

A Dynamic-SOA Home Control Gateway

Johann Bourcier, Antonin Chazalet, Mikaël Desertot, Clément Escoffier, Cristina Marin¹

Laboratoire LSR-IMAG, 220 rue de la Chimie

Domaine Universitaire, BP 53

F-38041 Grenoble, Cedex 9, France

{first-name.name}@imag.fr

ABSTRACT

The convergence of smart field devices and business services stands to profoundly change the way we interact with our environment. This is especially true in the home context. In this paper, we present an open architecture and a dynamic service-oriented gateway running home services. The gateway is based on the OSGi standard and provides mechanisms to integrate different service technologies.

Keywords

Home control gateway, Service-oriented computing, Dynamism.

1. Introduction

The connection of business and industrial processes is today of major importance for a wide range of manufacturers. The main challenge is actually to seamlessly integrate application software supporting business activities and industrial devices implementing field applications. Such computing elements have been until recently separated, primarily because of technical issues, including incompatible programming paradigms, network heterogeneity, differing time scales, and the lack of appropriate integration tools. With the Internet's emergence and the proliferation of smart communication devices, stronger coupling between previously autonomous activities is now possible. This trends towards increasingly ubiquitous network-enabled devices is known under the name of pervasive or ubiquitous computing [1][2].

Many believe that linking devices and business services stands to profoundly change the way we interact with our computing environment [3][4]. Networks of devices will assist us in our daily activities using the notions of goals, context and knowledge to autonomously decide on the best actions to be undertaken. In many cases, goals will be

derived from higher level objectives set by the business services. For instance, devices will cooperate to provide a secure and adapted environment to a disabled person. The execution context of a device contains current information about its location, its physical and computing environment, and its human users. Indeed, in order to enable natural and meaningful interactions, devices have to be aware of the humans' activities, desires, whereabouts, needs, etc.

Numerous manufacturers are already using, or planning to use, electronic devices to provide pervasive services to their customers; especially in the home context. Many houses are already covered by wireless and wired network technologies and filled with a plethora of electronic devices allowing the occupants to control their environment (be it for comfort or for entertainment). But several business and technical challenges complicate the manufacturers ability to develop and manage such innovative services. To begin, the business models aren't yet ready; it remains unclear how to commercialize such services and electronic devices, for example, or whether manufacturers should go the marketplace alone or with specialized partners. The second concern is technical. Putting pervasive services in place is a challenging task. It requires to implement the cooperation between heterogeneous, autonomous devices in complex environments in which topologies, communication protocols, security policies, and such are dynamic and differ from one customer to the next.

The purpose of this paper is to present ongoing work in the area of residential gateways carried out in cooperation with Schneider Electric (in the European ANSO project. This work is partially supported by the French Ministry of Industry). Specifically, this work deals with the design of an open computing infrastructure for the development of home pervasive services, including dynamic service-oriented gateways. The paper is organized as it follows. The next section presents a domain analysis and existing computing architectures in the home domain. Section 4 presents

¹ The names of the students entering the SCC Contest are given in alphabetical order. Their advisor is Pr. Philippe Lalanda.

service-oriented architecture benefits for the home automation software systems. Then, section 5 focuses on the notion of smart gateways and the use of the OSGi standard to implement home control gateways. This section also highlights the ways technical issues are solved in our architecture.

2. Problem Background

2.1 Presentation

Home automation is an area that gained more and more interest over the years. Also known as domotic, this subject has to deal with software and hardware concerns that try to improve the comfort, the quality of service and the security of the inhabitants of a house. A common example is a system that assists elderly at home so that they can be autonomous [6]. Another one is a software system managing the whole security of the home.

In order to provide these intelligent and interactive capabilities, smart equipments have to be deployed in the house's habitat – they will form the assistive environment. Equipments can embark sensors for environment monitoring or may offer intelligence likes embedded software services (for instance, a light switcher that can be turned off from distance) [5]. In order to provide the connectivity between in-house devices, wireless or wired network connections are deployed in the home's environment. Equipping home device with network appliances provides device-dependent services [7]. The latter can be accessed through terminals like PDAs, mobile phones or the classical home portal remotely accessible.

