A self-configuration middleware for smart spaces

Charles Gouin-Vallerand, Bessam Abdulrazak, Sylvain Giroux
DOMUS Laboratory, Université de Sherbrooke,
{charles.gouin-vallerand,sylvain.giroux,bessam.abdulrazak}@usherbrooke.ca

Mounir Mokhtari
Handicom Laboratory, Telecom SudParis
mounir.mokhtari@it-sudparis.eu

Abstract

The self-configuration process can simplify the complexity and reduce the cost of the management and deployment of devices, applications and services in smart spaces. Mechanisms inspired from the Autonomic Computing and Pervasive Computing can be used to automate management of the heterogeneous space's component. In this paper, we present our work on Autonomic Pervasive Computing and mainly the self-configuration process. We also introduce our self-configuration middleware that is deployed at DOMUS Laboratory.

1. Introduction

The majority of the smart spaces are composed of several devices: Personal Computers, SmartPhones, sensors, embedded devices, etc. According to the ubiquitous computing model, proposed by Weiser [1], the information processes of the environment are distributed among the devices, being closer to the application context. This model is closely related to the smart spaces by the relations between them: several devices that communicate information, dynamic environment where devices are rapidly entering and leaving, etc.

Among the benefits of the ubiquitous computing model, we can highlight a better interaction between users and devices through tangible interfaces or personalized interfaces. Also, the process distribution allows a better fault tolerance and a better dynamicity in the device interactions. However, this model has serious drawbacks as the complexity to implement the applications caused by the heterogeneity of the devices, the different used communication protocols, the important number of devices, etc. This drawback is one of the main reasons that prevent the democratization of the ubiquitous computing and the smart spaces.

The complexity of the smart spaces is similar to the problems of the large enterprises that owned several servers and large applications on them. In some cases, the complexity of these applications and the relations between them are so important that it is impossible for a single human to understand or manage them. To address the complexity problems, IBM, in 2001, started an initiative called the Autonomic Computing [2]. This model aims the creation of solutions that will simplify the management complexity of the computer systems with autonomic mechanisms. This model is based on four concepts called the four 'selfs' :

- self-configuration, automatic configuration of the applications and devices;
- self-healing, automatic detection and correction of the faults;
- self-optimization, proactive mechanism that continually enhance the performance of the systems;

- self-protection, automatic identification and resolution of the security threats.

With the self-configuration, it is possible to deploy large applications automatically by doing a few simple actions. The complexity is solved by intelligent mechanisms based on a set of rules or semantic descriptions of the applications, depending on the used approach. With the self-protection, it is possible to get systems that are more proactive in security matter and are trying to secure themselves. To get computer systems that are less attuned to physical or software faults, the self-healing proposes mechanisms that are detecting faults and trying to correct them. Finally, the self-optimization creates proactive systems that are always searching for opportunities to enhance their performances by, for instance, balancing their processing load between the environment devices. These four concepts can, of course, be used in smart spaces to simplify their management and utilisability.

At DOMUS Laboratory at Université de Sherbrooke [3] and Handicom Laboratory at Telecom SudParis [4], we are steadily confronted to the complexity of the smart spaces. The main goal of these laboratories is the creation of solutions for smart spaces, to help people with special needs (disabilities and aging) to live much longer in an autonomous way and to increase their quality of life. In our context, we are using smart spaces to improve the assistance that is given to inhabitants through specific human environment interfaces (HEI), all sort of physical or software services, web pages, mobile devices, embedded devices, etc. However, the large number of applications and devices, the heterogeneity between them, make difficult to implement and configure a smart space for a specific context. This observation is even more true when there are several spaces to implement. These aspects drove us toward the Autonomic computing and its concept of self-configuration.

Therefore, in this paper, we are presenting our recent works on a middleware that simplifies the management and the deployment of the applications and services in several smart spaces with the help of the self-configuration concept. Firstly, Section 2 presents our vision on the Autonomic Pervasive Computing approach. Section 3 presents our self-configuration middleware for smart spaces. Section 4 presents the ongoing work on the middleware and finally we are concluding this paper in Section 5.

