A software self-organizing middleware for smart spaces based on fuzzy logic

Charles Gouin-Vallerand, Bessam Abdulrazak and Sylvain Giroux
DOMUS Laboratory
University of Sherbrooke
Sherbrooke, Canada
{Charles.Gouin-Vallerand, Bessam.Abdulrazak, Sylvain,Giroux}@usherbrooke.ca

Mounir Mokhtari
Handicom Laboratory
Telecom SudParis
Evry, France
Mounir.Mokhtari@it-sudparis.eu

Abstract— Autonomic Computing Self-Configuration /Organization can simplify the complexity of software deployment, management and utilization in pervasive environments such as smart spaces. Software management complexity in pervasive environments is a serious problem, which leads to high management costs and slow down the broad utilization of the pervasive technologies. This paper presents our work on a self-organization middleware that reduce user manipulations and reasoning during software deployment and organization in smart spaces. Our approach is based on Fuzzy Logic and semantic description in order to support the self-organization reasoning on the environment context. (Abstract)

Keywords-component: Pervasive computing, Self-organization, Autonomic computing, Context Awareness, Fuzzy Logic

I. INTRODUCTION
The fast development of information and communication technologies is opening a large variety of new virtual services not only for valid people, but also for people with Special Needs (PwSN). Combining technical assistances allows elderly or people with physical/cognitive restriction, to benefit from augmented environments that are assisting them in their daily living activities.

In fact, these environments are using several devices and applications: sensors, effectors, human-machine interfaces, location services, etc., to give the proper assistances. They are applying the pervasive computing[1] and the mobile computing approaches to assist the users.

However, the complexity of the pervasive environment leads to serious difficulties. The high costs in the implementation and maintenance of the pervasive solutions slow the broad utilization of the pervasive.smart environments. The heterogeneity of the used devices and applications and often their lack of usability make their utilization difficult by domain-expert like caregivers.

More specifically, the configuration, the organization and the maintenance of the software applications in the environments are resource and time consuming since the important number of devices and their heterogeneities[2][3]. To organize the environment’s software in a suitable configuration, it is required to have a good knowledge of the pervasive environment, which is not the case for most non-expert people such as the healthcare givers or the elders.

At DOMUS Laboratory of the University of Sherbrooke and Handicom Laboratory of Telecom SudParis, we are steadily confronted to the problems of the complexity in the pervasive environments. The shared objectives aim to propose to elders and people with cognitive and/or physical impairments assistances through pervasive technologies. Since our targeted users are, most of the time, not domain-expert users and may have problems to learn or remember engineering concepts, the growing complexity of the new technologies curb the utilization of the pervasive approach to this population.

Facing these problems, our objectives are to simplify the software application provision to the environment devices and the organization of the application between the devices i.e. the software configuration of the smart spaces. As solution, we are proposing to eliminate most of the required manipulations by simplifying the information and feedbacks returned to the users and by moving the required context reasoning from the users to the environment systems. A reasoning engine based on semantic information and fuzzy logic replaces the user’s reasoning on the context. This solution allows to “automate” the majority of the user manipulation on the system, reducing in a significant way the complexity of the pervasive environments.

Therefore, we present in this paper our work on the self-organization and configuration of the software applications in pervasive environments and the developed Fuzzy Logic Organization Reasoning Engine (FLORE). Firstly, the Section 2 presents our research context and methodology. The Section 3 presents the related works on the self-configuration and self-organization. The Section 4 describes the self-organization and configuration middleware we are implementing and the FLORE. The Section 5 presents some results about the implemented middleware, Section 6 presents our future work and finally Section 7 concludes this paper.