One of the main research fields is today the construction of the in-house pervasive environment where intelligent electronic devices have to be deployed in order to provide the user with the desired software functionalities [13][15]. An important issue concerns equipments themselves and their integration in the house's environment. There are some general problems concerning the non-functional part of devices: discovery, configuration and integration within pre-existent frameworks, authentication, authorization, certification, non-repudiation, intrusion, etc. I.e.: the house's infrastructure has to be able to distinguish a coffee machine from a heart-rate monitor. These differences need to be accounted at resource allocation and execution times.

Another important concern deals with autonomic, i.e. with automatic, autonomous and dynamic management of the problems likely to occur inside the pervasive environment. That relates for instance to the management of the network issues or to the dynamic management of the services and software applications executed in the house (the house must be able to reduce or stop the coffee machine management application, if this one consumes excessive resources or if

the consumed resources are different from those allocated to it).

2.2 Home computing architecture requirements

We believe that the vision of pervasive home services is not yet achievable today for several reasons. Specific architectures and techniques are necessary to meet stringent requirements, including:

- **Heterogeneity.** Devices and networks that are currently found in a house are very diverse. Uniform and compatible implementations are not foreseen: today, more than 50 candidate protocols, working groups and standard specifications for home networking already exist, providing communication and interoperation between access and indoor networks.
- **Security.** Security policies (regarding children safety for instance) are different from one customer to the other and are very changing over time. In addition, most home owners will not change their security policies in order to suit a given service.
- **Scalability.** As environmental smartness grows, so will the number of electronic communication devices. The mere number of computing elements will raise challenging software issues regarding the development, installation, and maintenance of scalable in-house services.
- **Flexibility.** In the home context (as in the telecommunication one), services regularly evolve in order to stick to users desires and new technologies. Also, home owners must be able to freely change their installations, despite the presence of pervasive services.
- **Dynamism.** Evolutions regarding pervasive services must be done without stopping neither the execution platforms nor the services that are not affected by the modifications.
- **Autonomy.** The computing environment should be almost invisible. Automatic techniques will be needed to dynamically configure services, networks and devices. Human intervention should be very limited and part of a continuous learning cycle.

In addition to these nonfunctional properties, it is also clear that many functional requirements are difficult to meet. For instance, context awareness and autonomy are very challenging goals that will require researchers to focus on domain-specific issues. Resolving these problems in general seems too distant today.

This section has given a general overview of the home automation smart systems domain. We presented general open research issues in this context. This paper focuses the

attention of a limited number of problems. We are concerned especially with offering an intelligent in-house environment where heterogeneous intelligent devices are deployed. This paper presents a smart home computing architecture that solves the device's heterogeneity problem. Furthermore, being based on a service-oriented architecture the solution presented handles also the device's dynamism problem previously mentioned. The next section presents the benefits of a service-oriented architecture for an in-house smart environment.

3. Service-oriented architectures

3.1 Presentation

Service-oriented computing (SOC) [8] is the software paradigm that has gained interest in the last couple of years in the detriment of component-based software. It has appeared as a result to some disadvantages of CBSE. In particular, it addresses issues like platform heterogeneity, tight coupling between involved parties.

SOC relies on the idea that a piece of software - named service - is made available to third parties. For that, services must declaratively define their functional and non-functional capabilities and requirements in an agreed (standard) machine-readable format. Based on service descriptions, automated service discovery, selection and binding is made possible in a service-oriented architecture. Service-oriented architectures (SOA) are a way of re-organizing software applications and infrastructure into a set of interacting services. Delivering software functionality as a service that may be configured, bounded at delivery-time may overcome current limitations constraining software use, deployment and evolution.

In order to provide certain functionality, once the service's development is finished, the service's provider publishes the service description (WSDL, OWL-S etc.) in service registries (or service broker depending on the adopted vocabulary). A candidate service consumer (another application, etc) will query the registry in order to discover the appropriate service responding to all his requests. The binding is realized only when an agreement regarding the service's delivery is established between the involved parties. In this negotiation phase, a new actor may intervene - the service certifier. His role is to certify the correctness of the information offered by the service provider.

