A Middleware Platform for Application Configuration, Adaptation and Interoperability

A. Uribarren[1], J. Parra[1], R. Iglesias[1],J.P. Uribe[1] and D. López-de-Ipiña[2]
[1] *Software Technologies Area, Ikerlan – IK4*
{auribarren, jparra, riglesias, jpuribe }@ikerlan.es
[2]*University of Deusto*
dipina@eside.deusto.es

Abstract

Managing sensors, actuators and devices for supporting context-aware applications poses great challenges, such as the efficient coordination and cooperation among them, their self-configuration, as well as the management of their interactions.

This papers aims at creating a middleware platform for the provision of the following functionalities: Configurability, Adaptability, Heterogeneity and Interoperability (CAHI). The capabilities of this middleware platform are assessed by means of a smart-home scenario populated by different sensors, actuators and devices.

1. Introduction

Recent advances in communication infrastructures, computational resources and computing devices have paved the way for the development of pervasive computing environments. They are enabling a broad range of promising applications ranging from environment monitoring, smart homes, ambient intelligence in health care, or mobile multimedia applications.

The virtue of these environments is that the provision of device functionality and local and distributed software applications should be given in a flexible, integrated and almost transparent way for end-users [1]. However, the management of the state data and services coming from heterogeneous sensors and/or embedded and mobile devices everywhere and at all times is fairly difficult to be managed by end-users and appropriate systems should deal with it.

On the other hand, pervasive applications need to be aware of changes (i.e. network or device resources, devices that are removed, new devices introduces, devices that change their configuration depending on the services available) or user desires/actions in the environment, and adapt themselves to these changes. Context awareness plays an important role in applications in order to sense and react accordingly in the environment to provide high-quality and reliable data to applications. To achieve it, more research is needed on building software infrastructures to process the sensed information, providing more reliable services, adaptation or enabling interoperability of devices and applications, among others.

Nowadays, technology seems to be mature enough to build pervasive environments, in turn, the reality is very different. The majority of the devices are isolated offering a specific service and they do not work or collaborate together to achieve the application goal [2]. Several protocols have been developed for such a purpose, for instance, UPnP (Universal Plug and Play) [3], Jini [4] and HAVi (Home Audio and Video interoperability) [5].

To our knowledge, a middleware platform should deal with, communicating devices and applications, gathering context information and services from heterogeneous physical sensors and processing it. These processes should be transparent to applications, and the applications should adapt to the existing context configuration by using the processed information. Adaptation strategies to the ever-changing environment and user needs are important.

In this paper, we present a middleware platform whose function is to provide interoperable mechanisms of communication between applications and devices independently of the device APIs or protocols and context-based coordination, cooperation and adaptation of the services provided. This framework intends to be installable and run-able independent of the underlying hardware, operating system and of the application itself.

2. Requirements and challenges

978-0-7695-3553-1/08 $25.00 © 2008 IEEE
DOI 10.1109/SASOW.2008.36

The vision of Weiser's ubiquitous computing corresponds to a new paradigm in which computing systems are seamlessly integrated into everyday life [1]. With this new paradigm the difficulty of developing new applications, which should adapt to the environment with minimum human intervention has arisen [6]. Applications that should be adapted according to user position, the presence of more people or devices or user activity, among other contexts.

A pervasive environment should be aware of user presence, sensitive, adaptive and responsive to user desires, habits, emotions or activity [7, 8]. To fulfill these requirements, pervasive applications should provide: ubiquitous access, context awareness, intelligence and transparent interaction to users [7]. Therefore, these applications need middleware to interface between heterogeneous devices and applications [8]. It should hide complexity to applications, such as, management of protocols, communication failures or heterogeneity problems [8, 9].

In short, middleware should deal with [2]:

- Configurability: changing the configuration of applications depending on the services or network capabilities available.
- Adaptability: it should detect and deal with changes in the environment. For instance, detecting new available devices/services, failures or user movements, although we believe that some changes still need to be carried out by applications.
- Heterogeneity: given a diversity of devices (i.e. cheap/expensive, mobile/stationary), middleware should be scalable enough to fit types.
- Interoperability: enables devices in a heterogeneous networked system to work together to achieve the application goal.

Therefore, the middleware platform proposed, which is called the CAHIM platform, will attempt to address these four requirements.

3. Related work

Since this paper research covers a diversity of different areas, this section describes a non-exhaustive selection of related work that, to our knowledge, is more relevant to this research work.