II. METHODOLOGY
A. Pervasive Computing
In the context of our research on the self-organization and self-configuration, we are confronted to the problem of
defining what are the smart spaces, where they are starting and where they are ending. With the increasing works around the Open Space, Mobile computing and Internet of Things, it is now difficult to fix preliminary limits to a research project on the Pervasive computing. Therefore, we use a description similar to the pervasive community, where smart spaces or pervasive spaces can be described as:

- Physical environment with several user and non-user devices;
- Information processing or software applications are distributed in the environment devices, bringing closer the information processing to the information sources and contexts;
- Dynamic environment. Users, devices and applications can enter and leave rapidly without defined patterns;
- The used technologies: software, communication protocol or material architecture are heterogeneous;
- Several users are involving in the environments. Each user has their profiles and preferences.

Other projects[4] in our research group are focusing on the context awareness or the collaboration between agents in the Open Space and the Mobile Computing. As we are focusing our work around specific and defined pervasive environment such as smart home or apartment, we aim to merge our work in the future when they will reach maturities.

B. Autonomic Computing

One major way to reduce the complexity of the smart spaces and simplify the utilization, the management and the maintenance, is to reduce the amount of user manipulations and reasoning on the system. The Autonomic Computing[5], introduced by IBM in 2001, is an initiative where technologies use and interconnectivity are made simpler by abstracting the growing complexity of the computer systems. This initiative is based on four concepts called the four ‘selfs’:

- Self-configuration enables to deploy large applications requiring solely a few simple actions. Intelligent mechanisms cope with deployment complexity. These mechanisms can be based on a set of rules or a semantic description of applications, depending on the used approach. Self-healing and Self-protection make systems more secure and less fragile to occurring physical or software faults. Finally, self-optimization creates more proactive systems that are always searching for opportunities to enhance their performance, for instance by balancing the processing loads dynamically.
- We propose in earlier work our vision of the Autonomic Pervasive Computing[6] and an initial self-configuration solution[7]. This present paper focus on our fuzzy logic reasoning solution for software application provision in pervasive environments.

C. Context Description

To move the organization/configuration reasoning to the environment’s system, the system must have the required information on the environment context to apply the right reasoning. A powerful way to represent this information is by using semantic language, describing the environment information and links residing between the concepts. This semantic information can be further used to improve the precision of the reasoning or inferring new semantic information. Therefore, a popular way to reason about semantic information is by creating description logic rules in semantic frameworks. In our work, we decided to describe the pervasive environment in the Web Ontology Language (OWL), and implementing the resulting description in the JENA framework[8]. The used environment description approach is described in [9].

Beside the environment description, the user profile is an important concept in the context description. Depending on which user is using a specific device and what are his restrictions, state and preferences, the device applications can adapt their provided services. Moreover, in the application provision and the application organization, the user profile allows enhancing the precision and the capacities of the application deployment and management. For instance, the pervasive space can target the most usable device to deploy an application for a physical impaired person in an environment where several devices, with different capabilities, are present.

D. Fuzzy Logic

The contextual information of a pervasive environment cover a large variety of data type, ranging from quantitative information such as the room’s temperature to qualitative information such as the user state. The Fuzzy Logic approach[10], based on the fuzzy set theory, allows to easily compare quantitative and qualitative information in a same set of reasoning rules. Moreover, a reasoning algorithm based on the Fuzzy Logic doesn’t need to have an accurate knowledge of the model and can work with a high level of imprecision, which is the case of the pervasive environment, where it can be difficult to describe precisely the model and get accurate data. Finally, Fuzzy Logic gives us the support to describe situation where no clear evaluations and statements can be carried out, like saying that a device resources are used or partially used or in the “classic” case of the warm water in industrial processes[10].

Therefore, we decide to use Fuzzy Logic in the implementation of our self-organization solution and use it with other reasoning techniques like description logic. By using Fuzzy Logic APIs like JFuzzyLogic[11] and description format like the Fuzzy Control Language (FCL), it allows the self-organization middleware to be rapidly scalable and adaptable to new context.

