Smart Home Systems

Chapter · February 2010

DOI: 10.5772/8415 · Source: InTech

4 authors, including:

Johann Bourcier
Université de Rennes 1
41 PUBLICATIONS **303** CITATIONS

1

Smart Home Systems

P. LALANDA, J. BOURCIER, J. BARDIN & S. CHOLLET
Grenoble University
FRANCE

1. Introduction

The pervasive computing area has recently gained major importance from both industry and academia and is changing the way we interact with our environment [1]. This computing domain emphasizes the use of small, intelligent and communicating daily life objects to interact with the computing infrastructure. These devices tend to blend in their environment. This is especially true in our homes where new electronic devices such as photo frames aim to be as decorative as powerful. Devices are then not always perceivable by human beings. These new equipments have the ability to communicate with each other, to configure or repair themselves, and perform context-based cognitive and physical actions. The vision of coordinated or cooperating devices teaming up transparently to provide human beings with services of all sorts is actually getting closer and closer.

However, the main part of research efforts has focused so far on providing hardware that can actually enable such interactions. Consequently, plenty of devices providing this kind of features are already commercialized, whereas very few interesting applications take advantages of this new infrastructure.

Indeed, the complexity of building software that can actually benefit from this underlying hardware is often underestimated. Usual software engineering techniques and tools are not suitable. Several software engineering challenges remain to be solved before fulfilling the vision of a true pervasive world. Notably the high degree of dynamism, distribution, heterogeneity and autonomy of the devices involved raises important problems. Major security and privacy concerns have also to be considered while building such systems. Indeed, the Home Network emphasizes the envisioned environment openness to networked entities. It is open to dynamic connections: devices may enter and leave the network spontaneously, providing context-dependent features (e.g. according to user's activity). It is also open to heterogeneous devices: protocols and device types differ according to application domains and service providers. Moreover, devices are spread over the home space, which is not clearly delimited for wireless communicating devices.

In this chapter we describe our work dealing with the provision of a natural execution environment simplifying the construction of pervasive applications. More specifically, we describe our vision of smart home environments, and propose an infrastructure to support the execution of home applications providing end user services using different features provided on the home network.

The rest of this chapter is composed of a background part on the pervasive computing domain and its requirements. This is followed by our vision of the home environment, and how devices, networks and applications should be organized. Then, we describe our work on proposing a home gateway to support the home applications execution. This chapter is then concluded by describing the lessons learned from our experience on providing high level services in the home environment and the perspectives of this work.

2. Pervasive Computing

The pervasive or ubiquitous computing domain corresponds to a model of computing where the user interacts naturally with his environment. The proposed model consists of using the objects in the environment as a way of interaction between the computing system and the user. It was introduced for the first time in 1991 by Mark Weiser in articles [1, 2] presenting his vision of the 21st century computing.

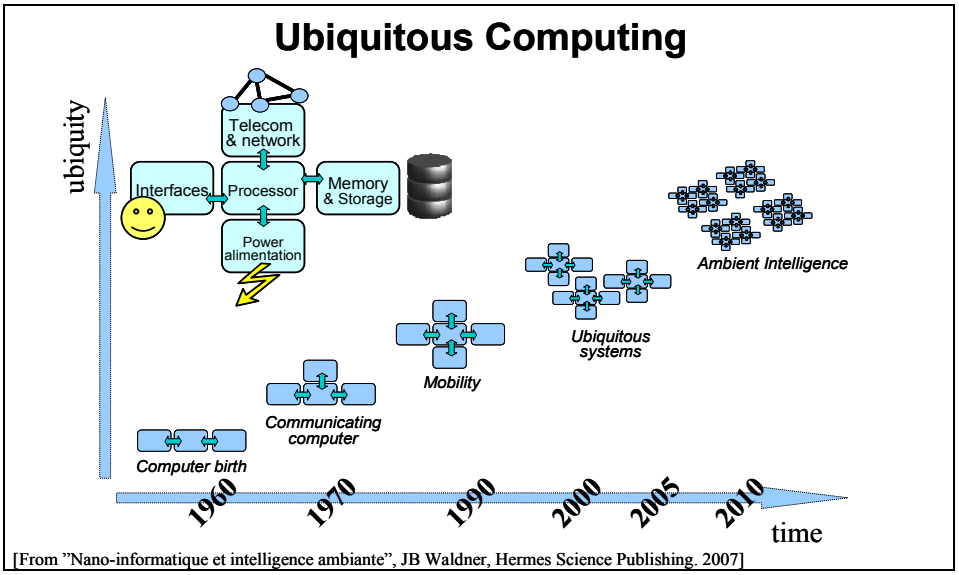

Fig. 1. Computer evolutions since 1960

This new trend of computing is a natural corollary to the evolution of communicating devices. Fig. 1 represents the evolution of computing devices from the 1960's to our days highlighting the main characteristics that lead to the development of pervasive computing. Devices involved in pervasive computing systems must have the following characteristics:

- **Miniature**. As devices must be able to naturally blend or disappear in the environment, the miniaturization of devices is necessary.
- **Communication**. The devices must be able to communicate and to interact with other equipments that are present in the environment.
- **Autonomy**. The devices must have their own source of energy in order to be autonomous.

