Autonomic Management of Pervasive Context

Colin Aygalinc, Gerbert-Gaillard Eva, German Vega, Philippe Lalanda

Autonomic management of pervasive context

Colin Aygalinc, Eva Gerbert-Gaillard, German Vega and Philippe Lalanda
Grenoble University
220 rue de la Chimie, Grenoble
France
firstname.lastname@imag.fr

***Abstract*— Pervasive computing promotes environments where smart, communication-enabled devices cooperate to provide services to people. Due to their inherent complexity, many pervasive applications are built on top of service-oriented platforms, providing a set of facilities simplifying their development and execution. In this demo, we present such a platform, iCasa, extended with an autonomic, service-oriented context module. It is validated on diverse applications developed with the Orange Labs in the health domain.**

Keywords—autonomic management; pervasive context; service-oriented computing; health.

I. PROPOSED APPROACH

A growing number of smart, communication-enabled devices are integrated in our living environments. This is essentially due to major advances in hardware and networking technologies, which make sensors more powerful, cheaper and smaller in size. Such digitalized environments, said to be pervasive or smart, are increasingly accepted in all places where social or professional activities take place. They support the creation of new added-value services that are delivered anytime and in a non-obstructive way. This new form of computing is raising huge economical and societal expectations in domains like manufacturing, buildings and homes, energy, commerce and even healthcare. From this last point of view, advances in pervasive computing are very promising and could, for instance, provide new services enhancing the potential of the elderly and patients to live and cope at home.

Applications integrated in such smart, pervasive environments are context-aware by essence. It means that they are able to adapt their behaviors and the provided services according to environmental conditions [1]. The notion of environment should be taken in its broad sense here. It includes any information that can be of interest for an application like, for instance, physical quantities, locations and expectations of human beings (implied or not in the functions provided by the application), the software itself, and even remote digital resources. Applications are also able to act on their context and, say, change physical quantities to meet their objectives.

We readily acknowledge that building a context module is a complex task. Multiple design trade-off decisions have to be made and implemented regarding data access, synchronization mechanisms, knowledge representation, reasoning, and presentation. A common solution to alleviate these difficulties is to use a context management framework. The purpose of such framework is to deal with a number of generic features like information synchronization or publication. As will see here after, most existing frameworks suffer from major flaws with regard to our use case. In particular, many impose an unwanted overhead in terms of representation and also execution time.

In our work, we concentrate on the fog level, which means that we focus on the context-aware part of the code that is executed as close as possible to the devices. This corresponds, for instance, to short-term supervision in pervasive environments. At that level, reactivity is an essential aspect. The supporting framework has to deal with dynamic environments where resources can appear and disappear without notice.

We have integrated an autonomic, service-oriented context module in our pervasive platform, named iCasa [2]. This platform was ideally suited for such extension since it is based itself on service orientation. Precisely, it is builds upon the Apache service-oriented component model, iPOJO [3].

Context appears as a dynamic set of services. Depending on the sources availability and the platform needs, different services are published in and withdrawn from the platform service registry. They are then opportunistically used by the pervasive applications, coded in iPOJO. As illustrated, the context module receives goals (that can be of different levels of abstraction) from the platform that are used for context self-adaptation. Another major point is that the context module tracks any contextual modifications and sends an event to alert applications using it. The overall approach is illustrated by Fig. 1.

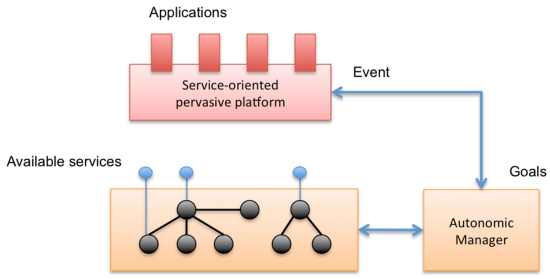

Fig. 1. Overall approach.

II. PROPOSED DEMONSTRATION

The proposed demonstration is based on the iCasa environment, which is made of three tools:

- An IDE based on an eclipse plug-in. It supports the development of iPOJO-based application.
- An execution platform, based on OSGi, running on a home gateway, which hosts several applications and offers dynamic deployment facilities.
- A smart home simulator (see Fig. 2) that supports the execution of predefined scenario, in order to quickly test pervasive applications.