2. The Autonomic Pervasive Computing

Few work has been already done on the integration of the Autonomic Computing to the Pervasive Computing. Trumler in [5] presents its work on the Autonomic Computing versus the Pervasive Computing model and proposes some solutions to the self-configuration and self-optimization. Ranganathan [6][7] proposes also some solutions to the self-configuration and the self-healing by using STRIPS planning, context awareness and fault-tolerance mechanisms. On the other hand, Helal in [8] describes the importance of the self-integration in the smart spaces, a problem similar to the self-configuration.

Thereby, the Autonomic Computing model appears to be the perfect approach to simplify the implementation and the management of the smart spaces. Moreover, the "four selfs" give us good directions to implement the needed tools and mechanisms. Thus, through our needs in matter of smart spaces, we forge our vision of the Autonomic Computing applied to the Pervasive Computing, the Autonomic Pervasive Computing, divided under the four originals concepts [9].

The self-configuration of a smart space mainly concerned the application management and deployment, the configuration of these applications and their integration to the smart space. To

simplify the deployment and the management operations, the smart spaces must be able to receive simple configuration requests from the space administrators, analyze them and autonomously apply the requests on the right applications or devices in the right contexts. For instance, to deploy an assistive cooking application, the administrators could sent a simple request to the smart space that would contain the application download URL, its context (location, users, time, etc) and its needs regarding the targeted device resources. Then, the smart space could find by itself the more suitable devices concerning the request and finally deploy the software. As the self-configuration is our main subject in this paper, we will describe further this aspect in the next section.

In our vision of Autonomic Pervasive Computing, the self-optimization is strongly related to the self-configuration. At each application deployment, the more suitable device, face to the application or service context and needs, is chosen, contributing to an optimized configuration of the smart space. However, the self-optimization can also be accomplished by balancing the application or service processing load between the devices of the smart space and having proactive services that are regularly proposing assistance to the users or to other services.

A self-configured and self-optimized environment is useless if the deployed applications cannot give the right service at the right moment. Thus, the self-healing must propose mechanisms that will enhance the fault tolerance of the space by detecting automatically the occurring faults, forwarding intelligible reports to the administrators and if possible, automatically resolve these faults.

Finally, since the smart spaces are highly dynamic and several users with their mobile devices can go in-an-out, it is important to get mechanisms that will enhance the protection of the smart space integrity by protecting the access to the services and data. Moreover, the smart spaces must be proactive in their protection and constantly try to detect the compromised devices or the malevolent users. They could further isolate automatically the compromised devices by blocking their access to the spaces.

3. Toward an Autonomic Pervasive Computing middleware for smart spaces

To implements all the mechanisms related to the Autonomic Pervasive Computing, we are proposing a middleware based on our previous work [10][11][12]. In these work, we implemented a middleware that reduced the complexity of the management and software deployment in several smart homes. It was aiming the reduction of the complexity by simplifying the operations, reducing the amount of information that is presented to the manager, introducing the SOA model with the Open Service Gateway initiative (OSGi) standard [13] and allowing the remote management of several smart spaces. OSGi is a Java framework standard that is offering SOA capabilities to device local modules and applications and an easy life-cycle management of the deployed modules.

The next step in this middleware corresponds to the self-configuration aspect of the Autonomic Pervasive Computing approach. Thus, we are proposing a new version of the middleware that includes the autonomic mechanisms that reduce, one more step forward, the amount of needed operations to manage or deploy applications in several smart spaces. Therefore, the main goal is to leave the space do the bulk of the work, by selecting themselves the configuration targeted devices, deploying or managing the applications and integrating them to the smart space.

Since the self-configuration is a base for the other aspects of the Autonomic Pervasive Computing, the three other "self's" depend on some mechanisms proposed by the self-configuration. Thus, it is required to implement in first place the mechanisms and services related to this aspect.