The recent evolution of computing equipment has contributed to the creation of devices which are coherent with the vision of a pervasive world. Since the invention of computing and the first computer, vendors are engaged in a race for miniaturization and processing power. While in the 1950's there was one machine for multiple users, we currently have multiple machines for one user.

Another important point is the emergence and popularization of communication technologies between computing equipments. The advent of the Internet in the 1990's corresponds to a key point, forming a network of devices from all over the world, using low cost and widespread technologies for communication. In more recent years, new communication technologies have been developed allowing wireless communication (e.g. Wifi, GSM, Bluetooth and RFID). These technologies are now embedded in devices allowing them to communicate with each other thus augmenting their mobility.

The progress achieved in energy autonomy plays an important role in the emergence of this computing domain. Some computing devices are now capable of functioning for years (e.g. RFID sensor) using a simple battery. Device autonomy represents a key factor. If this autonomy proved insufficient, the domain of pervasive computing would lose its interest because its main actors (i.e. the mobile devices) would not have the required features.

In the course of a day, a user is successively immersed in different pervasive environments and has the possibility of taking advantage of each one of these environments.

The pervasive environment in a vehicle is particularly interesting. An example scenario would consist in automatically regulating the destinations in the vehicle's GPS based on the information extracted from the schedule of the day stored in the PDA. The favorite songs of the driver can be downloaded from the MP3 player and played in the vehicle's stereo. Either at the expected time/mileage when the car needs a check up or due to an upcoming trip, the vehicle can inform the user and propose dates for an appointment in the closest auto repairer, taking into accounts the time constraints of the vehicle owner and the auto repairer schedule and availability.

In the work environment several scenarios are envisioned. Devices used in these environments differ according to the nature of the work. The devices may be handheld telephones, PDAs, printers, copiers, desktops, server computers, video projectors, or even coffee machines.

The environment of a restaurant or a bar can also offer interesting pervasive services. For example, the customer may get an interactive restaurant menu where photos, information and pictures of the dishes can be found. It is also possible to directly place an order by using the PDA which would then automatically calculate the bill.

These different environments have the objective of assisting the user in their daily life. A pervasive environment frees users from certain constraints of their daily lives by offering different services using objects from the day to day environment. The services correspond to the applications that execute on top of the host infrastructure of the pervasive environment. These services could be extremely simple, such as an electronic agenda, which interacts with single equipment, or more complex, such as a system which enables energy savings in a home, requiring the interaction with multiple devices.

3. The Home Environment

The home environment corresponds to a subset of pervasive applications that deals with the automation of home devices control. To that end, electronic devices present in the environment have the ability to communicate. The communication protocols used differ according to the type of equipment. An automated home typically allows the control of room luminosity, opening and closing of shutters, heating and air conditioning, or multimedia systems.
Home applications main objective is the comfort and simplification of the daily life of residents, and home support of elderly or convalescents. Several application areas are covered ranging from applications for the supervision of convalescents at home, to home theatre applications and energy consumption control.

3.1 Equipments

The equipment typically involved in pervasive homes include shutters, lighting devices, appliances such as coffee machines, refrigerators, or washing machines, televisions, or multimedia servers, which can all be controlled remotely. Micro-informatics equipment, such as computers, PDAs, monitors, are also part of the home automation sphere. The communication equipment is also essential in this type of system: Internet access points, routers and mobile phones allow access to information technology that may be outside of the home. Also, controllable electrical equipment such as remote electrical plugs, are a centerpiece of these new environments.
These devices must be able to be remotely controlled by the applications that coordinate their actions. Therefore, devices must provide protocols for allowing such type of communication. Nowadays there are more than fifty communication protocols, workgroups and standardizations of protocols for home communication. Among the most popular we can find X10, KNX, EIB, INSTEON, Zigbee, Bluetooth, UPnP and DPWS. These protocols allow the communication between home devices by using different transmission mediums: dedicated communication cables, communication over power lines or transmissions by radio frequency. Such protocols do not provide the same functionalities and hence are not used by the same types of devices. X10, KNX, EIB, INSTEON and Zigbee are dedicated to small devices like light dimmers and shutters. Their communication capacity is extremely limited, but their energy consumption is also limited. However, these protocols only handle communication and are not able to provide device discovery.
UPnP [3], Bluetooth and DPWS [4] are higher level protocols which handle not only communication but also device discovery. Bluetooth is a standard originally conceived for allowing wireless communication between computers and their peripherals. The goal of UPnP and DPWS is to allow peripheral devices to easily connect to each other and to simplify the implementation of home networks (e.g. file sharing, communication and entertainment) and enterprises. UPnP allows such capabilities by defining its communication and discovery protocols on top of existing Internet communication standards such as HTTP, while DPWS basically use the standard defined by Web Services.

