Developing User-Centric Applications with H-Omega

Clement Escoffier[1], Jonathan Bardin[2], Johann Bourcier[3], and Philippe Lalanda[2]

[1] akquinet AG, Bülowstraße 66, 10783 Berlin, Germany
`clement.escoffier@akquinet.de`
[2] Grenoble University, BP 53, 38041, Grenoble Cedex 9, France
`{jonathan.bardin,philippe.lalanda}@imag.fr`
[3] Imperial College London, 180 Queens Gate, London SW7 2BZ, UK
`jbourcier@doc.ic.ac.uk`

Abstract. The recent proliferation of ever smaller and smarter electronic devices, combined with the introduction of wireless communication and mobile software technologies enables the construction of a large variety of pervasive applications. The inherent complexity of such applications along with their non-expert clientele raises the necessity of building middleware solutions. This paper proposes to use the H-Omega application server to build user-centric applications. H-Omega relies on a service oriented component platform, iPOJO, and provides useful technical services for pervasive applications. This paper presents how to easily develop user-centric applications on the top of the H-Omega platform.

Keywords: pervasive application, application server, web portal, iPOJO.

1 Introduction

Our present living environments are being increasingly populated with ever smarter and smaller electronic devices. The introduction of such communicating devices has already changed the way we interact with social and physical environment [1]. However, this trend grows up, and it seems to be a mere beginning. Devices continue to feature progressively more and more sophisticated functionalities, and become to interact each other for providing new, higher-level services.

Most of the research efforts already done were focusing on providing hardware that can actually enable such interactions. Plenty of devices providing this kind of features are already commercialized. But very few interesting applications take advantages of this new infrastructure. This is mainly due to the complexity of building software that can actually benefit from this underlying hardware. Indeed, usual software engineering techniques and tools are not suitable. Several software engineering challenges remain to be solved before fulfilling the vision of a pervasive world. Among these challenges, we can notice the high degree of dynamics, distribution, and heterogeneity of the devices involved and also major security and privacy concerns.

We have investigated a global architecture for designing pervasive applications especially related to the home environment [2, 3]. The main result of our previous works was the design of a residential application server providing a suitable runtime for pervasive applications. This runtime is based on a service-oriented component platform, a set of common services and a mechanism reifying all available devices as services. This platform considerably simplifies the development of pervasive application as stated in [3, 4]. This paper describes how this infrastructure can be used to develop user-centric applications.

The rest of the paper is organized as follows. First, we present the background of this work, including pervasive computing challenges, and the service-oriented computing paradigm. This is followed by a description of the H-Omega application server and how it resolves the common pervasive application development issues. Examples of user-centric applications will also be presented. Finally this paper will conclude by pointing out major contributions, and ongoing work.

2 Requirements and Background

2.1 Requirements for Successful Pervasive Computing

The success of any pervasive system highly depends on several key elements, including provided functionalities, Quality of Service (QoS) and affordability. In short, pervasive applications must offer services that are useful, or somehow interesting to the user. In addition, provided functionality must be associated with domain-specific QoS guarantees, such as performance, dependability and usability. System performance allows users to experience natural, *real-time* interactions with the pervasive environment. Dependability ensures service reliability and security and implies that the same behavior is experienced every time the system is run in similar execution scenarios. Application usability implies ease of service exploitation, with no expert knowledge required and with minimal maintenance effort necessary for modifying and evolving the system. Finally, the overall utility of provided services must overcome the effort required to acquire and maintain the corresponding pervasive systems. Affordable pervasive solutions imply that clients are willing to invest the required resources in exchange for offered functionalities, both in the short term (e.g. acquisition and installation) and over long durations (e.g. maintenance).

Pervasive computing systems generally consist of various electronic devices and software entities capable of communicating with one another. Different types of software-equipped appliances may be available for a variety of purposes, such as interacting with the real environment, providing control and display services, or exposing data and application interfaces to other devices and applications. The main challenge of the pervasive computing domain is to provide coherent pervasive environments, offering useful applications and services, based on an entanglement of heterogeneous, distributed and dynamic devices and software services, communicating via various technologies and protocols. In this context, several characteristics specific to pervasive equipments make this domain appealing from a business perspective, while raising difficult problems for system development and maintenance. Such device properties include:

- Distribution. Devices are typically scattered across the physical environment and accessible via diverse protocols, generally over a wireless communication support.