Many existing middleware approaches have focused on how to endow mobile applications with adaptation, reconfiguration and interoperability to changing resources and changeable environments [10-15]. The research work described in [11] proposes a way to provide information related to resources such that the application interface is unique, WSDL (Web Services Description Language [16]). However, complete application development abstraction can only be achieved if service discovery and interaction are not limited to a concrete definition language, not forcing application adaptation. Another example is the work carried out in the ReMMoC project [10]. Although the ReMMoC middleware enable applications to use simultaneously different discovery and interaction protocols, it still requires the environment to be monitored to allow ReMMoC to detect the SDPs (Service Discovery Protocols) over time and interaction protocols that need be supported / integrated, due to the very dynamic nature of the mobile environment. However, this research focuses on interoperability between mobile clients and existing middleware applications, rather than on portability, adaptation and self-configuration of components across the middleware platform. The DRACO project aims to simplify the development of pervasive and ubiquitous environments [12]. In this research, a distributed runtime infrastructure is used to distribute software components on heterogeneous, networked and embedded hardware systems. A Java-based micro kernel runtime environment runs on their devices to overcome heterogeneity.

A similar approach can be found in [13] where a novel mechanism to make reflective middleware is presented. The usage of reflection enables adaptation and reconfiguration of systems at run-time, allowing the system's behavior to be altered at run-time to better match the system's dynamic operating environment. The cross-platform interoperability is also achieved using appropriate platform packages or programs for Java or .NET. A major drawback of this approach is that it is required a previous installation of the needed run-able and platform-dependent engine for installing portion of code in each device. Despide this fact the data transfer and applications exchange will be based on this research work.

Another middleware, in which this middleware platform has been based on, was developed in the AMIGO project [14, 15]. This middleware aimed at enabling several pervasive applications within a networked home environment by seamless integration of heterogeneous service technologies. This integration was carried out by using an abstract "unified service model" describing the service and providing different interaction protocols to be called by pervasive services and applications. The interoperability with concrete services upon different middleware platforms is achieved by a "bridge layer" which maps the abstract model to concrete services. This paper is an extension of [14] and [15] where the main challenge to be addressed is to deal with self-

configuration and context-awareness by abstracting context-aware applications from devices that provide context. Hence, a Control layer for data/messages analysis and management between ideally any upper high level applications and locally controlled devices will be introduced.

4. The middleware platform: CAHIM

We regard the system presented here as a core low-level service for pervasive and ubiquitous computing applications that operate in a mobile environment. Many existing approaches formulate means to deal with device information in general terms. These approaches look at aspects such as collection, distribution, transformation and inference of generic information. Here, one of the main scopes is to communicate context information, and focuses on how to allow components running on different devices to be aware of their network and related resources so that they can adapt and provide a pervasive service to upper applications. This feature will be covered and extended later in Section 4.2 "Context-aware service management".

Furthermore, by learning from the solutions and mechanisms found for a specific domain, it is useful to better understand and recognize the valid principles of generalized forms for environment handling. It is also crucial that solutions are generic that is, independent of the underlying hardware, operating system and applications involved. Moreover, solutions should not require complex modifications for each application. It is important to endow applications with a mechanism to provide hardware and software interoperability.

Two aspects need to be addressed: the access to inter-connected devices at anytime and anywhere and the coordination and adaptation of available services.

4.1. Interoperability support

An approximation to support plug & play capabilities can be found in our previous research work [14]. Here, the same mechanism is used; however it has been extended to provide adaptability by means of reconfigurability capabilities. Following a summary description of the approach is described (see [14] for further details).

The main goal of the previous middleware platform was to abstract applications from services provided by heterogeneous devices that use diverse technologies. Some authors use precise integrations between specific protocols (e.g. [17], [18]), while others have developed a middleware to integrate any standard in a well-defined protocol (i.e. [19]). We have followed the first approach, so in that way our middleware will not restrict developers to use a particular service discovery protocol. The adopted solution consists of extracting functional description of available services in heterogeneous networks, representing it in a unified syntax and offering such services to applications by using some standardized protocols.

4.2. Adaptability, Coordination and Configurability support

In order to support adaptation, coordination and self-configuration of services, a new Control layer (Fig. 1) has been added to our previous work [14]. This Control layer is comprised of a Coordinator component, a User Profiling component and a Context Information component.

The Control layer is responsible for maintaining the current state of the middleware platform online and updated and of providing a mechanism to allow reconfiguration based on current or stored status information, as well as on the rules already defined.