E. Service Oriented Architecture

Pervasive environments are not only dynamic on the device and communication perspective, several applications in these environments can been modified frequently. In some contexts, new software or plug-in can be deployed into the environment, according user needs. Moreover, as the complexity of the pervasive environment is increasing, their software applications are increasing too.

By using the Service Oriented Architecture (SOA) approach in the pervasive software applications, it is possible to enforce the modularization of the software and reduce the
excessive software coupling. By dividing the applications in several modules, which are communicating and exchanging through services, the code coupling is reduced. Moreover, as the smart spaces are highly dynamic, the SOA approach allows loading, unloading or updating modules dynamically and calling the module services on demand. In our current work, we are using the OSGi framework [12], which is offering the SOA needed functionalities:
- Dividing software in several modules called bundles;
- Sharing code bases and services between the modules;
- Giving service listening and handling mechanisms;
- Giving bundle management and module dependence functionalities.

F. Core Architecture

We designed a distributed multi-level architecture (L1…Ln) which provide structures, context descriptions, ontologies and services for the development of Context Aware applications.

It is instrumented with customized nodes. Each Li node (mobile or fixed) offers a set of Li services (Figure 1). L1 nodes are hardware abstraction gateways that enable the system to interact with L0 real world data (sensors and actuators). Lj>1 nodes are concentrators that aggregate the services offered by Li<j nodes in a Zone (location of influence). The hierarchy can be based on actor needs: location (e.g. L1 is for devices, L2 for a room, L3 for a floor…), processing requirements (e.g. Li>1 node concentrates instances of Li-1 nodes, to perform load balancing), security, confidentiality and ethical concerns.

L0 devices are providing information from sensors to build one part of the context. These data are retrieved by L1 devices and participate in the creation of the first Context Aware functionalities related to a component and its neighborhood. L2 devices are aggregating the information from the L1 devices, building a larger vision of the context for a given zone of influence (a Zone). Finally L3 to Ln layers’ devices are aggregating below layers’ contexts to provide a global vision of the known environment context.

The architecture used a meta-ontology description [9], both extensible and adaptable, that is used by the different layers to describe the context. The Meta-ontology description elaborated the 3 high level concepts: Being, Environment and Dynamic that better describe the Smart Space universe. It also included a classification of sub-concepts based on international classifications and standards. It also introduces a concept of referentiality in location (zone) and time.

Furthermore, we have developed a basic SOA OSGi-based middleware on top of presented architecture, in which all smart space’s components (nodes, sensors & actuators) are represented as services. Moreover, this core middleware provide services to manipulate context’s concepts in ontologies and are accessible to the different software applications build on top of the architecture.

III. SELF-ORGANIZATION MIDDLEWARE

A. Overview

To reduce the users manipulations during the software organization/configuration of a pervasive space and to replace their reasoning on the system, the solution needs to implement 4 main functionalities:
- The capacity to deploy new applications and managing the existing ones in the environment devices;
- The capacity to collect environment context information and keep them;
- A reasoning mechanism to evaluate the context according to the users or environment needs;
- An administration tool i.e. a graphical user interface, with which users can organize the environment software.

To coordinate the software organization and the deployment between the device and the organization/deployment request, we propose an architecture based on coordinators and device nodes (Figure 2), which uses organization, management, ontology and discovery modules.

Figure 1. Overview of the overall Architecture used for Open Smart Space Development

From these required functionalities appear other embedded requirements. A device discovery mechanism must be used to discover the environment devices. As the environments are dynamic and the system didn’t know, at priori, which device are in the environment, a device description extraction service must be available to gather the

Figure 2. The self-organization middleware architecture with its two main components: the environment manager coordinator and the device node.
environment device descriptions and add them to the coordinators’ ontology.

To simplify the evaluation of the dependences, the proposed solution resolves the software dependences and the dynamic dependences over remote services or dynamic device during the software installation in the device nodes. However, the static dependence over embedded or wired peripherals are described in the device description and are used by the organization reasoning engine, to choose the right devices, by matching the application dependences with the device functionalities.

