PECES Middleware Challenges – On Building the Bridge between Islands of Integration[A]

Marcus HANDTE[1], Muhammad HAROON[1], Wolfgang APOLINARSKI[1],
Pedro MARRON[1], Vinny REYNOLDS[2], Danh LE PHUOC[2], Manfred HAUSWIRTH[2]
[1]*University of Bonn and Fraunhofer IAIS, Germany*
Email: {handte, haroon, apolinar, pjmarron }@cs.uni-bonn.de
[2]*National University of Ireland, Galway, Ireland*
Email: {vinny.reynolds, danh.lephuoc, manfred.hauswirth}@deri.org

Abstract: Pervasive computing envisions seamless support for user tasks by means of cooperating networked devices that are invisibly integrated into the environment. Due to device integration and device mobility, the resulting networks can be highly heterogeneous and dynamic which complicates application development. In the past, researchers have designed several middleware systems to deal with the arising challenges. Usually these middleware systems are based on the concept of smart spaces. Thus, they enable the collaboration of devices in the vicinity of the user. However, they provide only little support for the collaboration of devices that are located in distant smart spaces. The goal of the PECES research project is to bridge the resulting islands of integration. As we report in this paper, the goal set out by PECES results in an extended set of middleware challenges.

1. Introduction

The ongoing miniaturization of computer technology and the proliferation of wireless networking technology have led to an ever-increasing number of networked computer devices that are invisibly embedded into all kinds of everyday objects. In the recent past, researchers have developed a number of middleware systems to facilitate the development of so-called smart spaces which integrate the devices present in a certain area. To keep smart spaces manageable, the middleware systems introduce artificial boundaries that are based on physical proximity [1], [2], [4] or geographical location [3], [6], [7], [8] such as rooms or buildings. Besides providing support for a single smart space, some middleware systems also offer mechanisms to migrate the application state created in one space to another [10] or adapt the set of applications leveraged by a user depending on the capabilities of the present space [11].

Although the existing middleware systems can be used to realize important parts of Marc Weiser's vision of pervasive computing [11], so far they only provide support for scenarios dealing with one smart space at a time. As a consequence, the existing solutions lead to islands of integration that do not interact with each other. However, many future applications will require the *simultaneous interaction of devices that are integrated in different smart spaces* in order to provide a truly seamless user experience. The spectrum ranges from applications that provide *proactive support for nomadic users* to applications that provide *support for remote collaboration of users*. Naïve solutions like increasing the

[A] This work has been partially supported by CONET (Cooperating Objects Network of Excellence) and PECES (PErvasive Computing in Embedded Systems), both funded by the European Commission under FP7 with contract numbers FP7-2007-2-224053 and FP7-224342-ICT-2007-2 respectively.

area that is covered by a particular smart space are usually not applicable, e.g. due to scalability issues or due to the fact that the smart space represents given administrative domains that are not subject to extension. Moreover, in many cases they are not desirable as potentially sensitive context information such as the physical location of objects or the presence of users is often shared freely between the devices within a smart space. Thus, there is a need for supplementary concepts and supportive middleware mechanisms that enable the networking of embedded devices across the boundaries of a single smart space. The design and development of such concepts and mechanisms is a key goal of the PECES research project. In this paper, we report on the challenges arising from this goal and we discuss how they impact the architectural design of pervasive computing middleware.

The remainder of this paper is structured as follows. In the next section, we provide more details on the objectives of the PECES research project and we introduce some motivating application scenarios. In Section 3, we describe the resulting challenges with respect to pervasive computing middleware. In Section 4, we outline the components of the middleware developed in the PECES project and we conclude the paper with a short summary and a discussion of future work in Section 5.

2. Objectives and Scenarios

As indicated previously, the primary goal of the PECES research project is the development of concepts to support the seamless interaction of networked embedded devices across the boundaries of a single smart space. As depicted in Figure 1, this goal fosters an extension of the typical system boundaries of a smart space which enables applications to dynamically extend their boundaries to include remotely located devices and services. To achieve this goal, the project consortium designs an integrated software infrastructure that consists of a set of context ontologies, a middleware system and an associated set of application development tools. To validate the concepts realized by this software infrastructure, the project consortium develops prototype applications for a broad range of scenarios.