3.2 Applications

This section presents three applications that are representative of the field of home computing. The first example is an application keeping convalescents or elderly at home.

This application requires the use of sensors specific to this field such as cardiac monitors, blood pressure gauges, sensors and video surveillance cameras. These devices collect data about the patient, and send it to an application that handles it. If a problem appears, an alarm is automatically triggered to call for help. In normal operation conditions, reports are built by the application and sent regularly to the hospital or to the doctors.
A second example is an application that manages a multimedia entertainment system from the house. The application offers the residents a set of movies from the movie library, and also offers the rental or purchase of movies from Internet service providers. The film is then automatically projected on the screen of the room in which the user is located and the atmosphere associated with the action of watching a video is applied: closing the shutters, and dimming the lights. The ring of the phone is automatically cut off. Instead, the user will have a discreet warning on their display screen on any incoming calls. He may decide to ignore it or to take the calls and the film will be automatically paused.
The last example of application consists of managing the security and minimizing energy consumption of the house in the absence of the user. When the user leaves his home, the lights are turned off, the temperature of rooms is automatically lowered, the alarm system connected and the shutters closed. In case of prolonged absence, a simulation of presence is triggered and regularly opens the flaps and lighting lamps according to the habits of the user. When the user returns home, the system is unplugged, the alarm is stopped; lighting and flaps operate according to the brightness outside.

Based on our experience, we have classified these applications into three categories based on different life cycles of these applications (see Fig. 2).

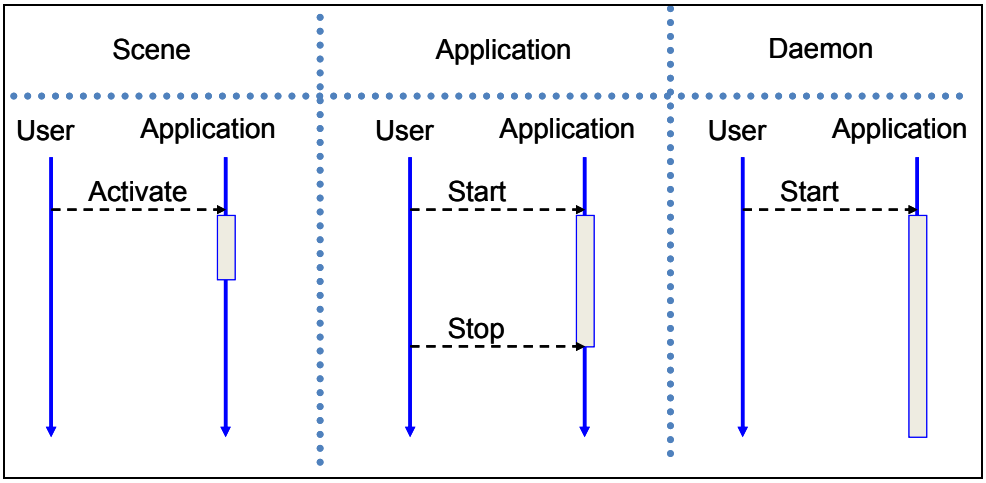

Fig. 2. The three types of home applications

A particular type of application is called scene. These applications consist of running a series of preconfigured actions at the time they are triggered. This type of application executes all of these actions and ends. For example, when the user wakes up, the application will open all shutters of the house, raise the temperature of the bathroom, heat the coffee machine and turn on the television on the favorite news channel of the user.

Another type of application, which will be called instanciable or simply application, runs for a delimited period in time. These applications have the particularity of being started and then stopped. For example, an application is activated at the time the user leaves the house. This application consists of managing the security while minimizing the energy consumed by the house. The application will turn off all the lights in the house, close all shutters, lower the heating, connect the alarm system. When the user comes back home, the application is stopped and disconnected, the alarm system is turned off and according to the circumstances, it may reopen the shutters or turn on the lights.
The last type of application is called daemon. This type of application is started and then runs continuously. For example, an application for the surveillance of patients in their home will collect medical data about the patient and then send daily reports to the doctor. Such applications are executed and never stopped.

