

Service Oriented Pervasive Applications Based On Interoperable Middleware

Aitor Uribarren, Jorge Parra, J.P.Urbe, Kepa Makibar, Ione Olalde, Nati Herrasti

Software Technologies Area, Ikerlan

{auribarren, jparra, jpuribe, kmakibar, ione.olalde, nherrasti}@ikerlan.es

Abstract. The widespread deployment of inexpensive communications technology, computational resources in the networking infrastructure, and network-enabled end devices poses an interesting problem for end users: how to locate and interact with a particular network service or device. The increasing presence of mobile and ubiquitous applications has created a need for distributed systems and applications that are dynamic and can efficiently adapt to changes in service availability, user requirements and protocols. This paper describes the development of pervasive applications based on a middleware proposal which extracts the functional description of available services in heterogeneous networks exposing it in a unified syntax, and offering them using some standardized protocols.

1 Introduction

Service discovery and interaction are composed of the ability to describe, disseminate, select, and use a service [1]. As technological services become present in more and more devices and the number of services rises, the density of services will be higher and higher[2]. In a networked environment, like a networked home environment which is being researched by IST Amigo [3] project, services from diverse domains are available. Therefore, a system that allows the integration of such a heterogeneous set of services is absolutely necessary. Further, in order to facilitate the development of applications that use those services and to reduce the complexity of their administration, a number of service technologies have been proposed: Jini [4], UPnP [5], SLP [6] and so on. However, this diversity means different service spaces, each one supporting particular protocols. That's the reason for having bridges amongst different service spaces. A general discussion about these problems and some possible solutions can be found in [7], [8]. However, this kind of solutions require dedicated ad-hoc bridges amongst specific service technologies. This paper presents a pervasive middleware architecture that aims at presenting the physical-hardware services as software services following standard service technologies. It's based on extracting the functional description of the available services; representing them using a unified service specification; and exposing them using standard service technologies. Thus, applications can consume and combine services from heterogeneous devices that use

diverse technologies. However, ad-hoc bridges amongst specific service technologies are not required anymore.

The rest of the paper is structured as follows: Section 2 describes the motivation of the proposed work. Section 3 gives an overview of the design principles applied. Section 4 describes the Unified Service Model specification. In section 5, the system model is presented. Section 6 details a prototype implementation of target applications. Finally, Section 7 concludes the paper with directions for future work.

2 Motivation

Pervasive systems have been researched from various perspectives: [9] and [10], and there are two aspects to support depending on the components involved. One aspect is the access to any available service provided by any device at anytime and anywhere. The other aspect is the capability of developing applications that consume the services provided by the previously mentioned devices in a standard way, without taking into account the base technologies.

In consequence, two basic requirements can be extracted from this study: hardware resource availability and software service availability. Pervasive applications should be able to consume services from heterogeneous devices, using diverse service technologies. But, obviously, ad-hoc solutions should be avoided: the system should be scalable in terms of supported base technologies (incorporating new device families) and in terms of service spaces (incorporating new service technologies).

3 Design principles

3.1 Middleware Design Principles

The middleware design is based on the following principles: first, be capable of getting available services from the heterogeneous environment; second, extract the service information in terms of syntax description (methods, properties and events) and meta-information (service types, names, identifiers, providers...); and third, offer the available services using diverse service technologies (UPnP, Web Services, etc.).

Figure 1 depicts an overview of the middleware model. Service oriented applications can discover and consume services provided by heterogeneous devices present in the network, regardless of the base technologies (network, protocol, platform, etc.) using standardized service technologies.

Starting from the bottom, the Drivers layer is responsible for giving access to a device, extracting the functional description of the available services, and generating a full description of the each service using a unified syntax.

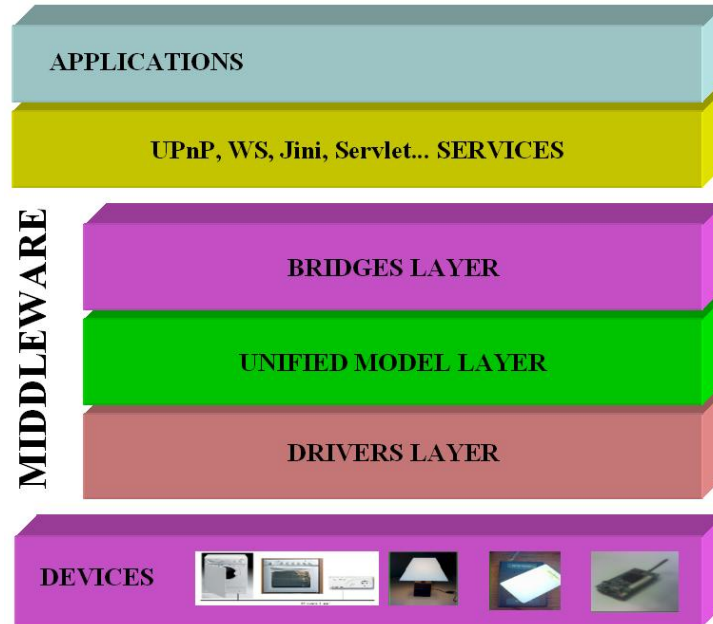


Fig. 1. Middleware model overview

Every service can be characterized by the following elements:

