions have been conducted mainly in the enterprise systems (e.g., [7]). Our originality here is to apply SOA extensively to the home network system, where heterogeneous and multi-vendor appliances must collaborate to provide the HNS integrated services.

3.2 Assumptions on Networked Appliances

To achieve the SOA-based framework, we assume that each networked home electric appliance satisfies the following conditions.

Condition C1: Each appliance has device control interfaces that can be accessed by software (e.g., APIs).

Condition C2: Each appliance has a storage to store application software (server and device control application), a processor to execute the application, and a network interface.

These conditions might pose a bit expensive assumptions for the current appliance vendors, due to the cost of embedded devices. However, we believe that these will be standard features for next-generation home networked appliances. As for Condition C1, there already exist standards prescribing a detailed object template for each category of appliances [2, 3]. Condition C2 is justified by a fact that; the price and size of processors/memories are becoming reasonable to embed them in home appliances. Indeed, there already exist some commercial products involving Web applications, so that the users can configure and control the product from PC through a Web interface (e.g., [18, 20]).

3.3 Basic Principle

Our key idea is to apply SOA to HNS in order to eliminate the reference model, without loosing the appliance interoperability. If this is achieved, then it would be unnecessary to take problem E1 (see Section 2.4) into account. In the proposed framework, each appliance provides its features in a form of self-contained services. Here, the appliance can export any services without concerning how the services are used by other appliances.

The interfaces to the services are well-defined as exported methods by a standard SOA framework, such as Web services. Any exported method can be executed by external software agents via network, in such a way that does not depend on the implementation of the appliance. The vendor of each appliance can freely update and modify the implementation of the appliance, as long as the vendor does not change the interface definition. This would help to cope with problem E2.

HNS integrated services are implemented by extensively using the services provided by the appliances, with a SOA-based service integration mechanism. This allows vendors to separate the integrated services from implementation of appliances. A service designer just chooses a set of exported methods and specifies an execution order and logic. For this, the designer just has to know the interface definition of the exported method, but does not need to concern implementation or dependency among the appliances. With an appropriate assistance to understand the interface definition, even a home user would be able to design own services. This would help to resolve problem E3.

Figure 3 represents an overview of the proposed framework, involving the same set of appliances shown in Figure 1. As will be explained in the next subsection, each appliance has two layers: service layer and device layer. HNS integrated services are created for each combination of appliances, by combining services provided by individual appliances.

3.4 Two-layered Design of Appliance

To achieve the basic principle, we propose a two-layered design for networked appliances. Specifically, we divide each appliance into two layers: a device layer and a service layer, as shown in Figure 4.

The device layer refers to the hardware portion of an appliance. According to Condition C1, each appliance has a
set of APIs, by which a software application can control the appliance. The device layer can also involve a middleware to drive the APIs.

On the other hand, the service layer is implemented as a software application installed in the storage of the appliance (see Condition C2), which is a core of the proposed framework. The service layer aggregates the features of the appliance as a set of services, and exports the services to the network with exported methods.

As depicted in Figure 4, each service invokes a set of APIs of the underlying device layers, in order to implement a self-contained feature of the appliance. The communication between the service and the device layers may require vendor-specific procedures and/or proprietary protocols, which is depicted by dotted arrows in Figure 4. Then, the service layer exhibits the interfaces of the service as a set of exported methods (depicted by boxes on the services). The exported methods are opened to network with a strictly-typed interface definition. These methods can be accessed (executed) from external software in a device-independent manner, which does not depend on the underlying device implementation of the appliance. For this, we use a generic SOA framework such as Web services (with WSDL and SOAP/XML).

Our framework does not force the appliance vendors to adopt any reference model. Moreover, for a given appliance, how to implement the service layer is completely up to the vendor. Also, it is possible for the vendor to modify or update the device and the service layers, as long as the interface definition of the service is not changed (see Section 3.1). What must be achieved in the service layer are that;

**Requirement S1:** Each service must be self-contained, which does not depend on the context or states of other appliances.

**Requirement S2:** The service must be executed with device-independent communication using a standard SOA framework such as Web services.

**Requirement S3:** Once the service is deployed, its interface definition must not be changed.

These requirements are quite reasonable in the context of SOA.

Figure 5 shows an example design of the networked TV, DVD player, Speaker and Light. In the figure, we assume that TV_A, DVD_D, and Speaker_P are developed by different vendors A, D, P, respectively, with the proposed two-layered design. The role of each method is as read in the figure. We assume that for each exported method, a usage manual and an interface definition specifying the type of parameters and return value are given by the vendor.