Moreover, for scalability purpose, the proposed solution is not bind to a single communication protocol, but can use multiple communication protocols in a transparent way for the applications. To address the interconnectivity problems, any pervasive solutions should implement mechanisms to abstract the communication layer from the application one and “hide” the used protocols. The exact implementation of this proposed architecture is presented in the following sections.

B. Fuzzy Logic Organization Reasoning Engine

As presented above, the objectives of the Fuzzy Logic Organization Reasoning Engine (FLORE) is to match the needs of the applications to deploy in a pervasive environment with the context and the resources of the environment’s devices. Fuzzy Logic allows to “fuzzify” the reasoner’s input i.e. transforming the numeric values into fuzzy values related to a quantitative set, compare and process them through a set of reasoning rules then doing the “defuzzification” of the outputs in numeric values which can be used by systems.

As the needs, context and the resources are a mix of quantitative and qualitative data; the Fuzzy Logic allows to compare them in a “fuzzy” perspective for every type of input then have numeric values as output, which will be used by the environment organization manager to choose the most capable device for a giving software. In fact, the numerical outputs, ranging from 0 to 100, represent the Device Capabilities Quotient (DCQ) i.e. the optimality of a device facing the application needs. The higher the quotient value, the closer is the device to the optimal device target.

Thus, we evaluate the DCQ based on several values: the contextual zone location of the device facing to the contextual needed zone of an application, the utilization of the hardware resources: the Central Process Unit (CPU), the Random Access Memory (RAM) and the Permanent Memory storage (PMS), and finally its embedded or wired peripheral. Contextual zone are environment division based on the activity zones, physical zones or logic zones. For instance, the kitchen or living room are contextual zones and can also be divided in several sub-zones like the fridge’s zone or the TV’s zone. The proposed FLORE consider the relation between the environment zones such the parent-child relations, the zone adjacent relation or the mutual parent relation. Thus, the FLORE algorithm implementation is divided in several steps (Figure 3). Each FLORE’s step is done for each application to deploy, which is contained into the organization request.

1. ∀ a ∈ ApplicationsToDeploy :
   1.1. Extract Device’ set from the coordinator’s ontology : d ∈ Device | {a.cpu ≤ d.cpu} (a.ram ≤ d.ram) (a.pms ≤ d.pms) (d.status = “started”);
   1.2. Keep Device’ set : d ∈ Device | a.neededPeripheral d.peripheral;
   1.3. ∀ d ∈ Device1 :
       1.3.1. Evaluate zone relations : r = evaluateZoneRelations(d,a) where r ∈ [1,4];
       1.3.2. Add {q,d} to DeploymentQuotient’s set | q = evaluateDeviceWithFuzzyLogic(d, a, r);
   1.4. deployApplication(a, d) | q = max DeploymentQuotient;
   If the deployment fail, take the next q = maxDeploymentQuotient;

Concerning the device and application zone matching, we evaluate the adjacent relation between the application zone target and the zone where device is located or related and the parental relation between the zones. For each relation type, we assign a value ranging from 1 to 4, signifying respectively dissociated, same parent, directly adjacent and same (Figure 4). These values correspond to fuzzy singleton set, which will be used by the FLORE.

\[
\text{evaluateZoneRelations}(d,a) \text{ where } d \text{ is a Device and } a \text{ is an Application :}
\begin{align*}
1. & \quad \text{d.zone} = \text{a.zone} \Rightarrow \text{zoneRelation} = 4 \text{ (i.e. same);}
2. & \quad \text{d.zone.parent} \land \text{a.zone} \Rightarrow \text{zoneRelation} = 4 \text{ (i.e. same);}
3. & \quad \text{d.zone} \land \text{a.zone} \Rightarrow \text{zoneRelation} = 3 \text{ (i.e. adjacent);}
4. & \quad \text{d.zone} \land \text{a.zone.parent} \Rightarrow \text{zoneRelation} = 2 \text{ (i.e. same parent);}
5. & \quad \text{if no values as been assigned to zoneRelation then zoneRelation = 0 (i.e. dissociated)};
\end{align*}
\]