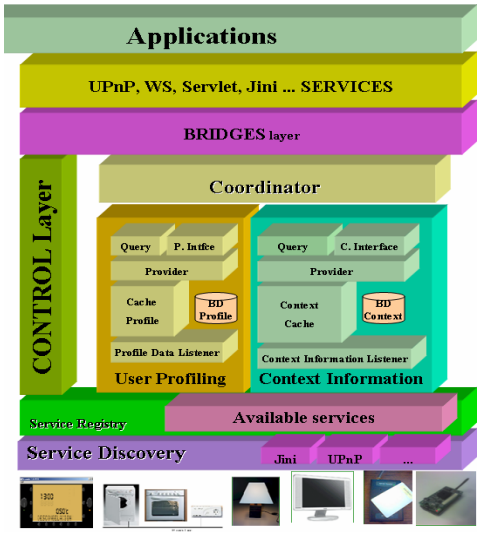

Fig. 1. Control layer view

The Context information component is in charge of maintaining the current state of the available devices and services in the platform. Moreover, it stores data concerning previous status changes of the devices. It stores any kind of information related to devices and communication states to provide basic information to the Coordinator in order to achieve the requested and defined behavior providing the needed information to upper applications. The User profiling component is similar to the Context Information

component but containing user data. The user and context information will be served to the Coordinator, so it can make a decision or take an action in case of conflict or new situations.

The Coordinator component plays an important role. It is a monitoring engine with some associated goals and predefined rules. It orchestrates between the involved devices in order to satisfy the goals of the upper applications. For example, if an upper application requests 'switch light on' in a room to the Coordinator, it checks which device is able to provide light in that room. The device identified as 'lamp' will be commanded by the Coordinator to be switched on. On the other hand, and according to user preferences, with the information received from the device identified as 'luminosity level reader' it checks if the requested light level is achieved or not, and if not, for example, the coordinator will perform a pre-recorded list of actions, opening the louvers in this case before giving a positive or negative 'action done' answer to the upper level application regarding the original request. The Coordinator is responsible for mediating and providing an adequate answer to the request of upper applications and to do that, it is able to adapt the services in each moment based on the Event-Condition-Action or ECA rules.

The deployment of the Coordinator component in the middleware platform can be distributed and distributable as well. In that sense, this component can be fragmented as required and each piece or subcomponent can be shared among the middleware's involved devices.

5. The role of the platform in smart-homes

Following, a smart-home scenario is introduced which serves as a proof-of-concept for the proposed platform. This example shows different pervasive computing applications in a smart-home environment.

5.1. Smart-home scenario

Ann is watching channel A on TV at the living room. It is raining, there is a storm. She starts the Main Home Controller application (MHCA) by means of the television screen. She can interact with MHCA via voice or by using a remote controller. She chooses to close all the windows at home. She can now continue watching television.

She is a bit hungry and she goes to the kitchen to prepare some popcorn. The home indoor location system detects her movement and switches the kitchen TV on and channel A is automatically selected. During that process Ann has switched the lights on, because due to the storm it is getting darker. Soon after, the MHCA shows an alert window on the kitchen TV. There is a problem with the bathroom window. The automated system cannot close the window. A new option is given to Ann: the system can close the louver. Ann accepts but she decides to go to the bathroom to see what happens with that window. She takes a PDA with her and goes to the bathroom. In that moment, the system detects that she is leaving the kitchen with the PDA and a lightweight version of the MHCA is installed in the PDA. On the other hand, the TV show now starts being played on the PDA. Once she enters the bathroom, the system detects that the luminosity is low, and as a consequence, the home automation system switches the light on. The problem is that the curtain impedes to close the window. Once she removes the curtain from the window, the window is closed by an automatic order from the MHC and the louver is open. Ann starts her way to the living room. The bathroom light is automatically switched off and a beep sounds in the PDA with a new message displayed to let her know that the popcorns are ready. She takes the popcorns and sits down on her sofa. She switches the PDA off and TV is switched on and shows Channel A. The ambient light has been also adjusted to her previous preferences.

5.2. Features to fulfill the requirements of the smart-home scenario

In the previous section, a smart-home scenario has been shown. In this section, that scenario will be used to identify and clarify how the proposed framework provides self-configurability and context-aware adaptability to the MHCA. The requirements of the MHCA are covered by the proposed framework and the main features of the framework will be emphasized.

It should be noted that, on the one hand, in this scenario it is possible to discover any device at home and they can interoperate by means of the interoperability mechanism in the proposed framework (UPnP Bridge has been used [14]). On the other hand, such devices' software is hot-upgradeable and the solution provided in [13] to support mobility and reconfigurability has been used. By means of the Control Layer, it will be shown that it is possible the self-configuration of different devices and adapt device behavior according to current context.