3.3 Challenges

This domain of computing encompasses a large number of applications particularly useful to help people in their daily life. The main challenge of pervasive computing is to provide a coherent pervasive environment, providing useful services and applications, involving a set of heterogeneous, distributed and dynamic equipments and software, communicating across different protocols. In this context, several characteristics specific to the field of pervasive computing makes this area attractive in terms of industry and users, while raising difficult scientific problems for the development and management of these systems.
Distribution. The devices are an integral part of the environment. They are scattered in the physical environment and are accessible through different protocols that can use cable or wireless technologies. Applications using the capabilities of such equipment do not necessarily run on the considered equipments and are therefore distributed.
Heterogeneity. There are currently a large number of software technologies and communication protocols for the field of pervasive computing. Today there are no plans on how to reach a consensus on a common and uniform communication protocol. More than fifty protocols, working groups and specification effort are already available for home networks. The standardization of communication protocols is not possible because the devices can be of very different nature, having an impact on the communication protocols used. For example, a lamp communicates through a very simple protocol, while a PDA or a media server can use more complex communication protocols, for example considering security. In addition, manufacturers supplying equipment and protocols have no strategic interest in this type of uniform protocol, since they would lose control of their equipment.
Dynamism. The availability of equipment in a pervasive environment is much more volatile than in other areas of computing. This problem is caused by several factors including: 1) users move freely and frequently changing their location having an impact on the position of equipment they carry; 2) users can voluntarily turn on and off the devices or they may inadvertently run out of battery; 3) users and providers may periodically update the deployed services.
Multiple provider. The devices in a pervasive environment generally come from different vendors. In addition, applications deployed and running on such equipment can be delivered by other suppliers. In this context, some applications will be established through collaboration between different providers involving the creation of applications with several administration authorities. It is envisaged that the equipment vendors and service providers

want to keep some control over their devices and software and thus limit the access to external entities.

Scalability. The number of devices present in a pervasive environment can be very important. This creates a problem of scalability in applications running in this type of environment. Thus pervasive systems must be capable of handling a large number of equipment that is also dynamic.

Security. Security is a key role in building pervasive environments. Indeed, such open systems allow people in the environment to have access to the computing system. However, access to certain devices or personal data must be highly secured. The applications running on this type of system should guarantee the confidentiality and integrity of data. Access to private pervasive systems such as automated homes or cars must also include access control to ensure, for example, that a thief will not be able to disconnect the alarm system by connecting the pervasive system.

Auto-adaptation. In addition to the dynamism of software and equipments, pervasive systems are constantly faced with the evolution in their execution context. These evolutions may include changes in behavior, location, mood and habits of users, as well as changes in behavior or availability of other software. The applications running on this type of environment must be able to adapt to these changes and develop strategies to address the various events that may occur during their execution.

Simplicity of use. Finally, an essential characteristic of a pervasive system is the simplicity of use and management. Indeed, this type of system is intended to be used by users who have no knowledge in informatics. As a consequence, pervasive environments must be accessible to any human being, and even transparent. The purpose of pervasive computing is to make the devices disappear from the environment. This means that the access interface to pervasive systems must be easy to use and the applications running in these systems must be capable of adapting to different events that can intervene to maintain services usable in all circumstances.

4. Architecture of the home environment

One of the major challenges for creating an intelligent house is the design of an open infrastructure for implementing home automation applications. Equipment manufacturers and Internet operators have proposed different architectures.

4.1 Current architecture

Most current systems are based on architecture similar to the one shown in Fig. 3, in which a web server is connected to an Internet gateway using the HTTP protocol or other protocols over IP. The objective of this Internet gateway is to bridge the local network connecting the various equipments in the house and Internet. The home automation services are implemented as distributed applications running on the Internet server and in house equipments. Several gateways provided by different actors (telecom operator and electricity suppliers, home automation equipment suppliers) may be present in one house. Although this architecture allows the implementation of pervasive services for home, it also suffers from some limitations. Most treatments and coordination are made on the server side, affecting the scalability and flexibility:

- The server must handle the additional load when multiple gateways are added or when the number of connected devices in a home increases. The amount of information transmitted between the Internet gateway and the server increases proportionally with the number of equipments in the house.
- The server must know each new equipment introduced into a home to allow the dynamic evolution of services. Thus, the life cycle of equipment must be managed manually because the automatic detection of equipments availability is not feasible in a network of this scale.

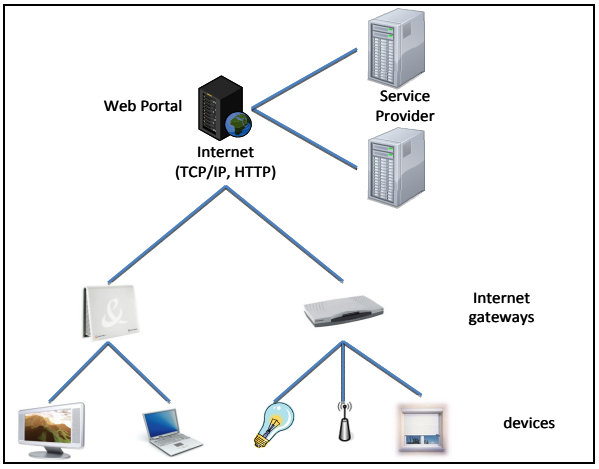

Fig. 3. Usual home computing architecture

4.2 Architecture for a home environment

To overcome the various limitations of commonly used architectures in this domain, we have proposed an innovative architecture [5] for home automation environments (Fig. 4). This work was partially supported by the European ITEA ANSO project. The home environments consist of various equipments from different vendors. We have classified equipments into three categories:

- The electronic equipments available in the house (for example controllable shutters or lamps) provide basic services to sense and act on the environment. Such equipments can be static as lamps or shutters, or may appear and disappear dynamically (such as cellular phones).
- Gateways provide an execution infrastructure for running high-level services or applications aggregating the behavior of basic services provided by the previous equipment.
- Interacting devices (such as televisions, mobile phones, or PDAs) allows users to interact with the system and potentially to manage it. Inhabitants will use them interchangeably to interact with their environment (depending on their habits and their current context).

