Client Side Personalization of Smart Environments

Ibrahim Armaç
Computer Science 3 (Software Engineering)
RWTH Aachen University
Ahornstr. 55, 52074 Aachen, Germany
armac@i3.informatik.rwth-aachen.de

Daniel Evers
Computer Science 3 (Software Engineering)
RWTH Aachen University
Ahornstr. 55, 52074 Aachen, Germany
evers@i3.informatik.rwth-aachen.de

ABSTRACT

In this paper we will describe our approach for supporting users by personalizing the multiple environments they visit in their daily lives. In our approach, user preferences are stored on a handheld device which can assist the user in selecting, installing and parameterizing personal services in the environments.

Categories and Subject Descriptors

D.2.11 [**Software Engineering**]: Software Architectures—*Domain-specific architectures*; J [**Computer Applications**]: Miscellaneous

General Terms

Design

Keywords

Smart environments, personalization, mobility, OSGi, JXTA

1. INTRODUCTION

In this paper we will describe our approach for supporting mobile users in personalizing smart environments, in particular smart homes or, as we call them, *eHomes*. These are environments equipped with devices and appliances, which are usually controlled by a gateway. The gateway acts as a runtime environment for basic and integrating *eHome services*. *Basic* services, such as a lamp driver, do not use other services to realize their own functionality. In contrast, *integrating* services are composed of various basic and, optionally, other integrating services. Integrating services are called *top-level* services if they are not used by other services. We further declare top-level services as *personal* if their functionalities depend on user preferences.

One of the main challenges of developing context-aware eHome services, which strongly depend on user location, is the user's mobility. There are two kinds of mobility important to us: *in-home* and *inter-home* mobility. In this paper, we will focus on the inter-home mobility: As Figure 1 denotes, users do not stay in a single environment all the time. Instead, they visit different environments in their daily lives, such as their workplace, hotels, friends' homes etc. How can we support users in personalizing visited environments by using personal services, assuming that the environments provide the necessary infrastructure? Usually, users have to enter their preferences manually for each environment, which can be a tedious task. In this paper we will describe our solution for client-side personalization based on a personal mobile device (called *handheld*).

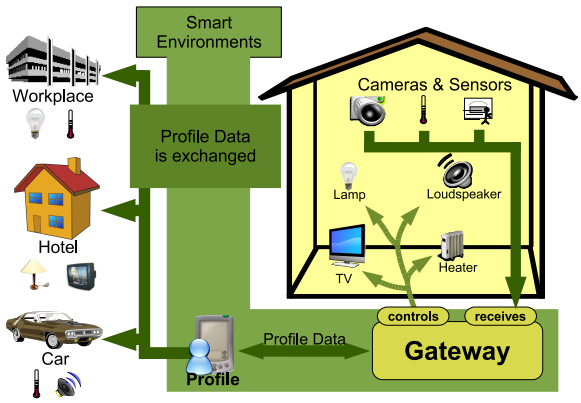

Figure 1: Inter-home mobility

A simple example of a personal service which could be used in different environments is the `Music-follows-Person` service. Combining multimedia devices and person detection, this service lets a music stream associated with a specific user follow this user through the home. Other similar examples of services adapting environmental variables to user preferences are `Light-follows-Person` or `Temperature-follows-Person`.

2. PROBLEM DEFINITION

The following problems have to be addressed when personalizing multiple environments.

Consider that a user already uses the abovementioned personal services in his home. Now, if he wants to use the same services also in his holiday flat where he spends his weekends and these services are not installed there already, the user has to select, install, and parameterize them manually. However, to repeat these steps for each service in every environment is not feasible.

SAM'08, May 10, 2008, Leipzig, Germany.

Similarly, if the services are already running in the environment but the user comes there for the first time (e.g., an hotel room or a friend's home), the services would have to be parameterized with the user's preferences, again manually (the permission of the host is assumed). Because the environments do not share user preferences, the settings are stored separately in each environment. As a result, if the user changes his preferences, he has to propagate these changes to every environment manually. This will apparently discourage the user.

3. CLIENT SIDE PERSONALIZATION

The approach presented here is based on the idea that the user carries a handheld. This device can discover the presence of an eHome by detecting a corresponding wireless network and notifying the user about it. Furthermore, it can assist the user in performing tasks such as selecting, installing and parameterizing eHome services.

All communication with the eHome services as well as the eHome's management facilities is based on well-defined interfaces. This ensures that the handheld knows all methods of the remote objects it communicates with, so that they can be used like local objects, similar to Java's RMI.

3.1 Motivation for Using a Handheld

Todays mobile devices are multifunctional. In our approach, we utilize the user's handheld (e.g. a cell phone or PDA) for following purposes: (1) As a unified remote control for smart environments, (2) as a platform assisting the user in performing personalization tasks and (3) as a mobile storage device for user preferences.