**3.5 Service Integration to Achieve HNS Integrated Services**

In the conventional approach, the HNS integrated services are well-defined on the reference model. However, in the proposed framework, we realize the integrated services by integrating the (existing) appliance services. Specifically, we implement an integrated service by executing exported methods in accordance with a certain order. For example, let us implement Auto-TV and Auto-DVD services (see Section 2.2) with TV_A, DVD_D, and Speaker_P shown in Figure 5. Then, the following two sequences of method invocations can achieve an implementation of these integrated services. In the following, an invocation of an
exported method \texttt{m()}} of an appliance \texttt{App} is denoted by \texttt{App.m()}. Also, the sequences are called \textit{service scenarios}, denoted by \texttt{SS}$_1$ and \texttt{SS}$_2$.

\texttt{SS}$_1$: \textbf{Auto-TV Service} (with \texttt{TV$_A$}, \texttt{DVD$_D$}, \texttt{Speaker$_P$})

1.1 \texttt{TV$_A$.ON();} /*TV$_A$ is turned on */
1.2 \texttt{TV$_A$.InputSelect(TV);} /*Input is set to DVD*/
1.3 \texttt{Speaker$_P$.power(ON);} /*Speaker$_P$ is turned on*/
1.4 \texttt{Speaker$_P$.input(TV);} /*Input is set to DVD*/
1.5 \texttt{Speaker$_P$.channel(2);} /*5.1ch is selected */
1.6 \texttt{Speaker$_P$.volume(60);} /*Sound is set to 60db */

\texttt{SS}$_2$: \textbf{Auto-DVD Service} (with \texttt{TV$_A$}, \texttt{DVD$_D$}, \texttt{Speaker$_P$})

2.1 \texttt{DVD$_D$.on();} /*DVD$_D$ is turned on*/
2.2 \texttt{TV$_A$.ON();} /*TV$_A$ is turned on */
2.3 \texttt{TV$_A$.InputSelect(DVD);} /*Input is set to DVD*/
2.4 \texttt{Speaker$_P$.power(ON);} /*Speaker$_P$ is turned on*/
2.5 \texttt{ Speaker$_P$.input(DVD);} /*Input is set to DVD*/
2.6 \texttt{Speaker$_P$.channel(5,1);} /*5.1ch is selected */
2.7 \texttt{Speaker$_P$.volume(80);} /*Sound is set to 80db */
2.8 \texttt{DVD$_D$.play();} /*DVD$_D$ plays contents*/

To execute the exported methods over different appliances, we need a certain \textit{service integration mechanism}. For this, two different methods in SOA can be used.

The first method is to use a standard service integration framework in SOA, such as \texttt{BPEL4WS} [1]. The framework specifies an execution order and logic among distributed services, and integrate and execute them as a \textit{business workflow}. Employing such integration framework to our SOA-based appliances, the HNS integrated service can be executed as a workflow. In general, the integration framework operates on an external server with a given set of scenarios (thus, workflows) of integrated services [13]. Thus, as shown in Figure 6(a), the appliances are managed in a centralized manner. This figure illustrates that the user executes the integrated service \texttt{SS}$_2$: \textbf{Auto-DVD Service}, just explained above. In the figure, an arrow represents an invocation of an exported method, labeled by a number corresponding to a method index.

The second method is to delegate the service integration to the appliance themselves, so that the integrated services are executed in an autonomous and distributed way (as shown in Figure 6(b)). Thanking to the service layer, any appliance can be controlled via device-independent interface. Therefore, it is possible for each appliance to trigger autonomously any other appliances. In our previous work [15], we put an application module (depicted by an octagon in Figure 6) into the service layer of each appliance. The module dynamically selects and triggers an appropriate exported method, according to a given service scenario. The implementation details of the module are found in [15].

No matter which method we use for the service integration, a HNS integrated service can be implemented as a sequence of invocations of exported methods.

4. Evaluation

In this section, we evaluate the proposed framework from the viewpoints of interoperability and evolution in HNS. The evaluation is qualitatively performed in accordance with practical scenarios of evolution in HNS.

4.1 Upgrading Appliance Implementation

After releasing an appliance, a vendor might think to upgrade its middleware or service layer, in order to fix problems in the appliance. For this, as long as the service interface definition is not changed (See Requirement S3 in Section 3.4), the vendor can upgrade implementation of an appliance without affecting integrated services. This is due to the significant advantage of SOA.