- Methods (name and signature)
- Properties (name and type)
- Events (name and signature)
- Meta-information (service name, service type, service identifier, etc.)

Therefore, the unified description consists of the required data to describe and consume a service.

The Bridges layer is composed of a set of components that act as gateways to standardized service spaces. A bridge, using the unified description of a service, is responsible for procuring an interface for the corresponding service space. For example, a bridge to UPnP should instantiate a UPnP device that offers a service based on the unified description. A bridge to Web Services, should also instantiate a Web Service based on the same unified description.

Any bridge in this layer is absolutely and completely independent of the base technologies. As a consequence of using a service abstraction, supplied by the unified description, the scalability and extensibility of the system is guaranteed. New device technologies can be incorporated, just adding new drivers. In a similar way, new service technologies could be integrated by means of new bridges. Further, it must also be emphasized that the proposed extensions don't influence the current environment at all.

3.2 Application Design Principles

Application design is based on the following principles: first, an application is a client of a set of services using specific service and interaction protocols; second, application developer chooses the service technology to be used; and third, applications use the available services, taking into account their dynamism. A service might leave the network and the application should be able to search for a similar one, if any, and dynamically bind to the new service.

4 Unified Service Model

One of the main objectives of the middleware proposal is to enable developers to select a service technology. This means that the native services must be mapped to upper-level ones. As previously mentioned, this is achieved in two steps and the Unified Service Model defines the intermediate level between them. This section is a deeper description of the Unified Service Model. Figure 2 depicts the class diagram for this model.

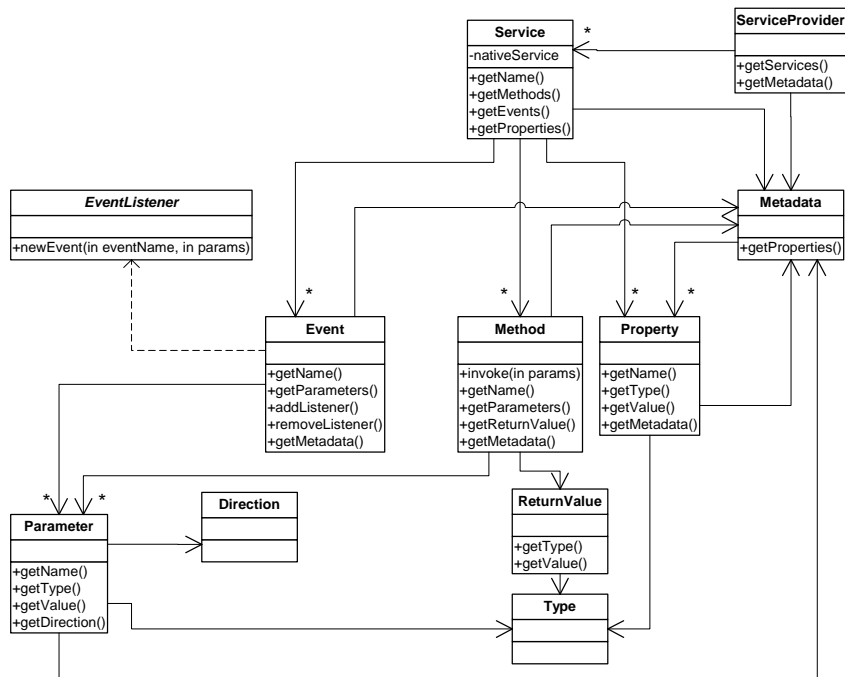


Fig. 2. Service Model class diagram

A service can be described in terms of a set of methods, properties, and events. Therefore, a service can be functionally modeled like a collection of methods,

properties and events. Thus, it can support any service described according to the object oriented programming paradigm. The non syntactic properties of some of the above mentioned elements (service provider, location, manufacturer...) can be also collected and modeled. This model is intended to be generic enough to gather all the information that can be extracted from the native service.

5 System Model

Figure 7 depicts the overall system model. It's composed of a heterogeneous set of devices, the above described middleware proposal and a set of service-oriented applications that consume the services offered by the middleware.