The proposed architecture is illustrated in Fig. 4. This architecture provides a middleware as a corner stone of the home environment. This middleware provides a substrate for running

residential applications coordinating the behavior of different devices and ensuring a natural interaction, sometimes invisible, with the user. For example, an application running on this middleware can coordinate the behavior of specific equipments such as shutters, air-conditioning or lighting systems.
This middleware provides an execution environment which may be distributed across several gateways. Each gateway is usually materialized in the form of a box embedding a computer with reduced electricity consumption. These boxes generally include a set of physical communication facilities to enable interactions with actual devices. One of these boxes enables Internet access and act as an Internet provider for our middleware.

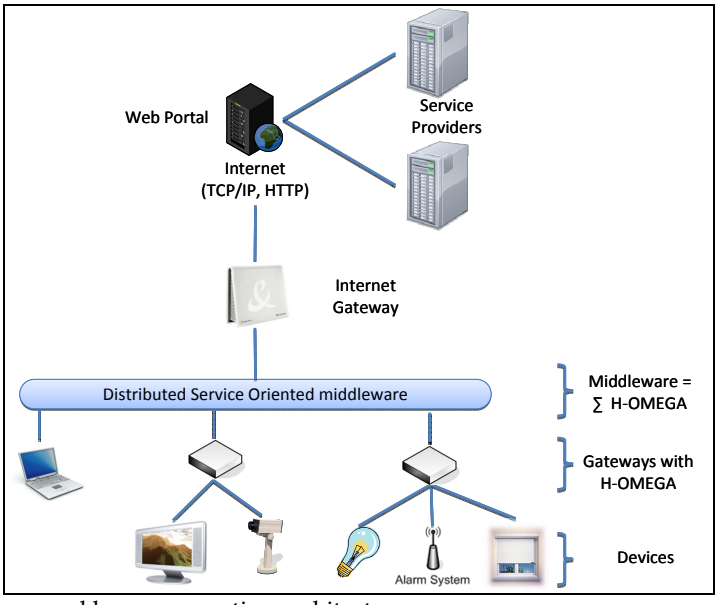

Fig. 4. Our proposed home computing architecture

In the industrial view of such systems, one gateway belongs to an equipments vendor and embeds physical facilities to communicate with these devices. In this vision, gateways present in the home are strictly isolated and do not permit any interaction with their devices or applications. Through this architecture, industrials aim to maintain a total control on their equipments and execution infrastructures. Nonetheless, this architecture does not offer the possibility to build an application coordinating the behavior of equipments from different vendors. As equipment suppliers generally provide one type of equipments, the isolation principle advocates by this architecture quickly becomes a limitation for designing innovative applications. For example, Schneider Electric is specialized on providing controllable lighting systems, and shutters, while Sony is specialized on providing multimedia systems. Thus, it is not possible to implement the multimedia entertainment system proposed in the section 3.2.
Our proposition is to take into consideration the industrial need to keep control on their infrastructure and equipments while allowing the construction of distributed applications involving pieces of software and physical devices from several gateways. Thus, our design

proposes to install on each gateway an application server dedicated to residential computing technology called H-OMEGA [5]. This application server provides an infrastructure for running residential applications. The set of application servers installed on each platform forms our middleware. Each gateway is then able to communicate with each other thanks to our middleware. Thus an application may be distributed across the different gateways present in the house and seamlessly coordinate the behavior of a set of electronic devices connected to different gateways.
H-OMEGA allows a uniform access to in-house equipments connected to the considered gateway. H-OMEGA also provides a way for applications to interact with remote services such as web services. The last feature offered by our applications server is the ability to provide an integrated and portable human interface for controlling the home system. This interface is either available from within the home, or remotely.

Our home environment, including both gateways and equipments, follows principles advocated by the Service-Oriented Computing. Service-Oriented Computing (SOC) [6, 7] is a relatively new trend in software engineering whereby services can be supplied by multiple service providers and feature various implementations. At runtime, a service consumer is able to invoke a service by relying only on a service specification, which specify both functional (service interface) and non-functional (QoS) part, while not referring to the service implementation. An important consequence of this interaction pattern is that SOC technologies support dynamic service discovery and lazy inter-service binding. Such characteristics are essential when building applications with strong adaptability requirements, such as pervasive and residential applications.
We propose to build smart home applications as service-oriented applications. The H-OMEGA application server is based on service-oriented architecture and interactions with remote devices follow the service-oriented pattern. The use of this technology presents several advantages. As previously stated, this technology allows the loose-coupling between different actors which allows the use of other services without having detailed knowledge of their implementation or the interaction protocol used to communicate with their equipment.
We propose to reify each device feature as a service on our middleware to provide an uniform access to electronic devices. This technique addresses the problem of heterogeneity of communication protocols between the various equipments. In addition, the use of the service-oriented components approach provides a natural support for dynamic applications such as the ones found in a house. Indeed, residential applications have to interact with equipments accessible through services, which may be intermittently available. Finally the use of such technology respect the vision of the existing protocols for home: UPnP and DPWS which propose a service-oriented approach. The integration of devices that do not comply with a service-oriented approach is made through the use of a third party mechanism which makes the link between our service-oriented gateway and different equipments.
The proposed application server also allows service providers to deploy, update and remove services and applications remotely. Suppliers can thus control from their own premises all applications and services deployed in all houses.