4.2 Replacing Existing Appliances

When a user replaces an appliance with a new one with \textit{different service interfaces}, the existing integrated service scenarios have to be modified, so that the new appliance is accommodated to the existing scenarios. This modification is achieved by replacing the old methods in the service scenarios to the new ones, with keeping \textit{functional equivalence} among methods.

For example, let us consider \texttt{SS}$_2$ (Auto-DVD service) in Section 3.5. This integrated service is for appliances \texttt{TV$_A$}, \texttt{DVD$_D$} and \texttt{Speaker$_P$} shown in Figure 5. Suppose that we replace \texttt{TV$_A$} with a new appliance \texttt{TV$_B$} shown in Figure 7(a). Note that \texttt{TV$_B$} has different service interfaces from \texttt{TV$_A$’s}.
API

Figure 7. New appliances: TVB and Lightx

Then, SS2 has to be modified for the new combination of appliances:

SS2′: Auto-DVD Service (with TVB, DVDp, Speakerp)
2.1 DVDp.on();
2.2∗ TVB.setPower(ON);
2.3∗ TVB.setInput(DVD);
2.4 Speakerp.power(ON);
2.5 Speakerp.input(DVD);
2.6 Speakerp.channel(5.1);
2.7 Speakerp.volume(80);
2.8 DVDp.play();

In SS2′, an index with ‘∗’ represents a method replaced from SS2 for TVB. For instance, TVB.on() in SS2 has been replaced with TVB.setPower(ON) in SS2′, assuming that both methods invoke equivalent services. Note that the modification of SS2 is done loosely and locally, by replacing exported methods of the replaced appliances. It does not influence the execution order of the methods, or the behavior of DVDp and Speakerp.

Strictly speaking, there is no guarantee that a method can be always replaced with another equivalent one, successfully. However, from the viewpoint of SOA, appliances in the same kind would provide similar (not necessarily the same) services, although the implementation varies among individual vendors. We are optimistic that it is not very difficult for the service designer to identify the equivalence among primary services of the same-kind appliances. For example, every appliance should have a service related to power, and every TV must have a service selecting channels. Focusing this point, the proposed framework allows any combination of multi-vendor appliances to implement a common integrated service.

4.3 Deploying New Appliances in HNS

Each appliance is designed to be self-contained. Hence, even if a user deploys a new appliance in a HNS, it does not interfere the existing integrated services at all.

For instance, suppose that a user has already had integrated services SS1 (Auto-TV) and SS2 (Auto-DVD) in Section 3.5, with TVA, DVDp and Speakerp in Figure 5. Assume that the user deploys a new appliance Lightx of Figure 7(b) in the HNS. Now he wants to integrate the feature of the light to the existing Auto-DVD service, in order to implement a new service, say, a DVD Theater service:

**DVD Theater Service:** When a user switches on the DVD player, the TV is turned on in DVD mode, the brightness of the lights is minimized, the 5.1ch speakers are selected, and the volume of the speaker is automatically adjusted.

With the proposed framework, an implementation of DVD Theater service will be as follows:

SS3:DVD Theater Service (with TVA, DVDp, Speakerp, and Lightx)
3. 1 DVDp.on();
3. 2 TVA.on();
3. 3 TVA.InputSelect(DVD);
3. 4 Speakerp.power(ON);
3. 5 Speakerp.input(DVD);
3. 6 Speakerp.channel(5.1);
3. 7 Speakerp.volume(80);
3. 8 Lightx.ON();
3. 9 Lightx.setBrightness(5);
3.10 DVDp.play();

where the method invocations 3.8 and 3.9 are to minimize the brightness of Lightx.

Note that the integration of Lightx can be done easily by reusing the existing service scenario SS2. Indeed, SS3 is created just by adding two method invocations Lightx.on() and Lightx.setBrightness(5) to SS2. This is based on the fact that the addition of these exported methods does not interfere with SS2.

Thus, for the addition of new appliances, a service designer can rapidly create new integrated services, by making full use of the existing and new services the appliances. What required for the service designer are to understand the interface definition of each service, and to specify how to combine the services. We believe that with an appropriate interface for service creation, even home user can program his own service.

In summary, the proposed framework allows evolution of appliances with sustaining interoperability. First, our framework does not require any reference model, thus, the problem E1 in Section 2.3 is circumvented. Second, according to the discussions in Section 4.1 and Section 4.2, the proposed framework allows evolution of appliances with sustaining interoperability, which achieves the problem E2. Finally, it was shown in Section 4.3 that the evolution of HNS integrated services is feasible, which copes with the problem E3.
5. Implementation

To show the feasibility of the proposed framework in more practical settings, we have implemented a prototype of a home network system.