3.1 The self-configuration overview

The self-configuration can be divided in three steps: the application deployment and management, the application configuration and the application integration to the smart space. During the application deployment or management step, a request is sent by the administrators to the smart spaces. Then, the more suitable device, face to the application context and its hardware resource needs, is chosen between the devices of the environment and the final request is sent to the selected device.

To choose these more suitable devices for a given context, we will use a Web Ontology Language (OWL) ontology that will describe the context of the spaces and will allow automatic selection of the best suitable devices versus the context of the applications through semantic search and matching. This automatic selection will reduce the amount of operations needed by the administrators and also the amount of data that are sent to them during an application deployment or configuration, notably when they have several smart spaces to manage. This ontology will be populated by the device discovery mechanisms and by the different context sources in the environments such as location systems, sensors, etc.

We are currently working on the environment description that will be implemented into the ontology. However, for development and test purpose, we are using, for the moment, a static Java object description that contains only some information like the device name, its deployed applications/modules and its location in the smart spaces. We will add in the final version more description field such as the device hardware resources, a more detailed location based on the device context in the space, the list of services that are exported by the device applications, etc. The utilization of these fields with the OWL ontology will add more flexibility to the middleware in the selection of the devices and will simplify the requests sent by the administrators. However, we are aware of the performance problems related to the utilization of ontologies. According to our tests made on the implemented part, the delays are reasonable. More detail on the implementation of the ontology and results will be published in a future paper.

In the application configuration step, the deployed application is configured with the device hardware and software resources. Thus, in this step, the services that are needed by the deployed application are searched into the space and gave to the applications. As the middleware is primary build on the OSGi framework to get its SOA capabilities, the majority of the applications in the space are importing or exporting services to other applications. However, OSGi, initially, allows only service exchange between the applications that are deployed on the same framework. To find and use services between devices, several protocol can be used, but the more interesting are Universal Plug and Play (UPnP), a P2P protocol based on SOAP and SSDP; Jini, a Java protocol based on the Remote Method Invocation (RMI) and JuXTApose (JXTA), a Java P2P protocol.

Fortunately, by using API that are converting OSGi services to a specific protocol and vice versa, as the Apache Felix UPnP Driver [14], it is also possible to import or export services to external devices in a transparent way for the local applications. This protocol transparency is very important, as we are aiming to get a middleware that is not particularly bounded to a

particular protocol. It is also at this step that the hardware resources are allocated and the context description is updated into the ontology with the new device state.

The last step concern the integration of the application with the other softwares and devices of the smart space. For an application that is offering some services to the space, it is at this step that the services are exported through a specific protocol driver. The integration step concerns also the discovery and the integration of devices to the middleware. Thus, it is important to have mechanisms that allow the discovery of the devices and integrate them to the environment in a transparent manner regarding the exact implementation of the discovery protocols. Once the discovery made, each device found must be added to the ontology, allowing to the services and applications of the environment, to search for the active device in the smart space.

3.2 The middleware implementation

In accordance with three steps of the self-configuration, we divided our middleware in four types of device or application, each corresponding to their played roles in the smart space :

- the Managing Tool, the GUI application to manage and configure the smart spaces;
- the Environment Gateway Device, the device that is receiving the management requests from the Managing Tool, analyze them and execute them on the devices;
- the Ontology Device, the device that is managing the smart space ontology and offering ontology services to the environment devices;
- the Environment Device, the software part that is installed on each environment devices and offering the services to manage the deployed applications on these devices.

The Figure 1 presents these four device types and the middleware architecture. The arrows represent the service calls and the number (T*) at the start of each label represented the service call order. Also, for the example purpose, only the UPnP Driver is represented in the figure but the architecture can get as much as wanted protocol driver. In the following lines, we will describe further these four types of device, their implementations and their roles in the middleware.

Firstly, the Managing Tool is the application used by the administrators to configure, manage and deploy applications in the smart spaces. This application, implemented under an Eclipse Plugin format, allow administrators or software developers to implements applications in the Eclipse IDE and deploy or update directly the smart space applications in the same working interface. Through different panels, the managers can view the active devices in each smart spaces, view the deployed applications on each devices, view their logs and situate them on an environment map. To send management requests to the smart spaces, the Managing Tool sends Simple Object Access Protocol (SOAP) requests to each Environment Gateway Device of each smart space. It is also through SOAP that information on the context of the smart spaces, such as the location of the devices into the environments, are retrieved.