To evaluate the device resources utilizations and the zone relations with the Fuzzy Logic, the FLORE uses the jFuzzy Logic API[11], a Java Fuzzy Logic framework which implements a Fuzzy Control Language (FCL) evaluator. Therefore, we define with the FCL, four device’s resource evaluation sets: free, used, very used and saturated. Each set has their corresponding membership functions, which are Gaussian functions (Figure 5).

\[
\text{FLORE resource evaluation sets = } (A, m) \mid m : A \rightarrow [0,1] \text{ where } \forall x \in \mathbb{R} \mid [0, 100],
\]

\[
m(x) = \frac{1}{\sqrt{2\pi}\sigma^2} \left(1 - \frac{(x-\mu)^2}{2\sigma^2}\right) \text{ with }
\]

- \(\mu = 0\) and \(\sigma = 8\) for saturated set;
- \(\mu = 33.33\) and \(\sigma = 8\) for very used set;
- \(\mu = 66.66\) and \(\sigma = 8\) for used set;
- \(\mu = 100\) and \(\sigma = 8\) for free set.

As presented above, the zone relation evaluation results are fuzzified following singleton sets, where they are classified in the same, adjacent, same parent and dissociated sets. To qualify the DCQ, FLORE classify the devices in four final sets, representing the deployment viability of the device: impossible, not-optimal, sub-optimal and optimal deployments. These defuzzification membership functions
are the same Gaussian functions as the resources fuzzification sets.

The classification process is done by applying nine fuzzy rules on fuzzified values (Figure 6). During the rule evaluations, FLORE used the minimal value as “AND”, the minimal value as activation method and the maximal value as accumulation method. Finally, the deployment viability defuzzification is done by finding the centroid value of the accumulation set. The resulting value represents the DCQ for the evaluated device.

1. IF CPUSaturation IS free AND RAMSaturation is free AND devicePMSaturation is free AND zoneSimilarity is same THEN deviceViability IS optimal WITH 1;
2. IF (CPUSaturation IS free OR RAMSaturation is free OR PMSaturation is free) AND (CPUSaturation IS used OR RAMSaturation is used OR PMSaturation is used) AND zoneSimilarity is directlyAdjacent THEN deviceViability IS subOptimal;
3. IF CPUSaturation IS veryUsed OR RAMSaturation is veryUsed OR PMSaturation is veryUsed THEN deviceViability IS notOptimal;
4. IF CPUSaturation IS saturated OR RAMSaturation is saturated OR PMSaturation is saturated THEN deviceViability IS impossible;

Figure 6. Example of FLORE fuzzy evaluation rules

C. Middleware implementation

The self-organization/configuration middleware is developed over the Apache Felix OSGi framework[13], which implement the OSGi Specification Release 4. Being a SOA framework, OSGi allows sharing services and coding bases between local application modules.

As the proposed middleware is distributed on environment devices, we use Apache CXF dOSGi [14], an OSGi RFC 119 implementation, which enables remote service invocation between OSGi framework through web services. The main benefit of dOSGi is its capability to hide the communication and transport layer, integrating the remote services to the frameworks, allowing the application modules to manipulate the remote services like they was any services. To discover the dOSGi remote services, the middleware uses the Service Location Protocol (SLP), with the jSLP implementation[15], which allows to advertise and discover services on the network. The SLP discovery mechanism is bonded with the OSGi/dOSGi service discovery mechanisms.