5.1 Appliances

We have implemented the following ten kinds of appliances using the proposed SOA framework: a DVD player, a TV, a speaker, a light, an illuminometer, a door (with a sensor), a telephone, an air-conditioner, a thermometer and a blind. We have used Java (Java2 SDK SE 1.4.1) for the implementation language. In our prototype, the device layer of each appliance was implemented virtually as a Java component (beads). In the service layer, each service was implemented as a Java class. For the service integration mechanism, we adopted the distributed architecture in Figure 6(b), based on the implementation method in [15]. We deployed the methods of each service as Web Services (Apache-AXIS 1.1), and we used Jakarta Tomcat for the Web server. Thus, the ten appliances were created as software applications, and installed on ten PCs, each of which emulates a networked appliance.

Then, we designed the following nine kinds of integrated services. These services are determined based on the actual HNS products [8][14]:

Auto-TV Service: When the user turns on the TV, the speaker’s channel is set to 2ch, and the volume of the speaker is automatically adjusted for the TV mode.

Auto-DVD Service: When a user switches on the DVD player, the TV is turned on in DVD mode, the 5.1ch speakers are selected, and the volume of the speaker is automatically adjusted.

DVD Theater Service: When a user switches on the DVD player, the TV is turned on in DVD mode, the blind is closed, the brightness of the lights is minimized, the 5.1ch speakers are selected, and the volume of the speaker is automatically adjusted.

Coming Home Light Service: When the door (sensor) notices that the user comes home, the light is automatically turned on. Then, the illumination of the light is adjusted to the optimal value based on the current degree obtained from the illuminometer.

Coming Home Air Conditioning Service: When the door sensor registers that the user has come home, the air-conditioner is turned on, and its temperature setting is adjusted to the optimal based on the current degree of temperature provided by the thermometer.

Ringing and Mute Service: When the telephone rings, the volume of the speaker is muted.

Blind Service: When sunlight is available, the blind is opened.

Sleep Service: When the user goes to bed or goes outside, all appliances are turned off.

Auto Illumination Service: When the user turns on the light, the illumination of the lights are adjusted to the optimal value based on the current degree obtained from the illuminometer.

We have also developed an XML-based language to program each integrated service scenario. Using the language, the service designer can write a service scenario involving parameter of exported methods as well as the execution order among the methods. The script is uploaded to the service layer of each appliance, so that the appliances autonomously collaborate to achieve an integrated service.

5.2 Integrated Service Builder

We have also implemented a support tool, called integrated service builder, to create and monitor the integrated services within our prototype. A screenshot of the tool is shown in Figure 8. It was implemented by Java Servlet and Macromedia Flash (Flash MX 2004).

The integrated service builder supports a service designer to build and execute the integrated services efficiently. Using the builder, a service designer can deploy or undeploy any appliances flexibly. Also, the designer can create any integrated service in the XML-based language mentioned in Section 5.1. The tool also provides GUI to execute the integrated services created, and to monitor each appliance so that the service designer can check if the integrated service is implemented as desired. More details of the tool are found in [11].

Using the prototype system and the integrated service builder, we have conducted several case studies. The case studies simulate the evolution of HNS according to the scenarios in Section 4. As a result of the experiment, it was confirmed that all the integrated services can be implemented successfully using the proposed framework.

6. Discussion and Concluding Remarks

This paper has presented a service-oriented framework for networked appliances in HNS. The proposed method extensively used to enable flexible evolution of HNS with sustaining interoperability. The framework was evaluated qualitatively with the scenario of evolution. We also implemented a prototype system and a support tool to show the feasibility to the practical settings.
The current limitation in the proposed framework is that the networked appliances satisfying Conditions C1 and C2 (see Section 3.2) seem to be so expensive that we have not yet found actual products. However, we believe that this limitation will be resolved in the near future as the price of devices decreases.

We are currently developing tools to support non-expert users to construct integrated services easily. As mentioned in Section 4.3, to design integrated services, a service designer has to first choose appropriate services for each appliance, and then understand the service interface definitions. This would impose a tough task for non-expert users. To facilitate the service creation, we are trying to utilizing service discovery technology such as UDDI [21], which could automate major portion of the service creation. It is also interesting for us to employ ontology [5] to manage systematically manage common services among multi-vendor appliances.

References


[18] PLANEX COMMUNICATIONS Inc., BRC-14V
   http://www.planex.co.jp/product/broadlanner/brc14v.shtml


[21] UDDI (Universal Description, Discovery and Integration of Business for the Web) - http://www.uddi.org/