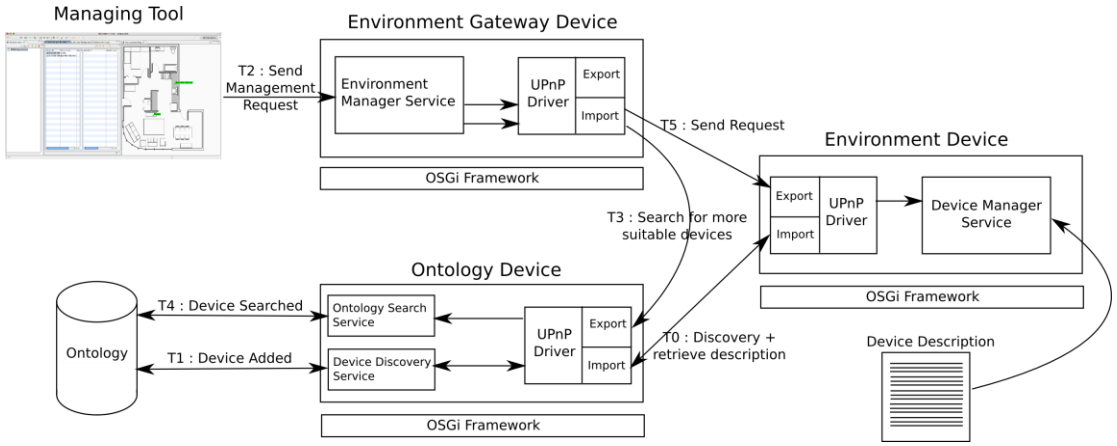


Figure 1 . The Self-Configuration Middleware and its four device types.

The second device type is the Environment Gateway Device. Its role is to receive the management and configuration requests from the Managing Tool, analyze them, find the best device configurations face to the context of the requests and finally send the final requests to the selected devices. More exactly, it is the Environment Manager Service that has the role to analyze the requests by extracting the required contextual information. Then, it ask to the Ontology Search Service, exported by the Ontology Device, to perform a semantic search with the extracted contextual information to find the more suitable devices. Also, once the requests are executed on the devices, the request results are returned to Environment Gateway Device, then to the Managing Tool.

Lastly, the Environment Manager Service selects the top list device and sends to the Device Manager Service of the selected device the final management request. Of course, if no results are returned by the semantic search, an intelligible message is returned to the Managing Tool that made the request.

Moreover, the Environment Gateway Device is also an intermediate between the external world and the smart space. Thus, it is important to get a control access to this entity and its services. We are currently using a Secure Sockets Layer (SSL) encryption and an authentication, through usernames and passwords, for the SOAP Server (Apache Axis) that is receiving the requests from the Managing Tool and sending them to the Environment Manager Service.

The Ontology Device is the entity that is managing and offering the access to the environment ontology. Through its Device Discovery Service, the Ontology Device discovers the active devices in the smart space, retrieve their descriptions and add them to the ontology. To do that, the service uses the different protocol drivers that are deployed on its OSGi framework and listens the events concerning the arrival or retrieval of Device Manager Service in the smart space. When one of these services is discovered, its contextual information is retrieved through a method call on the service and the information is added to the smart space ontology.

Inversely, when one of the Device Manager Service is retrieved from the environment, the Device Discovery Service is informed and it removed the device description from the ontology. When the self-healing mechanisms will be implemented, it will also be possible, in case of device failures, to remove the devices from the ontology.

The second service of the Ontology Device is the Ontology Search Service. This is the service that is used by the Environment Gateway Device to search for the more suitable devices for the context of the received requests. Thus, this service receives the search requests, submits them to the ontology and returns the results to the service callers. In that case, the major part of the service job is made by the OWL ontology framework, such as the Jena framework [15], a Java open source semantic Web framework.