Services are the basic unit of reuse in the paradigm software-as-a-service [9]. Service composition is one of the main concerns of the application development process. A service composition combines services in order to achieve a business goal or to solve a scientific problem. The composed service may itself be published, the process becoming recursive .

A SOA architecture is designed in such a way for that a client has no need to know where a service is executed - property named *location transparency*. Furthermore, the service's clients doesn't need to know what is the service execution platform - services are *technology neutral* - as long as they can interact with the service in a standard way.

SOC seems to be interesting solution in order to take benefit of all the intelligence brought in our home environment by the intelligent devices we employ in this environment. Exposing intelligence at service is a new trend in today software systems.

We consider that a service-oriented architecture is the right solution for creating the smart computing environment of home automation systems. Many research laboratories are studying the benefits taken from SOA architecture in this context [10].

In our vision, SOA architecture for the home automation will help to solve the heterogeneity problem and provides meaningful modalities to handle the device's dynamism.

The value added by a service-oriented architecture for a pervasive environment resides in a homogeneous vision where devices are depicted as service providers and in the same time as service consumers. The intelligence of the overall environment is offered by the composition of available services. Furthermore, SOA permits to dynamically discover and compose new available services offered by newly deployed devices or by devices that have changed location.

This paper presents a proposition of a smart SOA home computing environment based on OSGi and Web Services standards. The next section presents a closed analysis of the envisioned computing architecture.

3.2 Our solution

As illustrated by Figure 1, we are working on innovative architectures where home devices are represented as service providers and requesters [14]. Devices are structured into 3 categories:

- Electronic devices incorporated in the house (for instance, controllable shutters or lights) providing basic services to sense and act on the environment,
- Gateways providing computing resources allowing to run high level services aggregating basic services brought by the previous type of devices,
- Rendering devices (TVs, smart phones or PDAs for instance) allowing users to interact with home services and, possibly, to administrate them. Home owners will use them equally to interact with their environment (depending on location, convenience, etc.).

The whole architecture comprises all consumer electronic devices. All these devices are connected to Internet and to some service providers through a local Internet gateway and a web portal. The central actor in this house is the service-oriented middleware. His role is to deal with the service discovery and communication problem between every house's devices. It has to handle the inherent dynamism of all connected devices. In fact, some devices are very static (e.g. TV or Camera) and some others are much more dynamic (e.g. MP3 Player or PDA). The middleware has to take into account all these kind of devices and their mobility. Another big challenge, related to the middleware, is its capacity to enable the whole heterogeneous devices to interoperate.

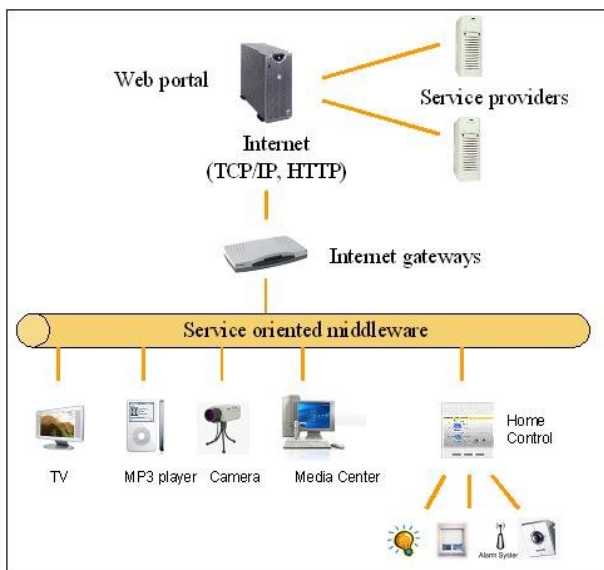


Figure 1 : Future architecture

Technically speaking, we can identify two types of devices in the home; those compatible with a service-oriented technology (like Jini or UPnP for instance – see www.jini.org and www.upnp.org) and those not yet integrating SOC capabilities. In order to be used in a service-oriented architecture, the latter have to be managed by a third party taking care of both the publishing and binding processes. Service-oriented technologies are however numerous and often not compatible, fact that raises thorny integration issues.