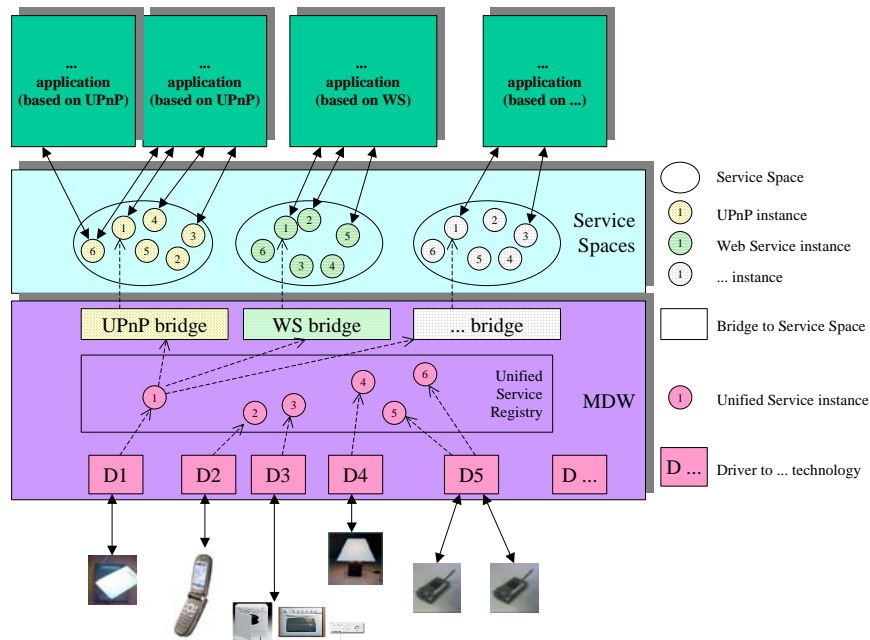


Fig. 3. System model

Applications are clients of a specific service technology. A UPnP based application will search for UPnP devices in the network to be able to use the services they provide. In the same way, a Web Service based application will consume the available Web Services. Both applications can access the same service, but using different discovery and interaction protocols. This is achieved by the instantiation of service instances in the corresponding service space.

D1 is a driver to a technology T1, which gives access to a service offered by a T1 based device. It can extract all the information about the service. Therefore, a Unified Service object, with the whole information of the base service, can be instantiated. These objects are registered in the Unified Service Registry.

Bridges are not base technology dependent. They just consume Unified Service objects from the Unified Service Registry in order to proportionate instances of the services in the corresponding service space. Thus, a UPnP bridge will proportionate a UPnP device to access the service functionality, while a Web Service bridge will procure a Web Service to invoke the service functionality. Several bridges can coexist, creating their own Service Space.

6 Target Applications

As mentioned above, the system goal is to present the physical-hardware services as unified-standardized-generic software services to the high level applications. This section will describe two simple pervasive applications to demonstrate the middleware capabilities. They involve off-the-shelf wireless sensors (to measure light level and falls), RFID locators (to get user's location), home automation actuators (lamps and buzzers) and cell phones.

Applications combine services from devices that use different protocols (currently not interoperable), so they are suitable to show how the middleware makes this heterogeneity disappear from the application developer's point of view.

6.1 Ambient light adjuster

Based on current light level, and on user location, the system accommodates the ambient light. The lamps located in the user's workplace are switched on whenever the user is there and the light level is not high enough; otherwise, the lamps are off.

RFID based locators, TinyOS operating system enabled Crossbow's MicaZ sensors for light level monitoring and two different automated lamps (a standalone RS232 controlled lamp and a BDF[12] home automation bus based lamp) have been used to carry out the prototype. RFID locators provide a discrete location of the users, i.e. whether the user is in a specific location or not. MicaZ modules inform about the current light level in their location. Both lamps can be remotely switched on/off according to their specific protocol.

The application is based on UPnP. This means that it just requires a UPnP control point (UPnP client) to discover and to interact with any UPnP device in the network. Therefore, if the middleware incorporates a UPnP bridge, all the available services will be also accessible via UPnP. Application doesn't need to take care of the nature and base technology of the required services: it doesn't need to know if the light sensor is connected to a serial port, if it uses some specific protocol to communicate... From the application's point of view all those peculiarities of the devices are avoided, and all the devices are used as standard service providers.

The UPnP devices are instantiated by the UPnP Bridge. In this way, application developers just need to search for UPnP locators, light sensors and lamps. All the

available services will reply and, from that moment on, the application will be able to invoke the actions and subscribe to their events.