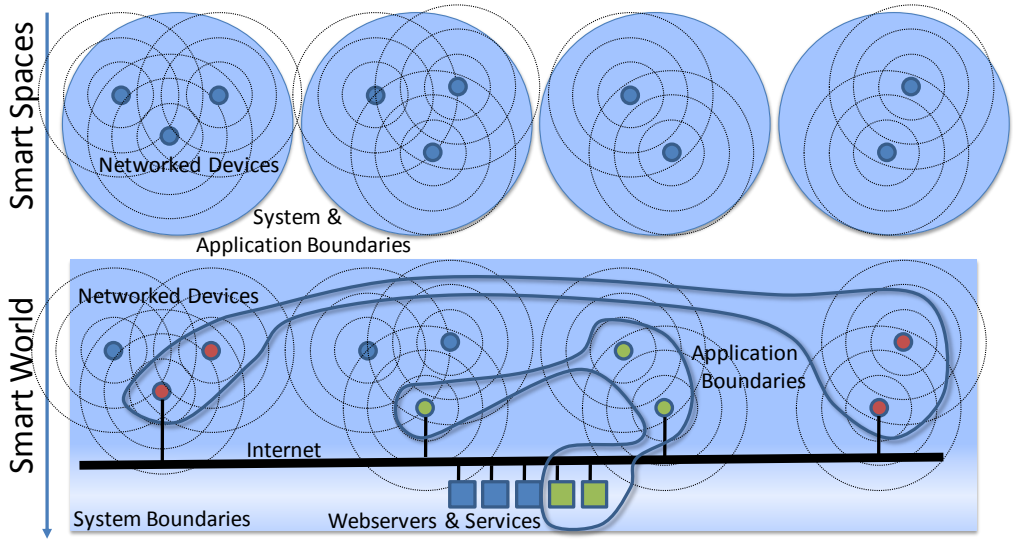

Figure 1 – PECES Objectives

The first application scenario is targeted at integrated trip assistance that provides *proactive support for nomadic users*. Imagine a salesman that shall participate in a meeting with a customer. When entering his car, his smart phone joins the smart space formed by the devices integrated into the car. Since the salesman wants to drive to the customer, his smart phone programs the in-car navigation system appropriately. By validating the route, the navigation system detects that the salesman must travel via a toll road and since the

final destination lies within a city center, the salesman must use an underground parking lot. After asking for confirmation, the navigation system uses the smart phone to contact the traffic management system of the toll road to pay for the passage. Furthermore, it reserves a parking at the underground parking lot. When arriving at the toll road, a camera system reads the license plate of the car and since the salesman has already paid for the passage, he may continue the trip to his destination without distraction. When arriving at the underground parking garage, the system that is managing the garage directs him to the reserved parking. At the site of the customer, the person at the front desk authorizes him to access guest services of the smart space on the site. Thus, his devices join the smart space and they use the services to find the meeting room where the customer is already waiting.

The second application scenario is targeted at collaborative medical alert handling that relies on *support for remote collaboration of users*. Imagine an elderly man living alone at home. To improve his medical safety without moving to a special home for elderly persons, he wears a number of sensors that continuously monitor his personal condition. Such sensors may include accelerometers that can detect his movement or whether he has fallen over. Furthermore, they may include sensors to gather blood pressure values or insulin levels. Using wireless networking technology, the sensors are integrated into a smart space that is managed by an inexpensive WLAN router with Internet access. When the sensors detect a potential medical emergency, they report this to the router. Depending on the severity of the emergency, the router determines an appropriate response. In severe cases, the router may immediately contact a remote medical service. Thereby, it may transfer the measurements so that the ambulance can prepare their equipment. In less severe cases, the router may also contact a collaborating neighbour that has agreed to look after the elderly. To ensure that the neighbour handles the alert, the router first contacts the smart space of the neighbour. If the smart space agrees, the alert is forwarded to one of the devices in the vicinity of the neighbour. If the neighbour is unavailable or cannot handle the alert, the router can either contact another neighbour or, as a last resort, it may contact the medical service. To minimize the effort for dealing with false alerts, both – the medical service and the neighbour – may first contact the elderly through his devices before visiting him.