4. Modeling Analysis

In the previous architecture, gateways are smart equipments aggregating a number of electronic devices. The purpose of

a gateway is to coordinate multiple devices and to ensure natural, sometimes invisible, interactions with the users. These interactions are obviously directed by high-level goals that can be set either by the user or by Internet services the user has subscribed to. Some electronic devices can be exclusively connected to a given gateway. For instance, a home control gateway can coordinate the behaviors of specific devices like shutters, heaters or lights. This approach meets proven market constraints stating that most manufacturers won't provide access to their devices, neither to any operator nor to another device (for safety, security and obviously business considerations).

Schneider Electric (www.schneider.org) has already developed a home control gateway enabling occupants to remotely control or configure a set of automated devices by entering a single command. For instance, one can press a single button to arm a home security system, control temperature gauges, switch appliances on /off, control lightning, etc. This is called a **scene**. In the current implementation, electronic devices are connected to the gateway through wireless networks.

In this paper, we focus our work on the home control gateway and the scene aspect. As shown in Figure 2, we are currently working on modeling this specific computing area.

4.1 Devices

The main goal of a home control user is to control his house, more precisely electronic devices included in his house. During the modeling phase of this work, we first classify electronic devices into three main categories :

- The first one concerns sensors. These devices are intended to sense the whole environment and convert their observations into numerical data. They periodically harvest information and store them. These devices are able to interact with other software entities in two main different ways: push mode (periodical message sending), and pull mode (response to a request). An example of such devices is a temperature sensor.
- The second category concerns controllable devices. These devices are able to perform actions on the environment. They could be control by other software entities and expose for this APIs. The only way of interacting with these devices is to call an API operation. An example of such devices is a shutter or a light. The light API comprises two operations: turn on/off.
- The third category concerns devices that are part of both previously mentioned categories. These devices are able to sense and act on their surrounding area. They combine the possibility of controllable and sensor

devices. Generally speaking, the sensor and the controllable part of a device have some strong functional relations. For example a heater generally integrates a temperature sensor to control the heating power. These devices are thus able to integrate and expose some higher-level operations.

4.2 Business Services

But the device part of the model is clearly not sufficient to enable home control. Obviously, with the increasing number of devices, users will not be able to control separately all devices. This observation leads us to model software entities, called business services, which are able to automate specific behaviors.

Business Services are entities that are able to retrieve information through sensor devices and take action on controllable devices. A business service could generate actions on multiple devices and harvest information from multiple sensors. They contain the “clever” part of the Home Control gateway.

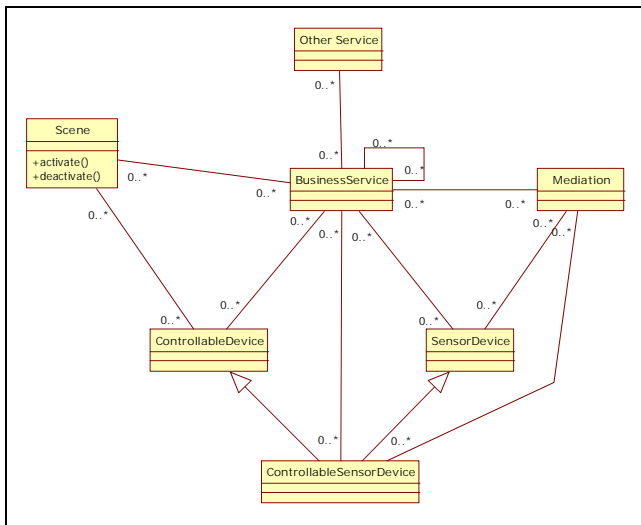


Figure 2 : Home control gateway model

As business services contain the business logic part of the whole system, they need to be written by programmer specialized in the particular field of their action. For example a temperature manager, needs to be written by specialists of this domain.

As we point out in previous sections the flexibility needs, a business service is able to call other services. This feature is very important to gradually raise the abstraction level of such business services. To come back on the temperature manager, it could be split in different temperature managers in charge of each room, and one coordinator. The

coordinator offers higher-level services than each room temperature manager services.

We also provide a means to call other types of services, which are generally not hosted in the house. For example, a business service is able to use a weather forecast service, hosted in a remote place. Allowing this possibility raises some big issues on service technologies integration, which will be details in further sections.