The last device type, the Environment Device, correspond to the software part that is deployed, when it is possible, on each devices of the smart space. Through the protocol drivers, the Device Manager Service is exported to the smart space. Then, the Device Discovery Service of the Ontology Device discovers each instance of the Device Manager Service and retrieves the device descriptions through the service instances. The device descriptions are written in a file in each device and they are accessible through a method call on the Device Manager Service. Lastly, the device description is added to the environment ontology. It is also through the Device Manager Services that the management and configuration requests are sent to the device OSGi framework.

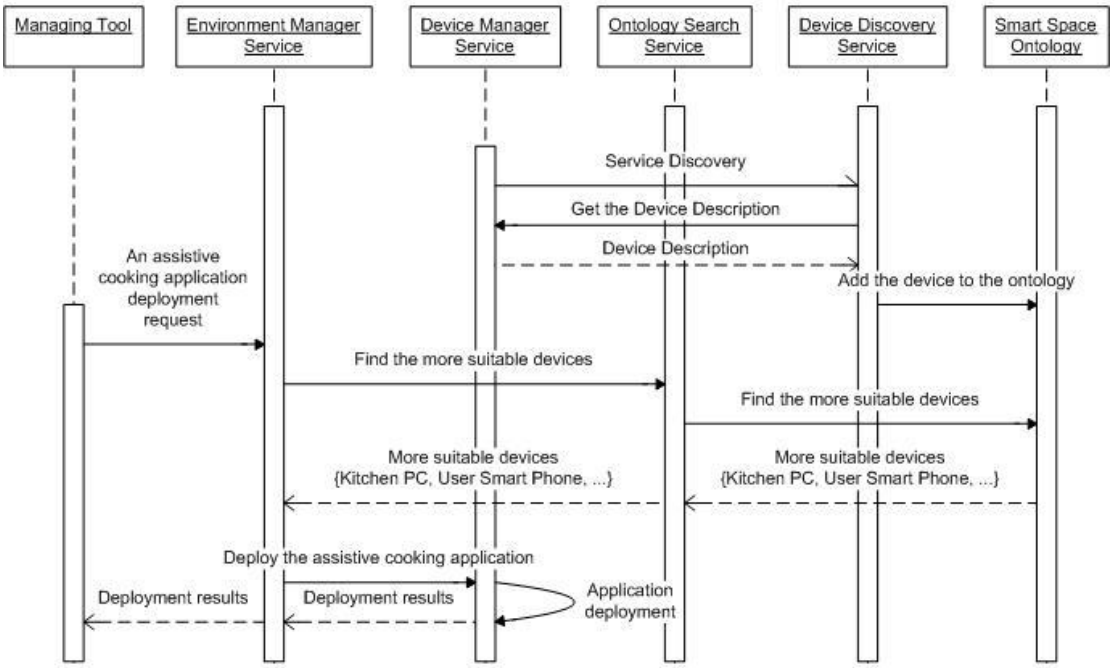


Figure 2 . An example of an application deployment with the self-configuration middleware.

As the majority of the devices in a smart space are heterogeneous, the hardware resources and specifications can vary greatly. For example, a smart space can contain some capable computers that can run practically every type of application, such as personal computer, but can also contain small electronic devices, such as embedded devices, that can run only small and dedicated applications. Thus, it is possible that some devices cannot run the OSGi framework and/or the Environment Device services and software libraries. Moreover, some applications or services can be strongly related to some devices, such as a fridge manager application and a refrigerator. However, these devices cannot have the sufficient resources to host the services. In

these two cases, a surrogate solution can be used, by hosting the Environment Device services and libraries on other devices of the smart space. Thereby, the hosting devices are acting as a proxy for the smaller devices.