In this environment, several applications have been developed in various domains, including safety, comfort and healthcare at home. These applications differ in terms of technical requirements and needed context. In our demo, we will consider two different applications. Actimetrics is a home care application measuring through the execution platform, and analyzing, through a cloud infrastructure, the motor activity of elderly. It tracks behavioral changes to early diagnose degenerative diseases like Alzheimer. We also designed a LightFollowMe application: for each room of a house, it controls the light depending on the presence of a user, and adjusts intensity according to the current moment of the day.

For these applications, we will present how the context is defined, deployed and run on the pervasive platform. Context at runtime will be presented in a graphical form. Let us focus on the LightFollowMe application. The context needs for this application can be decomposed as it follows. First, the application relies on an appropriate abstraction of lightning devices, spatial zones and time (modeling). Such abstraction must be synchronized with the physical devices (gathering). Then, the application requires knowing which device is in which zone (enrichment). The application computes physical parameters: presence and illumination per zone (processing). Finally, the application necessitates retrieving all this information (disseminating) and applying its business logic to affect the environment.

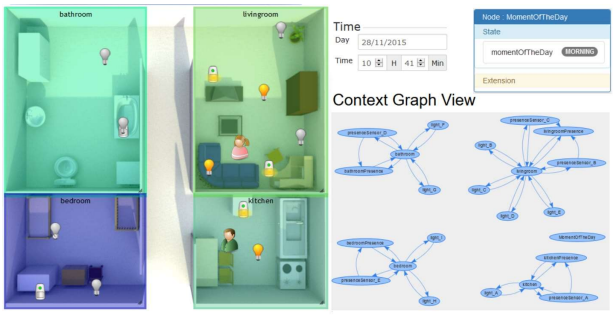

Fig. 2. *Light Follow Me* application – iCasa simulator overview.

Our framework gives an understandable representation of the global context. The bottom right corner of Fig. 2 presents the graph of the model displayed on the iCasa simulator: devices are gathered by location, each room is enhanced with a physical parameter aggregating and synthesizing presence status, and an independent entity provides the moment of the day. In addition of the graph view, the simulator can display state properties and state extensions of an entity. This functionality is shown on Fig. 3 with light_A.

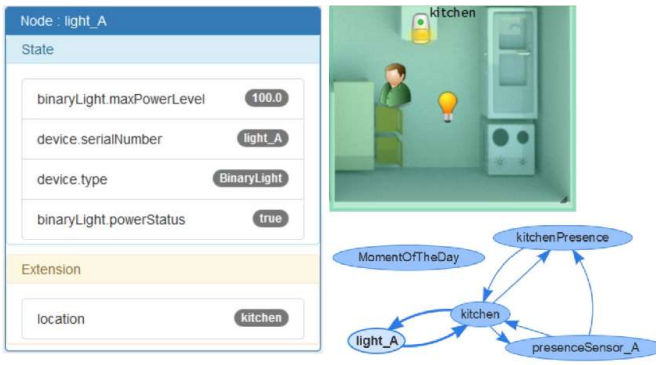

Fig. 3. *Light Follow Me* application – zoom on the kitchen.

Our framework gives significant results from application development perspective. Inevitably, building the context adds an additional development task and the resulting architecture is more complicated. However, it's easier to implement applications on the top of it. The context is shared among them, is extensible and can be adjusted to fit needs of an update. The whole software is more consistent, testable and maintainable.

REFERENCES

[1] M. Baldauf, S. Dustdar and F. Rosenberg, "A survey on context-aware systems," in International Journal of Ad Hoc and Ubiquitous Computing, vol. 2, no 4, pp. 263-277, 2007.

[2] C. Escoffier, S. Chollet and P. Lalanda, "Lessons learned in building pervasive platforms", in the 11th IEEE Consumer Communications and Networking Conference (CCNC), pp. 7-12, 2014.

[3] C. Escoffier, R. S. Hall and P. Lalanda, "iPOJO: an extensible service-oriented component framework," in Service Computing, pp. 474-481, 2007.