The communication between sensor devices and business services could potentially be very heavy, depending on the amount of collected data. In order to lighten these communications and to present data in more usable forms, we introduce software entities called mediators [11]. This entity performs transformation and aggregation operations upon data stemming from Sensor Devices and other mediators.

4.3 Scenes

The home control domain relies on the necessity to make the user life more and more easier. This implies that users will have very simple interactions with the system. In order to limit the number of interactions, industrials like Schneider Electrics create the scene concept, which basically represents a configuration of a set of house devices. A very simple example is a scene called “I leave”, which is able to turn off every lights and close every shutters. In this case, the scene could directly called actions on devices.

But scenes also refer to some much more complicate home configurations, which involve not only devices but also business services. A scene is thus software entity acting on devices and business services. A scene does not contain the business logic. It corresponds to the user interface of the system and aggregate calls to a set of devices and business services. An example of such a more complicated scene is the “I am on vacation” scene that turns on the alarm manager, the presence simulation and turns down the temperature goal.

5. Current implementation

This section shortly presents the technical solutions we employed for the current implementation of our demonstrator. A demonstration is available on our web site at <http://www-adele.imag.fr/Homega/>.

5.1 Global architecture

Our home automation control system is composed of five parts (Figure 3). The central part is a dynamic service platform, the execution environment of service applications.

The service platform hosts services representing device functionalities, business services and so on. In order to tackle heterogeneity, it communicates with other service technologies through specialized services called bridge. Bridges functions as mediator between the dynamic service platform and other service technologies like UPnP, Jini, Web Service susceptible to be used in home automation systems. They have two roles : first they expose internal services to be accessible in other service technologies; secondly, they allows access to services implemented in other service technology.

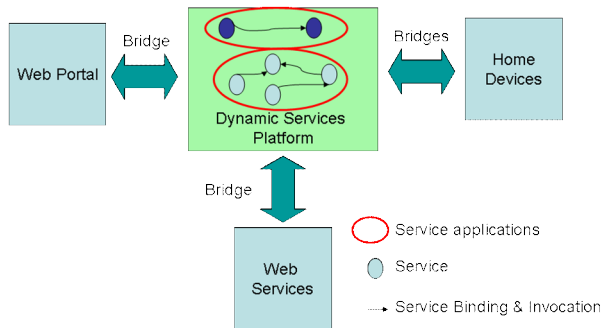


Figure 3 : System Overview

The services hosted by the service platform can be started 'on demand'. Moreover, the service platform permits to remove, update, deploy new services without interrupting the execution of already deployed applications.

In our system, the dynamic services platform communicates with home devices to create service applications for home automation. These devices are exposed as services and interact with the service platform through a bridge. Consequently, devices need to support non-functional properties to allow bridges to discover their services dynamically :

- **Discoverability:** The gateway needs to discover all house's devices in order to manage them.
- **Dynamism:** The gateway needs to be informed of all departures and arrivals of devices.
- **Identifiable:** The gateway must be able to identify each device and their functionalities. This requirement includes the necessity to classify each device in one of the previously mentioned categories.
- **Remote Invocation:** In order to control the devices, the gateway will have the capacity to remotely invoke their provided functionality.

In the current implementation of our demonstrator, devices are simulated on the platform. This allows us to provide a simple mean to add and remove devices through a web interface. In our architecture, it is easy to change the simulated device by other device type. For example, we can change with UPnP devices which met all requirements previously mentioned. We are also currently working with Schneider Electric on enabling the control of their proprietary devices.

Furthermore, service applications running on the home gateway needs to interact with Web Services. To achieve this, the dynamic service platform uses a bridge allowing the transparent access to Web Services.

Finally, the web portal allows management the service platform, and the deployed service applications. To launch applications and to manage the service platform, the web portal calls Web Services exposed via a bridge. This bridge expose service applications, a platform management service and a registry (listing the available services).

5.2 The Dynamic Service Platform

This section describes the execution environment of service applications. We used an implementation of the OSGi specification as dynamic service platform. Moreover, we improve OSGi on two points:

- We provide a service-oriented component model to help service application development.
- We develop bridges, allowing the interaction between OSGi and other service technologies.