Concerning the three device types: the Environment Gateway Device, the Ontology Device and the Environment Device, it is important to note that it is possible to assemble them on a same device without a problem, thanks to the used SOA model and to the protocol drivers. For example, it could be possible to merge the Ontology Device with one of the Environment Device or with the Environment Gateway Device. Moreover, as we said before, the Environment Device services must be installed on each device of the smart space. Thus, the Ontology Device and the Environment Gateway Device have also these services that enable the management of their applications.

To resume the middleware, the Figure 2 presents an example of a deployment request sent by the Managing Tool to a smart space. This figure is presented in an Unified Modeling Language (UML) sequence diagram, the middleware services are represented by the vertical components and the service method calls are represented by the arrows exchanged between the vertical components.

4. Ongoing Works

As specified before, we are currently working to improve the self-configuration middleware, by improving the services related to the ontology. Also, we are working to replace the current Java-object environment description by an OWL description. This environment description will be inspired by the concepts used in the Archipel project [3], a framework that brings assistance to users in the realization of tasks in a smart space context. However, the Archipel environment description is not using OWL but rather uses XML and Java objects. Thus, we are converting the Archipel concepts to OWL and adding the other concepts that are missing such as the device resource description or the application/service context description. The OWL environment description will offer the needed semantic search and matching tools that are currently missing in the middleware.

Also, to enhance the middleware and to bring it more versatile, we are implementing an OSGi-JXTA Driver that will be similar to the UPnP Driver and will export the OSGi services to the JXTA protocol and, vice-versa, will import the JXTA services that are exported by the other devices, to OSGi. This new driver will allow to cover more device type. Also, JXTA will allow to manage mobile devices and their applications when they will leave the smart space, thank to the JXTA relay node concept, that is dealing with the Network Address Translation (NAT) problem in subnetwork such in smart space.

Also, to improve the self-integration of new devices to the smart space, we are planning to add some mechanisms in line with the SODA and Stepstone projects [16]. These two projects are addressing the self-integration problems in the healthcare context and proposing some interesting SOA solutions, based on the OSGi framework.

5. Conclusion

In this paper, we presented our first step toward an Autonomic Pervasive Computing solution. Through the presented self-configuration middleware, we simplify the management and configuration of smart spaces by reducing the amount of needed operations to deploy or manage devices, applications and services. In our middleware, the smart space participates

actively in the application configuration by doing the major part of the tasks by selecting the right devices, face to the software contexts, deploying or managing them and integrating them to the environment.

In our future works, we will focus on the three other aspects of the Autonomic Pervasive Computing: the self-healing, self-optimization and self-protection, and will integrate them to the presented middleware. Each of these aspects will offer some interesting challenges, such as the difficulties to detect the device failures or the software faults in the self-healing part or to automatically detect the compromised devices in the self-protection part. At the end, it will be possible to get smart spaces that will practically manage themselves, leaving more time to managers and programmers to really improve the smart spaces, and by the way the smart space user experiences.

6. References

[1] M. Weiser, “The computer for the 21st century”, Scientific American, pages 94–104, 1991.

[2] J. O. Kephart and D. M. Chess, “The vision of autonomic computing”, IEEE Computer, 36(1):41–50, Jan 2003.

[3] S. Giroux, J. Bauchet, H. Pigot, D. Lussier-Desrochers, and Y. Lachappelle, “Pervasive behavior tracking for cognitive assistance”, International Conference on Pervasive Technologies Related to Assistive Environments, PETRA 2008, July 2008.

[4] M. Ghorbel, M. Mokhtari, and S. Renouard. “A distributed approach for assistive service provision in pervasive environment”, In Proceedings of the 4th international workshop onWireless mobile applications and services on WLAN hotspots, WMASH ’06, page 95, 2006.

[5] W. Trumler, R. Klaus, and T. Ungerer, “Self-configuration via cooperative social behavior”, In L. T. Yang, H. Jin, J. Ma, and T. Ungerer, editors, ATC, volume 4158 of Lecture Notes in Computer Science, pages 90–99. Springer, 2006.

[6] A. Ranganathan and R. Campbell, “Autonomic pervasive computing based on planning”, International Conference on Autonomic Computing, pages 80–87, May 2004.