Our work is focused mainly on the purposes two and three, i.e. using the handheld to assist the user in personalizing the visited environments. This is achieved by storing the user preferences on the handheld, to be available when needed; secondly by providing interfaces to the eHome's management facilities, so that the handheld can select, trigger the installation of and parameterize services based on user preferences.

Furthermore, the handheld can be used for indoor localization as well as for the execution of top-level services.

3.2 Service Specifications and Interfaces

In order to provide automatic service selection by the handheld, the user has to take along the specifications of his personal services. In particular, a service's specification contains the list of its provided functionalities, describing the service's behavior. We describe functionalities by so-called "functional labels" [5]. E.g., the functionality of the `Heating` service can be described by the label `Heating`. If this service can be parametrized to regulate the temperature based on the current time, the specification can contain the label `TimedHeating`. It is also possible to order functional labels hierarchically, so that e.g. `TimedHeating` always includes `Heating` and extends this functionality.

To make services with the same functionalities interchangeable, we require that every label is associated with a unique interface, which has to be implemented by each service providing this functionality. This allows the usage of a service based on its specification, independent of its implementation. Thus, services with different names and implementations can be exchanged, as long as they provide the same functionalities.

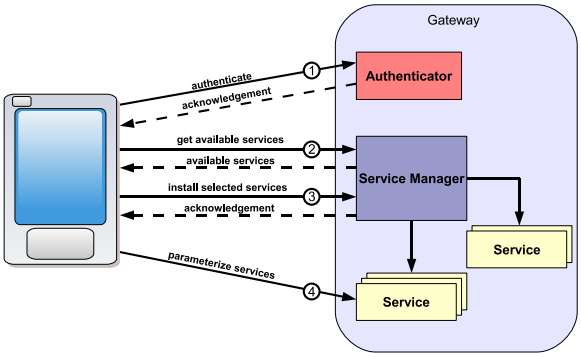

Figure 2: Interaction handheld ↔ environment

3.3 Automating Service Selection, Configuration and Deployment

For access to the environment's management facilities and to allow a semi-automatic configuration by the handheld, the environment has to provide amongst other components an Authenticator and a Service Manager (see Figure 2). The *Authenticator* is responsible for user authentication and access control. The *Service Manager* provides access to all service specifications and instances. The handheld can retrieve a list of personal services which the user is allowed to use. Depending on the user preferences, the handheld can request the Service Manager to install and/or instantiate selected services for the user. Every instance of a personal service is responsible for one user's preferences.

The automatic parameterization of the selected (personal) services is the last necessary step in personalizing the environment. The parametrization can be automated if preferences corresponding to the selected services are stored on the handheld. Like functional labels, we assume that parameters also adhere to a consistent naming schema, where each parameter name is unambiguously defined.

3.4 Parametrization and Service Interfaces

Figure 3 gives an example of how the preferences on the handheld are stored. The user created preferences for temperature, light intensity and music. He has also stored the specifications of the personal services he would like to use in visited environments: `Temperature-follows-Person`, `Light-follows-Person` and `Music-follows-Person`. These services are running in his home environment along with other (non-personal) services, such as `TV` and `Alarm`. If the user now visits the environment on the right side, the two personal services running there would be instantiated and parameterized for the user.

Each personal service implements a method which returns a list of personal parameters. These imply getter and setter methods for each parameter; using them allows personalization of the service. For example, a heating service may have the parameter `preferredTemperature`. This implies the methods `getPreferredTemperature()` and `setPreferredTemperature()`. This way, the parametrization of personal services can be fully automated. Furthermore, user changes on preferences are automatically propagated to the corresponding services. As a result, the user is required to change his preferences only once. They are then transferred for every environment visited in the future.

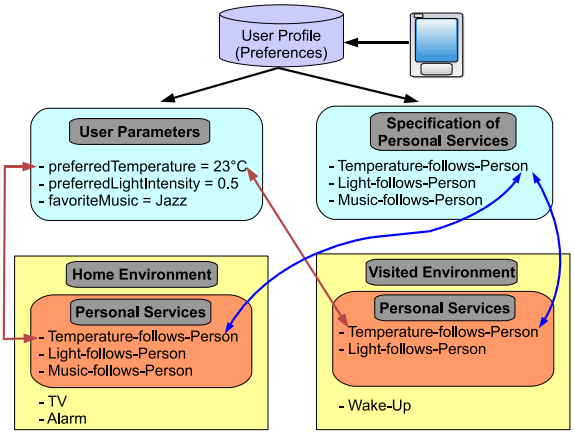

Figure 3: Example setup

3.5 Evaluation

We evaluated the handheld software on a PDA, a Dell Axim X51v, capable of Wireless LAN, in combination with our existing eHome prototype. This prototype contains a 2D simulation environment, the eHomeSimulator [1], which can be executed on PCs to simulate different smart environments. The implementation is based on Java on top of the Eclipse Embedded Rich Client Platform, which uses the OSGi framework for providing a plug-in based architecture. OSGi provides a SOA-based runtime environment for services and applications. The communication on top of the wireless network is based on JXTA, a language-independent P2P protocol. We implemented our own RMI-like communication over JXTA, called "SimpleRMI". We used IBM's WebSphere Everyplace Micro Environment as the Java Virtual Machine.