All services running on the dynamic service platform are implemented using a new service-oriented component model, iPOJO developed in our team.

This section presents OSGi, our service-oriented component model named iPOJO, and the mechanism to handle heterogeneity - bridges.

5.2.1 OSGi

OSGi Alliance (<http://www.osgi.org/>) is a consortium providing an open specification of a service gateway. At the beginning of OSGi, its target was mainly residential networks and industrial gateways. However, with the evolution of the specification, several other domains used OSGi like plug-in application (Eclipse IDE for instance), application servers (Jonas on Demand [16]), vehicle gateways...

The OSGi specification define a Java based centralized service platform and tools for service deployment. OSGi uses a deployment and packaging unit called bundle. Bundles are a Jar file containing java classes, resources and metadata. Bundles contain service providers or service

requesters (or both). The OSGi platform provides a dynamic modularity layer. This layer manages bundles and provides mechanisms to install, start, stop, update, and uninstall bundles, without stopping other bundles, nor the platform.

On the top of this modularity layer, OSGi provides a service layer. When a bundle is started, it is able to publish services, to discover and use other services. To achieve this, OSGi supplies a service registry containing service descriptions, and primitives to publish, discover and invoke services.

OSGi services are specified through a Java interface and attached properties. The properties are often described in the documentation of the interface. Thus, to discover a service, a bundle needs to know its interface name. A LDAP request on properties can improve the service discovery.

OSGi is a dynamic service platform. Indeed OSGi supports the dynamic apparition and departure of services. To support this, OSGi used its modularity layer (for the dynamic class loading and unloading) but also its service layer. Moreover, the OSGi framework provides an event channel diffusing the arrival and the departure of services. Thus, service consumers can listen to this channel and are aware of the newly available services.

Nevertheless, OSGi has a big drawback. The development model is very hard to whelm. Indeed, the dynamics of OSGi platform complicates the development of services. Besides, to implement the business logic, developers need to programmatically manage the service dynamism.

5.2.2 *iPOJO: a service-oriented component model*

To tackle the main OSGi drawback, we propose a service-oriented component model managing the service dynamics. It deals with service lookup, invocation, publication and furniture mechanisms. *iPOJO* (see <http://incubator.apache.org/felix/>), for injected POJO, provides a new way to develop OSGi components. Its main goal is to simplify the OSGi component's implementation by managing transparently the dynamics of the environment and other non-functional requirements.

The *iPOJO* framework allows developers to clearly separate the business logic (the POJO) and the non-functional requirements (managed by the *iPOJO* Runtime). To perform this, *iPOJO* provides a simple and extensible component model based on the concept of POJO. It state for Plain Old Java Object and advocates the idea that the simpler the design, the better.

5.2.3 *Meditation between service technologies: Bridges*

A bridge is a special service allowing interactions between OSGi services (implemented by *iPOJO* components) and other services (not in the same OSGi framework). To make these interactions transparent for the service consumer, we identify two types of bridge :

- One allowing a service from other space of services (other service technologies or remote OSGi framework) to interact with OSGi services.
- One allowing OSGi service consumer to interact with a service from an other space of services.

These two types of bridge have to deal with different functions to fulfil their functionalities. For the first type of bridge, it needs to :

- expose internal OSGi services in the other service technologies;
- manage the OSGi events and transform it properly into the targeted service technology;
- create a proxy between a service provider and a service consumer coming from two different spaces allowing their interaction.

The second type of bridge aims to allow transparent interaction between an OSGi service consumer and a service provider from an other services space. This bridge has to deal with :

- the discovery of services executing in the targeted services space;
- the creation of a proxy for each external service, exposing this services as OSGi services and allowing transparent interaction with it.

We have already developed a prototype of bridge to expose OSGi services in web services. This prototype is able to automatically expose a web service and interpret a request coming from a web service. It transforms the web service request into understandable request for OSGi services.

We also have developed a bridge to allow the communication between an OSGi service consumer and a particular web service (*globalWeather*). This bridge is specific to a particular web service, and we need to redeveloped it for each targeted web service. We are currently working on enabling the automatic generation of these bridges.