5.2.1. MHCA and TV. In this scenario, the MHCA is a high level application that uses the

middleware capabilities to perform the desired tasks. Ann wishes to use the MHCA, which is running somewhere at home, on television. First, the system checks whether or not there is a user interface installed to use the MHCA (MHC_UI). If it is not installed, MHCA searches and copies the required software from the Home Application Repository (HAR), and it installs and starts this application on TV.

It is considered that TV and in general other computing devices are resource limited and sometimes, it is required to unload/uninstall applications in order to load/install others (i.e. a game application, a music player). The proposed framework takes care of this mobility software issue, with application independence.

Different computing devices [20] may operate on a generic hardware platform to control and manage any kind of device (i.e. TVs, fridges, ovens, lamps, louvers and so on) and software can be attached and installed in these devices, if required, by using the proposed framework. Therefore, TV can perform the role of MHC_UI, a game player or an internet browser by loading/unloading the run-able components (using a reflection mechanism). They are adapted according to user needs at each period of time.

From the TV, Ann selects the option of closing windows in the MHC_UI. Then, MHC requests for closing all automatic windows. The windows have automatic locking/opening, moreover they can provide their status to other systems. In that sense, once the action is performed, the MHCA should receive acknowledgement and the pending request is deleted from the Message Delivery Controller (MDC). If there is a problem, a warning message or an alternative service can be offered to Ann. In this case, a warning message (an adaptive solution) 'all windows were closed except for a window in the bathroom' is sent to the active MHC_UI to show it on television. Although there are some automatic actions, in some cases, the MHCA requires user intervention. We believe that user intervention cannot be totally avoided to fulfil user expectations. Another example, it is requested to Ann whether to move the MHC_UI to the kitchen TV.

So far, we have seen different components of the MHCA to attach, install, send and show to Ann, showing the adaptation of the application.

Later on, when Ann removes the curtain, the MHC can request for closing the window properly. An 'isClosed' message is sent to the MHC and it is unqueued from the MDC. Moreover, the MHC will close the window (the original request) before opening the louver. It is somehow an automatic system recovery, getting back to the previous well-known or desired state to be fulfilled. The MHC already stored the state before changing the main goal and as a result it can roll back until the desired one: opening the louver.

5.2.2. Notifying the user and Lighting, Switching on/off. With regard to the PDA two main aspects are remarkable. The first one is the fact that how the MHC application is installed and executed on it. The MHC needs to check whether or not the MHC_UI is already installed in the PDA. If not, it checks if there are enough resources to install and run a specific version. Regarding the MHC for PDA's (or in general for any handheld device) there are different versions stored in the Home Application Repository. So the main system checks the availability in the PDA and depending on that it will decide to install full or customized version. In this case the main restriction used to decide has been the free available memory.

The second aspect is how TV is shown on the PDA in her way to the bathroom. We have not TV tuner on the PDA. In the kitchen TV when Ann moved, the MHCA has switched TV on and tuned the desired channel to view. To watch TV on the PDA, first the MHCA has detected that the destination of the video information needs to be broadcasted; PDA hasn't got a TV tuner, so in consequence, MHCA has started the home entertainment multimedia set box and has selected the desired channel to broadcast the TV show as multimedia streaming. On the other hand, it has started a media player on the PDA and has configured to connect to the media server, to receive the online video content. So, the MHCA, although the goal was to show TV, depending on the requirements of the device, needs to adapt and reconfigure components to provide the same TV service.

There are other actions involving the cooperation of different devices and components. For example, notifying Ann that the popcorns are ready to eat, the microwave device sends a message to the MDC. The coordinator deals with it and sends an order to the current user interface. If the user interface is being used through TV, the message will be directly shown on TV screen. In any case, the user will be notified in a different way (i.e. vibrating, beep sound, notification balloon) depending on the user context.

The coordinator also deals with other actions related to user presence. The coordinator automatically adjusts the lighting level according to user location and by switching lights on/off if required. Moreover, devices, like TV, are put on/off depending on user presence. RFID-based locators are used for user location. TinyOS operating system enabled by Crossbow's MicaZ sensors[21] are used

for monitoring lighting level and many different automated lamps (i.e. with RS232 controlled lamps or lamps based on the BDF home automation bus [22]). These types of actions are supervised by the MHCA and the coordinator is in charge of following up the correct execution by the corresponding device.