[7] A. Ranganathan, C. Shankar, and R. Campbell, “Application polymorphism for autonomic ubiquitous computing”, Multiagent Grid Syst., 1(2):109–129, 2005.

[8] S. Helal, “Programming pervasive spaces”, IEEE Pervasive Computing, 4(1):84–87, 2005.

[9] C. Gouin-Vallerand, B. Abdulrazak, S. Giroux, and M. Mokhtari, “Toward Autonomous Pervasive Computing”, International Conference on Information Integration and Web-based Applications & Services, iiWAS 2008, Nov. 2008.

[10] C. Gouin-Vallerand and S. Giroux, “Managing and deployment of applications with OSGi in the context of smart home”, Third IEEE International Conference on Wireless and Mobile Computing, Networking and Communications, WiMOB 2007, Oct. 2007.

[11] S. Giroux, M. Castebrunet, C. Gouin-Vallerand, and B. Abdulrazak, “A pervasive framework for multi-agent personalization in smart homes”, 2nd International Workshop on Distributed Agent-based Retrieval Tools, DART 2008, Sept. 2008.

[12] B. Abdulrazak and A. Helal, “Enabling a plug-and-play integration of smart environments”, Information and Communication Technologies, 2006. ICTTA ’06. 2nd, 1:820–825, April 2006.

[13] The OSGi alliance website. http://www.osgi.org/.

[14] The Apache Felix UPnP driver website, http://felix.apache.org/site/upnp-driver-architecture.html.

[15] Jena - a semantic web framework for java - website. http://jena.sourceforge.net/.

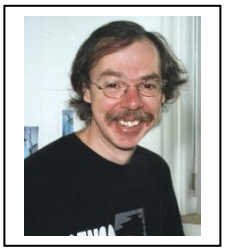
[16] S. de Deugd, R. Carroll, K. Kelly, B. Millett, and J. Ricker, “SODA: Service oriented device architecture”, IEEE Pervasive Computing, 5(3):94–96, c3, 2006.

Authors

Charles GOUIN-VALLERAND, MSC, is a Ph.D. student in computer science at the University of Sherbrooke, Canada, and at Institut Telecom

(ex GET/INT)/University of Pierre and Marie Curie, France. He received his M.Sc. in computer science from University of Sherbrooke, in 2008, and B.Sc. in computer science from University of Sherbrooke, in 2005. His main research interests are pervasive computing, distributed computing, smart spaces, context awareness, middleware and assistive technologies.

Bessam ABDULRAZAK, PHD, is an Assistant Professor of Computer Science at the University of Sherbrooke, Canada. Previously, he was a Post-Doc at the University of Florida (US), and before that at Telecom-SudParis (France). He received his Ph. D. in computer science from Telecom-SudParis, M.S. of Robotics from Paris VI (France), and B.Sc./ Ing, from USTHB (Algeria). His research interests include ubiquitous and pervasive computing, ambient Intelligence, smart spaces, assistive technologies and rehabilitation robotics.



Sylvain GIROUX, PHD, is a Professor of Computer Science at the University of Sherbrooke, Canada. He received his Ph. D. in computer science from the University of Montreal, in 1993, and M.Sc. in mathematics from University of Sherbrooke in 1990. His main research interests are mobile computing, pervasive computing, distributed artificial intelligence, multi-agent systems, user modelling and intelligent tutoring systems. He is one of the heads/founders of the DOMUS (Domotics and Mobile computing) laboratory, University of Sherbrooke.

Mounir MOKHTARI, PHD, is an Associate Professor at Institut TELECOM (ex GET/INT) in France. He received his Ph. D. from University Pierre and Marie Curie (France)/INSERM laboratory, in 1997, Masters from University Val de Marne (Paris) France in 1992 and Master of Research in networking from INT in 1994. His research activity focuses mainly on Human-Machine interaction, rehabilitation robotics and health telematics. Dr. Mokhtari, who is leading several European and national projects, is the head and founder of Handicom Lab (Handicap Engineering and Communication Lab, founded in 1999).