To contain and manipulate contextual information, the middleware implementation uses the Jena framework and an OWL environment description. As we want to dissociate the middleware from the OWL semantic framework implementation, we use Java Beans to access and manipulate Jena’s OWL concept instances, hiding the Jena manipulation in the beans, and use OSGi services to manipulate contextual concept instances. For instance, the OSGi JenaOntologyQueryServiceImpl implements the abstract class OntologyQueryService, which allows to access the Jena’s Device Java Beans that are generated from the data contained in the Jena framework. Moreover, the FLORE uses the JENA SPARQL functionalities to retrieve the device and application descriptions from the ontology.

To illustrate the relation existing between the middleware’s components, the Figure 7 presents an example of an application deployment using the self-organization features with a sequence diagram.

Figure 7. Sequence diagram presenting an application deployment which is using the self-organization features.

To manage the software static dependences, the self-organization middleware uses the OSGi Bundle Repository (OBR) application manages the software dependences of the OSGi bundles. Concerning the dynamic dependences, we override the OSGi module activation class and add an additional step before the “bundle” activation by running unit tests. These tests, in fact JUnit tests, evaluate the availability of dynamic resources in the pervasive environment or in the device’s neighborhood. For instance, a JUnit test method could test the presence of a specific Bluetooth device in the neighborhood of a middleware’s device.

IV. RESULTS AND DATA

As we are aiming to reduce the complexity and related processing time for software organization and configuration in smart spaces, we need to evaluate if the proposed solution is computing in a decent interval time and if it gave results that are matching the targeted behaviors.

Concerning the computing time, our first requirement was to get results faster that the time spent for a user to do the task itself. We raise this requirement by aiming middleware’s results in less than the user’s experienced response time[16], around 1 or 2 seconds, allowing simple “click and result” from the smart spaces’ organization interfaces. Concerning the targeted behaviors, we aimed a DCQ repartition that would highly prioritize the less resourcesaturated devices and devices that are in the same zone than software-targeted zone. Therefore, we evaluate and test the proposed implementation of the self-organization middleware and its Fuzzy Logic Organization Reasoning Engine (FLORE) by running benchmarks and tests, and by collecting the resulting data. These tests and benchmarks have been run on a Core 2 Duo 2.4 GHz, 2 Go 667 MHz RAM computer and with the Apple’s Java 1.6 virtual machine. Moreover, we use the JENA framework version 2.5.7 with the file loading and persistence features.

Firstly, we evaluate the computing time for an application deployment using the FLORE. Since the pervasive environments can have a large number of devices, we run the
application deployment test for several groups of devices ranging from 2 to 500 devices. The devices are simulated and their properties and resources have been randomly created. The Table 1 and Figure 8 present the resulting average computing times for the FLORE execution for an application deployment request containing a single application. These computing times are not including the time spent for service invocations, the OBR software dependences management and the dynamic dependences management, which are not include in the FLORE.

<table>
<thead>
<tr>
<th>Number of devices</th>
<th>Computing Times (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0.0122</td>
</tr>
<tr>
<td>10</td>
<td>0.0330</td>
</tr>
<tr>
<td>25</td>
<td>0.0430</td>
</tr>
<tr>
<td>50</td>
<td>0.0676</td>
</tr>
<tr>
<td>75</td>
<td>0.0948</td>
</tr>
<tr>
<td>100</td>
<td>0.1186</td>
</tr>
<tr>
<td>125</td>
<td>0.1662</td>
</tr>
<tr>
<td>250</td>
<td>0.2630</td>
</tr>
<tr>
<td>500</td>
<td>0.3908</td>
</tr>
</tbody>
</table>

This DCQ repartition reflects the designed reasoning rules and membership functions and match with our objectives to give a high DCQ for devices with a low saturation (approx. 0 to 25 %), an intermediate DCQ for an intermediate saturation (approx. 30 to 75 %) and a low DCQ for a high saturation (approx. 80 to 100 %). Of course, the resource saturations for a single device will rarely be the same in non-simulated environments and the zone relation variation between the environment devices will give a different DCQ attribution, without these 3 observed clusters.