6. Conclusion

In this paper we presented our vision of the home automation domain. This particular area raises a lot of issues which actually remain unsolved. Our main contributions are our propositions for the architecture and modelling part of the home control domain.

In our architecture, we propose a global middleware connected to internet, devices and gateways aggregating devices. This middleware is intended to facilitate the device coordination, enable discovery and dynamic use of other devices and provide an execution platform for high level business services. The proposed architecture meets proven market constraints which state that most manufacturers want control the access to their devices (for safety, security and obviously business considerations).

In the modelling part of this work, we provide a means to build high level business services. We clearly separated different concerns and identified each actor. This separation allows the integration of different services from different actors.

Another contribution of this work concerns the underlying infrastructure which provides the execution platform. This infrastructure deals with the inherent dynamics of devices and services in general, enabling business service developers to focus on the implementation of their business logic.

Nevertheless, this work does tackle neither the security part nor the autonomic behaviour which are necessary to enable this business model. We are currently working on facilitating the development of this type of applications. On the gateway infrastructure level, we raise the API abstraction and improve the life cycle management. We are also working on a Model Driven Architecture (<http://www.omg.org/mda/>) tool to facilitate the development of such applications. Enabling the autonomic behavior remains unsolved, however we are exploring solutions based on enrich architecture models.

7. References

- [1] M. 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] P. Lalanda, "E-Services Infrastructure in Power Distribution," *IEEE Internet Computing*, vol. 9, no. 3, 2005, pp. 52–59.
- [4] P. Lalanda, L. Bellissard and R. Balter, "Asynchronous Mediation for Integrating Business and operational Processes," *IEEE Internet Computing*, vol. 10, no. 1, 2006, pp. 56–64.
- [5] S. Helal, W. Mann, H. El-Zabadani, J. King, Y. Kaddoura, E. Jansen, "The Gator Tech Smart House: A Programmable Pervasive Space," *Computer*, vol. 38, no. 3, pp. 50-60, Mar., 2005.
- [6] V. Stanford. "Using pervasive computing to deliver elder "care." *IEEE Pervasive Computing*, 1(1):10–13, 2002.
- [7] P. Schramm, E. Naroska, P. Resch, P. Platte, H. Linde, G. Stromberg, T. Sturm, "A Service Gateway for Networked Sensor Systems," *IEEE Pervasive Computing*, vol. 03, no. 1, pp. 66-74, Jan-Mar, 2004.
- [8] M. P. Papazoglou, "Service -Oriented Computing: Concepts, Characteristics and Directions" *wise*, p. 3, Fourth International Conference on Web Information Systems Engineering (WISE'03), 2003.
- [9] M. P. Papazoglou and D. Georgakopoulos. Service-oriented computing. *Commun. ACM*, 46(10):24–28, 2003.
- [10] M. Aiello. The Role of Web Services at Home. *IEEE AICT/ICIW 2006*..
- [11] C. Marin, P. Lalanda, D. Donsez, "A MDE approach for power services development", *Int. Conf. on Service Oriented Computing (ICSOC 2005)*, Amsterdam, December 2005.
- [12] C. Wege, "Portal Server Technology", *IEEE Internet Computing*, volume 6, number 3, p73-77, 2002.
- [13] L. Tarrini and V. Miori. LIGHT: XML-Innovative Generation for Home Networking Technologies. *ERCIM News*, 62, 2005.
- [14] P. Lalanda and J. Bourcier, "Towards autonomic residential gateways", *IEEE International Conference on Pervasive Services (ICPS 2006)*, June 2006.
- [15] Masahide Nakamura, Hiroshi Igaki, and Ken-ichi Matsumoto, "Feature Interactions in Integrated Services of Networked Home Appliances -An Object-Oriented Approach-" *Int. Conf. on Feature Interactions in Telecommunication Networks and Distributed Systems (ICFI'05)*, pp.236-251, July 2005.
- [16] Mikael Desertot, Didier Donsez and Philippe Lalanda, "A Dynamic Service-Oriented Implementation for Java EE Servers ", *In proceedings of the 3th IEEE International Conference on Service Computing (SCC'06)*, 18-22 September 2006, Chicago, USA