5. Our Residential Gateway Proposition

5.1 Architectural view

The architecture of our residential gateway, H-OMEGA is illustrated in Fig. 5. This architecture is based on three basic elements used to simplify the design, implementation, development and administration of residential applications. The main elements of this architecture are:

- An infrastructure for service-oriented execution,
- A remote service manager (including equipments, services offered by other gateways and web services),
- A set of facilities or commonly used services to develop this type of application.

The remote service manager can manage both the available devices in the environment of the gateway, the services offered by other H-OMEGA gateways presents in the home and remote software services from outside the home. The role of this entity is specifically to manage the lifecycle of services acting as proxy for remote services either offered by remote equipments, or remote gateways. These local representatives are able to interact directly with the remote service. They follow the life cycle of their corresponding remote service. The role of the manager is to ensure a coherent behavior of these proxies. Applications on our framework have the possibility to transparently use remote services or features from remote devices through their local representatives.

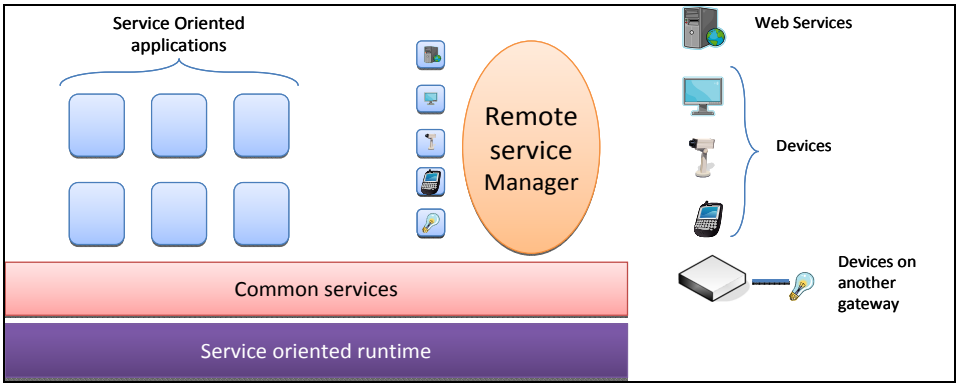

Fig. 5. H-OMEGA application server architecture

The commonly used services in applications are provided by the framework. The applications running on the framework have access to these services. The goal of creating these services is to free developers of applications from this tedious, repetitive and sometimes complex development. The use of these services helps to reduce the bugs in this type of application, because these services are developed once and widely tested. Our framework currently provides:

- A persistence manager to enable applications to store and retrieve persistent data;
- A tasks scheduler for repetitive or delayed tasks;
- An event-based communications infrastructure for enabling asynchronous communications;

- A remote administration module to easily manage deployed residential applications from the vendor premise.

The service-oriented infrastructure allows the design of residential applications with the benefits associated with this type of infrastructure. Applications can be opportunistically bound to services provided on the gateway. The services available to applications through this mechanism include the services provided by other applications, equipments and remote services accessible through local proxies and the common services provided by the platform.

5.2 Implementation

The Fig. 6 shows the stack of technologies used to develop our applications server. Our framework provides a Java-based environment to develop residential applications. It is based on service-oriented technology called OSGi [8] which is a service-oriented architecture featuring management facilities. On top of this technology, we use iPOJO [9]: a service-oriented component runtime that aims to simplify the development of service-oriented applications.

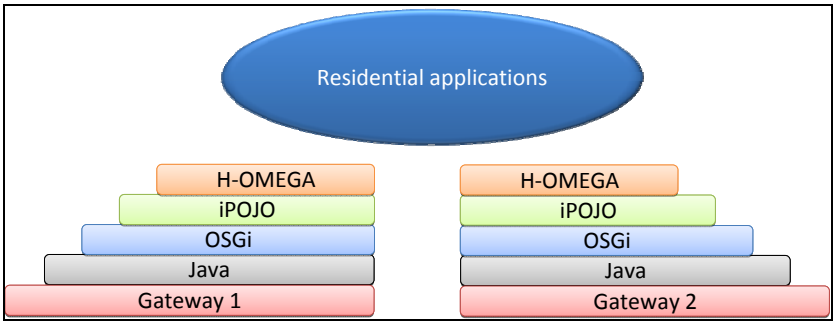

Fig. 6. Stack of technologies used by H-OMEGA

iPOJO is a service-oriented component runtime that aims to simplify the development of applications on top of OSGi SOC 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 include service provisioning, service dependency and lifecycle management. 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 components with 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. As many component-oriented platforms (e.g. Java EE and .NET), iPOJO separates a component's application-specific business logic from its application independent functions. As such, iPOJO components consist of a component implementation that is managed by a reusable container (Fig. 7).