- Heterogeneity. A vast range of appliances, software technologies and communication protocols are available in the pervasive computing domain. A consensus on uniform and compatible implementations is not presently foreseen.
- Limited resources. Resource availability is generally scarce on the physical execution platforms employed in pervasive systems. In the pervasive home context, software applications typically run on a small gateway, with little memory space and low processing capabilities.
- Dynamism. Device availability is by far most volatile in pervasive systems with respect to other computing system types. This is due to several facts, including: i) users may freely and frequently change their locations and hence the locations of equipments they carry; ii) users may voluntarily activate and deactivate devices, or devices may unexpectedly run out of battery. This directly impacts on the availability of services running on these devices; iii) users and providers may periodically update deployed software services.

In addition to hardware and software dynamism, pervasive systems are constantly confronted with changes in their execution contexts. This may include modifications in the user's current behavior, social circumstance, location, mood, or general routine, as well as changes in other software applications' availability and behaviors.

Nowadays, there is no frameworks dealing with all those requirements and providing facilities to design, develop, deploy and execute pervasive applications. Such framework must provide a way to create dynamic and secure (safety, privacy…) applications dealing with heterogeneous devices and services.

2.2 Service-Oriented Computing

Service-Oriented Computing (SOC) [5, 6] is a new trend in software engineering that uses services as basic elements for building applications. In this computing paradigm, a service represents a computational entity described by a specification. This specification contains both functional (service interface) and non-functional (QoS) parts, without referring to the service implementation. Consequently, services can be supplied by multiple service providers and feature various implementations. At runtime, available services are made accessible by registering into service brokers where service consumers can dynamically discover them. A service consumer is then able to invoke the service by only relying on the service specification.

iPOJO [7, 8] is a service oriented component runtime that aims to simplify the development of applications on top of OSGi [9] platforms. iPOJO allows the straightforward development of application logic based on Plain Old Java Objects (POJO). iPOJO subsequently injects non-functional facilities into the application components, as necessary. Such facilities cover service provisioning, service dependency and lifecycle management. So components are bound by following the service-oriented interaction pattern. So, they can be developed and evolve separately. In addition to providing a reusable set of non-functional capabilities, iPOJO is seamlessly extensible to include new middleware functionalities. The iPOJO framework merges the advantages of component and service oriented paradigms. Specifically, iPOJO application functionalities are implemented following the component orientation paradigm. Each component is fully encapsulated, self-sufficient and provides server and client interfaces exposing its functionalities and dependencies, respectively.

3 H-Omega: An Application Server for Pervasive Applications

In our previous works, we developed an infrastructure to execute pervasive applications on residential gateways. The resulting infrastructure, named H-Omega, is an application server targeting specifically pervasive applications. This work follows a software engineering trend that aims to build flexible and modular application servers providing suitable runtime environments for applications such as JEE and Spring. The existing application servers do not suit to our particular application domain, which requires specific technical services, dynamism and flexibility. The main part of the H-Omega architecture is a residential gateway that could be seen as a home application server. Residential applications are deployed and run on this gateway. This gateway has the ability to discover and interact with dynamic devices inside the home and also with external services.

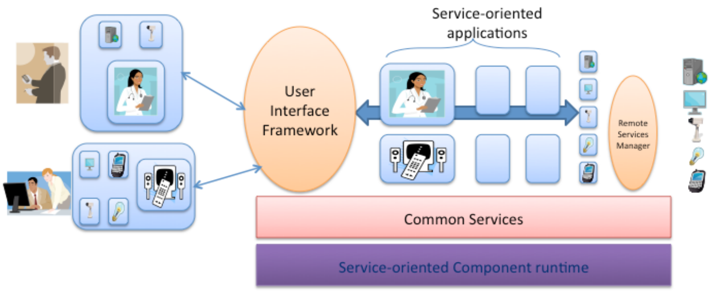
Service-oriented applications
User Interface Framework
Remote Services Manager
Common Services
Service-oriented Component runtime

Fig. 1. H-Omega architecture

The figure 1 shows the internal design of the H-Omega residential server. This computing infrastructure is based on the service-oriented computing paradigm. Home applications are built using the iPOJO service oriented component model. All available devices are reified as services in the framework. The life cycle of these services follows the availability of the device they represent. These services take in charge the communication between applications and the real devices or web services. Application developer can then rely on these services to build their own application without taking care of all the tricky problems of device distribution, heterogeneity and dynamism.

The H-Omega server also provides common services in order to further simplify the development of residential application. These common services are required functionalities across applications. These provided facilities currently include event communication, scheduling of tasks, data persistence and remote administration facilities. To support user-centric applications, H-Omega also provides an infrastructure to monitor the context. This context contains information related to inhabitants (location, medical data), as well as physical metrics (temperature, luminosity…). The context itself is an aggregation of context sources that can be sensors, or applications. Applications can also be notified when the context changes and react to context changes. Applications interested in the context just specify the monitored properties thanks to a LDAP filter, and will be automatically notified when a change occurs changing the LDAP filter evaluation. Each application can also defines its own private context service containing only the domain-specific data (in the domain language). This private service creates a bridge between context sources and the global context service and the application context. Thanks to this context service, applications can directly collect "understandable" data and be notified of meaningful changes.