These results on the processing speeds and behaviors of the FLORE are the first collected data on the implemented self-organization middleware. We want, in our future work, to evaluate more in depth our middleware implementation and get more measurements and comparative data like conformity face to usage scenarios and reasoning resources utilization.

V. RELATED WORKS

Some work has been done on autonomic application provision and self-configuration/organization in the pervasive environment. One of the most significant is the work of Trumler et al. [17]. In their project, called AMUN, they propose a middleware that facilitates the management and usability of the devices by giving some self-configuration, self-healing and self-optimization mechanisms. They also propose a solution to the self-configuration based on the social behavior of a cooperative group in the context of job assignments[18].

In Gaia project, Ranganathan and Campbell[19] also propose some mechanisms related to the software self-organization. To address the application deployment complexity, they propose a solution based on the STRIPS planning algorithm, where the application deployments are binded with the user goals. Furthermore, Ranganathan proposes a solution where an ontology, describing the context of the smart spaces, is used to find the right device configuration through a semantic reasoning[20].

Ghorbel et al.[21] in their work on the service provision propose solutions on the software self-organization/configuration. They are mainly focusing on the utilization of the semantic reasoning to choose the right end-user service, for a mobile device, according to the profile of
the user using the mobile device and the environment context.

Grid computing is also confronted to the complexity of the software provision or deployment on several devices[22]. Much work has been done on the resource analysis and node selection[23] or the processing load balancing between the computing nodes.

The previous presented related work are, for the majority, considering the benefit of the semantic description and reasoning. Their reasoning techniques are mainly based on the description logic reasoning, where they are comparing semantic data to deduce the organization results. Description logic is powerful when it is time to compare qualitative or semantic values, but have difficulties to evaluate and compared quantitative with qualitative values. By using a fuzzy logic approach, it is possible to have larger range of results, have more result expressivity and easily evaluate quantitative values. More benefits of the fuzzy logic in the self-organization/configuration are presented in the next section.

VI. CONCLUSION

To increase the broad utilization of the pervasive computing and see more smart spaces, it is critical to get solutions that will simplify the implementation, the management and configuration of the software applications in these environments. The Autonomic Computing approach gives hints and directions in the implementation of such solution, by proposing to reduce user’s manipulations through automation and reduce user knowledge and reasoning on the systems by environment reasoning. Through the presented self-organization middleware, it is possible to reduce the amount of needed operations to deploy or manage software in smart spaces. With the addition of the Self-Healing, Self-Optimization and Self-Protection mechanisms to the proposed middleware, it will be possible to get pervasive environments that will practically manage themselves, reducing the management costs and giving environments with a better usability and more user’s, healthcare giver’s and support staff’s scalability.

VII. FUTURE WORKS

The proposed self-organization middleware implementation and its FLORE are the initial effort of a middleware that will reduce the complexity and the number of the user required manipulations to deploy new software applications in pervasive environments. As we present above, we want in our next work, to extend the middleware capabilities by adding the users and their profiles in the organization/configuration reasoning. We will extend the environment description and the FLORE to evaluate the position and the orientation of the users in the pervasive environment face to the device locations and interaction capabilities.

Moreover, by taking in account the user profile properties in the FLORE, the DCQ attribution will be more complex, depending on a larger type of data. A second reasoning algorithm will be required, which will include new fuzzy rules and membership functions that will evaluate the user profile properties.

Once the self-organization/configuration middleware will be upgraded, we plan to use it as the base for the other Autonomic computing ‘self’: the self-optimization, self-healing and self-protection. Thus, the self-optimization will use the self-organization functionalities to re-organize the environment software configuration when new devices, providing more suitable context or resources for environment applications will enter the pervasive environment. We will use the DCQ to evaluate if the new devices are providing better suitability for the deployed software than during the first deployment evaluation. Finally, the self-organization/configuration will be used by the self-healing and self-protection to move or redeployed applications on other systems in case of software faults, hardware failures or compromised devices.

REFERENCES