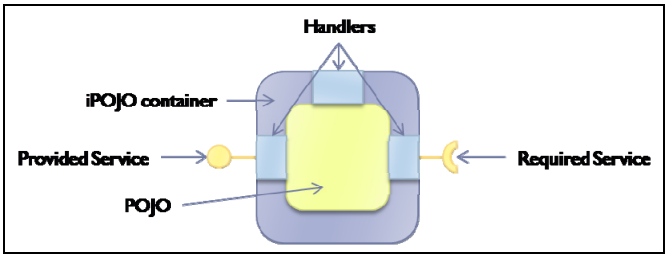

Fig. 7. Internal design of an iPOJO component

iPOJO containers provide common middleware functionalities to the component implementations they manage (e.g. distributed communication and lifecycle management). Each component container can be configured with a different set of middleware services, implemented as "handlers". Once an iPOJO component is deployed, its provided functions are published and made available as services, in conformance with the SOC paradigm. In order for a component's services to become valid, all the component's dependencies must be resolved. For this purpose, available services corresponding to a component's required (or client) interfaces must be found and available.

The use of iPOJO allows us to benefit from all the facilities provided by this technology, particularly the dependencies manager which automatically deals with the dynamic availability of services (specifically services provided by mobile and remote devices). In addition, the extensibility feature of iPOJO enables the specialization of the environment for the residential application needs. We thus have developed handlers to simplify access to commonly used features of our framework:

- A handler to describe the automatic planning of repetitive or delayed actions. This handler (called cron handler) uses the scheduler services provided by our framework.
- A handler to automatically save and restore the state of a service. This handler (called persistency handler) uses the persistence service provided by H-OMEGA.
- A handler to simplify the reception and sending of asynchronous messages. This handler (called Event Admin handler) uses the event-based communication infrastructure provided by OSGi.
- A handler to describe the provisioning of administration features of a service. This handler (called JMX handler) uses the JMX standard provided by the JVM to offers this functionality.

An application developer using H-OMEGA will thus have an easy access to all these features.

6. Examples

This work has been validated as part of the ANSO European ITEA project. The middleware presented in this chapter has been used as a basis of the final demonstrator of the project. ANSO means Autonomic Network for SOHO, where SOHO is used for Small Office Home Office. The objective of this project was to develop an open source platform, intelligent and reliable for different home automation environments to greatly accelerate the development

of new services in this context and to allow their compositions in innovative applications for increase the use of services for the digital home in Europe.
In this context, we have developed several home scenarios to demonstrate the interest of our framework. The applications developed are presented in section 3.2.
The first application is a home hospitalization application to help maintain elderly or convalescents at home. Based on fall detectors and blood pressure sensors, our application constantly monitors the considered person. These data are processed through complex analyzers to detect irregularities or unexpected behaviors. In such cases, an alarm is sent to the closest hospital emergency. In normal operational condition, this application continuously stores information on the patient health and builds reports which are regularly sent to the doctor in charge.