6. Conclusions and future work

In this paper, a middleware platform has been introduced as a mechanism to provide flexibility to pervasive applications. This application flexibility refers to challenging issues, such as, configurability, adaptability, heterogeneity and interoperability. This platform attempts to address new challenges in user-centered and ever-changing pervasive environments. Although it has been shown the validity of the middleware platform in a smart-home environment, it can be generalized to other pervasive applications.

As per future work, we believe that intelligent software agents can facilitate the deployment of context-aware pervasive application, and we plan to evaluate the use of agent technologies in the Coordinator component. In this sense user behaviour patterns could be applied for anticipating to the user's future needs and actions, improving the human-system interaction.

7. References

[1] M. Weiser. The computer for the 21st century. Scientific American, 256(3):94-104, 1991.

[2] E. Aarts and S. Marzano, eds. The New Everyday: Visions of Ambient Intelligence, 010 Publishing, Rotterdam, The Netherlands, 2003.

[3] UPnP Forum. Universal Plug and PlayTM Device Architecture, July 2008. http://www.upnp.org/.

[4] JINI technologies http://www.sun.com/software/jini/.

[5] HAVi http://www.havi.org/.

[6] Diego López de Ipiña, Juan Ignacio Vázquez, Daniel García, Javier Fernández, Iván García, David Sainz and Aitor Almeida, EMI2lets: a Reflective Framework for Enabling AmI, Journal of Universal Computer Science (J.UCS), vol. 12, no. 3, pp. 297-314, March 2006.

[7] A. Ferscha. Coordination in pervasive computing environments. IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, pp. 3-9, Jun 2003.

[8] D. Saha and A. Mukherjee. Pervasive computing: a paradigm for the 21st century. Computer Vol. 36(3), pp. 25-31, March 2003.

[9] C.A. da Costa, A. C. Yamin and C.F.R. Geyer. Toward a general software infrastructure for ubiquitous computing. IEEE Pervasive Computing, Vol. 7, No. 1, pp. 64-73, Jan-March 2008.

[10] http://www.comp.lancs.ac.uk/computing/research/mpg/_reflection/remmoc.php.

[11] Carlos A. Flores-Cortés, Gordon S. Blair, and Paul Grace. An Adaptive Middleware to Overcome Service Discovery Heterogeneity in Mobile Ad Hoc Environments. IEEE Distributed Systems Online Volume 8, July 2007, pp. 546–563.

[12] P. Rigole, C. Vandervelpen, K. Luyten, Y. Vandewoude, K. Coninx, and Y. Berbers. A component-based infrastructure for pervasive user interaction. In International Workshop on Software Techniques for Embedded and Pervasive Systems, pages 1–16, May 2005.

[13] A. Uribarren, J. Parra, J.P. Uribe, M. Zamalloa, K. Makibar. Middleware for Distributed Services and Mobile Applications. Proceedings of the first international conference on Integrated internet ad hoc and sensor networks, May 2006.

[14] A. Uribarren, J. Parra, J.P. Uribe, K. Makibar, I. Olalde, and N. Herrasti, "Service Oriented Pervasive Applications Based On Interoperable Middleware", Proceedings of the 1st International Workshop on Requirements and Solutions for Pervasive Software Infrastructures, 2006.

[15] F. Le Mouël, N. Ibrahim, Y. Royon, and S. Frénot, "Semantic Deployment of Services in Pervasive Environments", Proceedings of the 1st International Workshop on Requirements and Solutions for Pervasive Software Infrastructures, 2006.

[16] http://www.w3.org/TR/wsdl

[17] J. Allard, V. Chinta, S. Gundala, and G. Richard III. Jini meets UPnP: An architecture for Jini/UPnP interoperability. In The 2003 International Symposium on Applications and the Internet (SAINT- 2003), Orlando, Florida (USA), January 2003.

[18] A. Sameh and R. El-Kharboutly, "Modeling Jini-UPnP Bridge using Rapide ADL," Proceedings of the IEEE/ACS International Conference on Pervasive Services (ICPS'04), Beirut, Lebanon, July 2004, p. 237.

[19] Lin, Cheng-Liang; Huang, Chi-Chih; Wu, Zheng-Ying; Wang, Pang-Chieh; Hou, Ting-Wei. A Collaboration Proxy for Converging UPnP and Jini Devices Based on OSGi. CCNC 2007. 4th IEEE Volume, Jan. 2007 Page(s):916 - 919

[20] http://www.gumstix.com/products.html.

[21] http://www.xbow.com/

[22] http://www.fagor.com/es/_bin/cast/productos.php.