The H-Omega server[1] constitutes an open infrastructure in which service providers can freely deploy and withdraw applications, taking advantages of the available devices in the home and of the context. The H-Omega framework copes the requirements of pervasive environment and provides a way to design, develop, execute and manage pervasive applications. The H-Omega server was developed and used in the ANSO ITEA project. Moreover, it is still used by France Telecom and Schneider Electric. Akquinet A.G. also investigates using the H-Omega server for mobile applications.

4 The Follow Media Application

To illustrate how to use H-Omega to create user-centric application, this section describes a media-on-demand application selecting the media render according to the user location. Despite this application focus on media, it is not the only purpose of the H-Omega gateway.

This application relies on the UPnP Media profile [10] and the context service provided by the H-Omega server to track user location. Medias are stored in UPnP media servers (Figure 2), or in any media repository reified as a media server service in the gateway (for example a video-on-demand Internet service). As soon as a new media server service is available in the house, contained media will be added to the application. The user can list available music and movies. When the user begins to listen or watch a media, the application selects the media renderer that is in the same room as the user, and plays the media to this renderer. If the user moves to another room, the media is stopped. As soon as the user is back in a room with another UPnP media renderer, the application continues to play the media on this renderer.

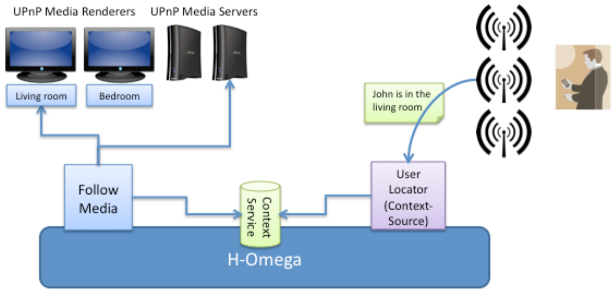

Fig. 2. Follow Media Application

[1] H-Omega is an open-source project available at http://ligforge.imag.fr/projects/homega/

Thanks to the H-Omega application server, the application knows at anytime where is located the user and which media server and renderer are available. One of the main advantages of this application is that it can interact with heterogeneous repositories. UPnP Media service are obviously supported but the application maps them as "raw" media servers. So, any other repository exposed as a "raw" media server can be used by the application.

5 Conclusion

Despite the democratization of smart objects, developing pervasive applications is far from easy. The H-Omega application server provides a suitable runtime to develop pervasive application for home context applications. Moreover, H-Omega provides abilities for applications to track context changes. This paper has demonstrated how H-Omega can be used to create user-centric applications. Two examples were presented.

We are currently investigating several perspectives. First, we are building an autonomic toolkit on the top of H-Omega to create applications with a higher-degree of autonomy. Moreover, we are considering the generation of domain-specific IDE to help the design, development, and management of home-context applications.

References

1. Weiser, M.: The computer for the 21st century. Scientific American 265(3) (1991)
2. Bourcier, J., Escoffier, C., Lalanda, P.: Implementing home-control applications on service platform. In: 4th IEEE Consumer Communications and Networking Conference (CCNC 2007), Las Vegas (January 2007)
3. Escoffier, C., Bourcier, J., Lalanda, P.: Toward an Application Server for Home Applications. In: 5th IEEE Consumer Communications and Networking Conference (CCNC 2008), Las Vegas (2008)
4. Bottaro, A., Bourcier, J., Escoffier, C., Lalanda, P.: Context-Aware Service Composition in a Home Control Gateway. In: IEEE International Conference on Pervasive Services (2007)
5. Papazoglou, M.P., Georgakopoulos, D.: Service-oriented computing. Commun. ACM 46(10), 24–28 (2003)
6. Huhns, M.N., Singh, M.P.: Service-Oriented Computing: Key Concepts and Principles. IEEE Internet Computing 9, 75–81 (2005)
7. Escoffier, C., Hall, R.S., Lalanda, P.: iPOJO An extensible service-oriented component framework. In: IEEE International Conference on Service Computing. Salt Lake City (2007)
8. Escoffier, C., Hall, R.S.: Dynamically adaptable applications with iPOJO service components. In: Lumpe, M., Vanderperren, W. (eds.) SC 2007. LNCS, vol. 4829, pp. 113–128. Springer, Heidelberg (2007)
9. OSGi Alliance. OSGi Service Platform Core Specification Release 4 (August 2005), `http://www.osgi.org`
10. The UPnP Forum, `http://www.upnp.org`