If Web Services had been chosen as service technology in the application, the existence of a bridge to Web Services would be a requirement. Obviously, both of them can simultaneously coexist, allowing the development of applications that can use the same service using diverse protocols.

6.2 Fall detection application

The intention of this application is the following: if the system detects a fall, the incidence must be notified. Depending on the location the system will procure different notifications. If somebody is close enough, a buzzer will inform about fall. Otherwise a SMS will be sent.

RFID based locators, MicaZ accelerometers for fall detection, a buzzer and a SMS sender are involved in this prototype. Again, RFID locators provide a discrete location of the users; wearable MicaZ modules inform about a probable fall; the buzzer rings on demand; and a SMS sender application is used to send a message to any cell phone. In this case, the application follow exactly the previously detailed schema. It doesn't need to know the nature of these devices, but to be able to discover and use them according to a unique service technology.

7 Conclusion and future work

This paper has presented the high level design of a programming middleware and supporting infrastructure for developing pervasive applications and services. The authors evaluate the possibility of any application to receive, by means of this middleware, in a comprehensive and structured form all relevant information. Studying this particular form of service awareness provides valuable insight that helps with formulating systems that handle heterogeneous services provided by devices of diverse nature.

The work related to the topic of this paper covers a number of different areas: service interoperability, service models, service discovery and mechanisms for application and network adaptation. This section also describes a non-exhaustive selection of related work from these areas. Many existing approaches focus on how mobile applications can adapt to changing resources [13]. Although the work presented in [14] proposes a mean to supply information on communication resources so that the applications' interface is a fixed one, WSDL, complete application development abstraction can only be achieved if discovery and interaction are not limited to a specific definition language, instead of forcing application adaptation. Other strategies for application adaptation like [15] are important because they provide hints on what kind of information is relevant, but they don't focus on device heterogeneity and multiple ways of accessing those devices. [16] elaborates on interoperability methods for service discovery, but doesn't address the service interaction issue. The common set of events for service discovery interoperability is a minimal subset of the full capabilities of the set of service technologies involved.

There are still many challenges to overcome. Integrating other service discovery protocols like Jini, SLP, etc. is one of them. So, bridges from Unified Service Model to these protocols should be researched further. The refinement of the Unified Service Model specification must also be addressed in our future work.

Acknowledgements

The work described in this paper has been supported by the European Commission under the IST program, as part of IST Amigo project.

References

- [1] A. Friday, N. Davies, N. Wallbank, E. Catterall, and S. Pink, "Supporting service discovery, querying and interaction in ubiquitous computing environments," *Wireless Networks.*, vol. 10, no. 6, pp. 631–641, 2004.
- [2] M. Weiser, "The computer of the 21st century," *Scientific American*, vol. 265, no. 3, pp. 66–75, Sept. 1991.
- [3] IST Amigo project. <http://www.hitech-projects.com/euprojects/amigo/index.htm>
- [4] Sun. Technical White Paper: Jini Architectural Overview. 1999.
- [5] UPnP Forum. Universal Plug and Play Device Architecture, <http://www.upnp.org/>.
- [6] SLP. Service Location Protocol. <http://rfc.net/rfc2165.html>
- [7] 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.
- [8] 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.
- [9] *UbiComp 2003: Ubiquitous Computing: 5th International Conference*. Springer-Verlag Heidelberg, 2003.
- [10] *Annual Conference on Pervasive Computing and Communications(PerCom 2004)*. IEEE Press, 2004.
- [11] Open Services Gateway Initiatives. OSGi Service Gateway Specification Release 3.0, March 2003. <http://www.osgi.org/>.
- [12] Fagor, January 2006. <http://www.fagor.com/es/productos/index.html>
- [13] G. Coulson, G. S. Blair, M. Clarke, and N. Parlavantzas. The design of a configurable and reconfigurable middleware platform. *Distributed Computing*, 15(2):109--126, 2002
- [14] Paul Grace, Gordon S. Blair and Sam Samuel. "ReMMoC: A Reflective Middleware to Support Mobile Client Interoperability". In *Proceedings of International Symposium on Distributed Objects and Applications(DOA)*, Catania, Sicily, Italy, November 2003
- [15] N. Limam, J. Ziembicki, R. Ahmed, Y. Iraqi, T. Li, R. Boutaba and F. Cuervo, OSDA: Open Service Discovery Architecture for Efficient Cross-domain Service Provisioning, *Computer Communications Journal*, Special Issue on Emerging Middleware for Next Generation Networks. 2005.
- [16] Y.-D. Bromberg and V. Issarny. INDISS: Interoperable Discovery System for Networked Services. *Middleware 2005*.