For communication, we set up a WiFi network using a standard router without encryption. The eHome discovery is done by the JXTA framework. It searches the network when the wireless interface connects to an access point. The existence of a JXTA peer group named "eHome group" indicates the presence of an eHome gateway. After the handheld joined this group, the user is notified that an eHome was found. He can then decide to authenticate or not.

We tested the three personal services depicted in Figure 3. After the authentication of the user using the PDA, the eHomeSimulator adapted the light intensity, the temperature, and the music in the current room to the user preferences on the handheld. Preference changes (e.g. temperature) cuased the eHomeSimulator to adapt automatically to the new values in the corresponding room. We have also developed UIs for the handheld. Thus, the remote control of the eHomeSimulator is possible.

4. RELATED WORK

The authors of [4] interconnect multiple OSGi gateways, based on JXTA, to create a *Virtual Home Environment*. This approach focuses on enabling users to access their home devices remotely; in contrast to us, personalization of visited environments is not considered.

Gaia [6], is a CORBA-based middleware operating system for active spaces, which are similar to our smart environments. In contrast to Gaia, we do not require the smart environments to be connected together to migrate applications or services. Furthermore, in our work the users take along their preferences on a mobile device, allowing the personalization of visited environments without connection to a centralized infrastructure. This is required in Gaia for application mobility.

There exist also further research projects about context-aware smart environments, such as the *MavHome Project* [7], *Nexus* [3], the *Adaptive House* [2] etc. These projects focus on similar aspects, such as comfort enhancement and adaptation to the behavior of known users in single smart environments based on different technologies, such as agents or artificial intelligence. In none of these projects we found work supporting mobile users in client-based personalization of visited environments.

5. CONCLUSION AND OUTLOOK

In this paper we have discussed the client-side personalization of smart environments. Our approach is based on handhelds which assist the user in selecting, configuring and installing services. The user preferences are on a handheld and therefore available everywhere. The handheld can also assist the user in setting up personal services corresponding to his preferences and can act as a UI to the environments.

The presented solution provides starting points for further improvements. Some examples are standards for service specifications and corresponding interfaces (e.g. based on ontologies), service execution on handhelds, synchronization between the profile manager on the handheld and on the environment gateway, as well as conflict management for multi-user conflicts. Furthermore, security and privacy aspects have to be considered. We are working on negotiation-based authentication using anonymous credentials and role-based access control.

6. REFERENCES

[1] I. Armac and D. Retkowitz. Simulation of Smart Environments. In *Proc. of ICPS 2007*, pages 257–266. IEEE Press, 2007.

[2] D. J. Cook and S. K. Das, editors. *Lessons from an Adaptive Home*, chapter 12, pages 273–298. Wiley, 2004.

[3] F. Dürr, N. Hönle, D. Nicklas, C. Becker, and K. Rothermel. Nexus–A Platform for Context-Aware Applications. In *Informatik-Bericht der FernUniversität Hagen*, pages 15–18, London, UK, 2004. Springer-Verlag.

[4] C. Loeser, W. Mueller, F. Berger, and H.-J. Eikerling. Peer-to-Peer Networks for Virtual Home Environments. In *Proc. of HICSS 2003 - Track 9*, page 282.3, Washington, DC, USA, 2003. IEEE Computer Society.

[5] U. Norbisrath, I. Armac, D. Retkowitz, and P. Salumaa. Modeling eHome Systems. In S. Terzis, editor, *Proc. of MPAC 2006*, pages 1–6, New York, NY, USA, 2006. ACM Press.

[6] M. Roman, C. K. Hess, R. Cerqueira, A. Ranganathan, R. H. Campbell, and K. Nahrstedt. Gaia: A Middleware Infrastructure to Enable Active Spaces. *IEEE Pervasive Computing*, 1:74–83, Oct-Dec 2002.

[7] G. M. Youngblood, L. B. Holder, and D. J. Cook. Managing Adaptive Versatile Environments. In *Proc. of PERCOM 2005*, pages 351–360. IEEE Computer Society, 2005.