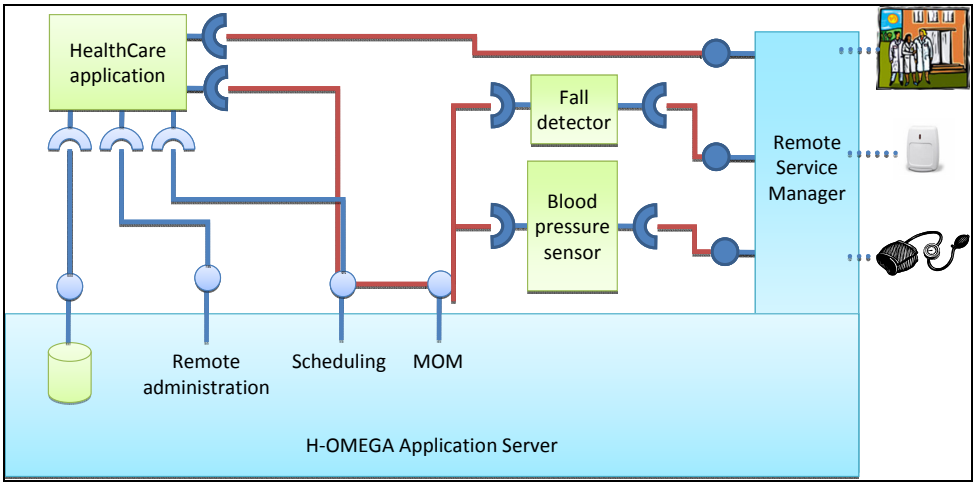

Fig. 8. Home hospitalization application

This application has been designed using the various features of H-OMEGA. As, we do not have access to the real sensors, this application has been built using simulators of the real sensors. To keep the simulation close to the real sensors, we have developed and executed these simulated sensors on a remote computer, and we have used standard protocols such as UPnP to remotely discover and access them. Thanks to our remote service manager, proxies of these sensors are automatically installed on the gateway. All data are transmitted through an event-based communication system to a service in charge of performing anomalies recognition. Data are also stored on a persistent support through the persistency service. The application uses the facility of the automatic planning of repetitive actions to plan the creation of a daily report. Finally, this application uses the remote administration feature to provide a way for the hospital to remotely tune the thresholds of the anomalies detector in order to suit with the patient health evolution.
The second application is a home multimedia entertainment application using standard UPnP media server and renderer devices. In this application, we do not simulate any devices. This application aims at providing a multimedia experience to the user seamlessly integrating several multimedia devices and the shutter and lighting system of the home.

First the user chooses a media to listen or view, and then the system uses the maximum of its capacity to maximize the user comfort. The media follows the user while he is moving throughout the house, and the suitable ambiance for watching media is also set in each visited room. This application mainly benefit from the facilities provided by iPOJO to manage the dependency between the media controller service and the available media renderer in the home. Thus, the application is able to view the media in the room where the user is located. This application is distributed across two gateways: one belonging to the multimedia vendors, the other belonging to the vendor of the shutter and lighting systems. The application mainly runs on the multimedia gateway, but uses the feature of our middleware to access the lighting services on the other gateway.
The third application is an application aiming at minimizing the energy consumption of the house while maximizing the security when inhabitants are away. This application is in charge of running the alarm system, closing shutter and turning off all lights when inhabitants leave the home. If the inhabitants' absence last more than one day, this application launches a service in charge of simulating the presence. This last service makes extensive use of the planning feature offered by our middleware to simulate the inhabitants' usual actions, such as closing shutters, turning off lights in different rooms, etc.

7. Conclusion

Developing correct and maintainable pervasive services is a real challenge today. It is clear that most techniques currently available are not mature, hard to master and, consequently, raise major challenges for the major players of the market.
We believe that two important aspects have to be improved: development environments and runtime environments for pervasive services. In this paper, we have presented recent developments in the area of service-oriented home gateways.
This chapter mainly focuses on the description of our work on a runtime addressing the main limitations of current approach: dealing with a growing number of homes and dealing with heterogeneous mobile devices. The design of our residential application server also respect the industrials main will to keep the control on their own equipments, while encompassing the main limitations of the traditional approach: entirely isolated gateways.
The work described has been implemented on top of an open source project called iPOJO (available as an Apache Felix subproject) and is currently available as an open source project on http://ligforge.imag.fr/projects/homega/. This work has been validated in the ITEA ANSO project and through the creation of several applications validating the usefulness of our framework.
This work, on providing an open infrastructure to enable the development and execution of home applications seamlessly integrating heterogeneous and mobile devices, open several research perspectives. We are currently working on adding autonomic features to home applications in order to reduce the maintenance cost of such applications [10]. This work aims at providing architecture and its corresponding runtime to support the creation of self-configuring, self-optimizing and self-repairing applications.

8. References

[1] Mark Weiser, “The computer for the 21st century”, Scientific American, 265(3):66-75, September 1991.

[2] A. Ferscha, “Pervasive computing and communications”, Beyond The Horizon Thematic Group, IST, 2005 (http://www.cordis.lu/ist/fet/id.htm).

[3] UPnP Plug and Play Forum, "About the UPnP Plug adn Play Forum," in http://www.upnp.org/, 1999.

[4] E. Zeeb, A. Bobek, H. Bonn, and F. Golatowski, "Lessons learned from implementing the Devices Profile for Web Services," in Inaugural IEEE-IES Digital EcoSystems and Technologies Conference (DEST '07) 2007.

[5] C. Escoffier, J. Bourcier, P. Lalanda, J. Yu, “Towards a home application server” 5th IEEE Consumer Communications and Networking Conference (CCNC’08), January 2008.

[6] M. N. Huns and M. P. Singh. Service-Oriented Computing: Key Concepts and Principles. IEEE Internet Computing, vol. 9: pages 75–81, Jan./Feb. 2005.

[7] M. P. Papazoglou and D. Georgakopoulos. Service-oriented computing. Commun. ACM, 46(10):24–28, 2003.

[8] OSGi Alliance. “OSGi Service Platform Core Specification Release 4” http://www.osgi.org, August 2005.

[9] C. Escoffier, R. S. Hall, P. Lalanda, “An Extensible Service-Oriented Component Framework”, IEEE Service Computing Conference, 2007.

[10] P. Lalanda and J. Bourcier, “Towards autonomic residential gateways”, IEEE International Conference on Pervasive Services (ICPS 2006), June 2006.