The previous scenarios clearly motivate the need for interaction between devices that are part of different smart spaces. Besides from dynamically joining the appropriate smart space, the devices of the salesman in the trip assistance scenario must also collaborate to contact other devices in remote smart spaces such as the traffic management system and the parking system. Similarly, the devices in the collaborative medical alert handling scenario require the collaboration of the devices of the elderly person with remotely located devices of neighbours or a medical service. It is worth mentioning that the collaboration between the devices in distant smart spaces is fairly dynamic and cannot be predefined statically, in general. Instead, the devices in one smart space must determine the appropriate devices in other smart spaces on the basis of the user's context. For trip assistance, the navigation system selects the traffic management system and the parking system on the basis of the route. In medical alert handling, the appropriate person to contact is selected on the basis of the severity of the emergency and the suitability and availability for handling the alert. Last but not least, it is important to note that the communication between remotely located devices usually relies on insecure networks like the Internet and that the data that is transmitted via these networks may represent sensitive information such as credit card information for trip assistance and medical information in the alert handling scenario.

3. Challenges and Implications

As a first step towards developing the concepts for the PECES middleware, we analyzed the application prototypes to identify the resulting requirements. In the following, we report on the results of this analysis by presenting the key challenges that must be tackled by

middleware that supports the networking of different smart spaces. Since support for smart spaces is a necessary prerequisite for their integration, the challenges can be classified into a set of traditional challenges and a set of extended challenges. The traditional challenges are immediately resulting from the idea of pervasive computing and smart spaces whereas the extended challenges are arising from the goal of networking smart spaces.

3.1. Traditional Challenges

As discussed previously, the first set of challenges represents the traditional challenges that all middleware developers must consider when building support for pervasive computing applications – even when they solely target support for a single smart space. The challenges can be summarized as suitable *support for heterogeneity* and adequate *support for dynamics*. Both challenges can be motivated easily by looking at the typical characteristics of smart spaces envisioned by most pervasive computing scenarios – including the ones presented in the previous section.

Due to the invisible integration into everyday objects, the devices in a smart space may encompass not only powerful general purpose computers such as laptops, desktops and servers but also resource-poor specialized devices. The trip assistant scenario, for example, relies on resource-poor mobile devices such as the smart phone of the salesman as well as embedded devices such as the navigation system of the car. In addition to such devices, the alert handling application introduces devices with more severe restrictions such as WLAN routers and wearable sensors. Due to energy constraints and specialization, devices may be equipped with different short-range communication technologies such as Bluetooth, WLAN, IrDA or Zigbee. Thereby, it is noteworthy that although some devices may be equipped with multiple technologies, not all devices will provide support for all technologies. The smart phone of the salesman, for example, could be equipped with WLAN and Bluetooth. However, the wearable sensors in the alert handling scenario will most likely not rely on WLAN but they will use a communication technology with low energy consumption such as Zigbee. Thus, the integration and specialization of devices lead to highly heterogeneous networks of devices. As a consequence, a suitable middleware for pervasive computing must be minimal to support resource-poor devices. However, in order to be able to leverage the resources of resource-rich devices, it must also be extensible so that it can be tailored to the features provided by the device.

Besides from heterogeneity, middleware that targets support for smart spaces must also handle a potentially high degree of dynamics. Due to the miniaturization of devices, a user may carry multiple devices at a time. In the previous scenarios, the salesman is carrying a smart phone and the elderly person is equipped with wearable sensors. When a user enters a smart space, the mobile devices of the user must be integrated dynamically. Similarly, when a user leaves, the unavailability of the devices must be handled. At the lowest layer, this requires some form of device discovery that takes care of detecting the presence and the absence of a device. On top of discovery, the dynamic integration of devices also requires support for spontaneous interaction between them. Considering that not all devices may be able to communicate with each other directly, e.g. because they may not be equipped with the same communication technology, spontaneous interaction also requires some form of technology-overarching mediation.

3.2. Extended Challenges

The second set of challenges can be classified as extended challenges that result from the goal of supporting interaction of devices across the boundaries of a single smart space. The challenges can be summarized as support for scalable *context-based interaction* and flexible support for secure *context-based access control*. Again, both challenges can be easily

motivated through the trip assistant scenario and the collaborative medical alert handling scenario introduced in the previous section.

As indicated previously, the interaction between devices residing in different smart spaces is often dynamic and thus, it cannot be predefined in general. In the scenarios introduced above, the navigation system and the WLAN router rely on the current context of the user to determine the set of smart spaces to interact with. The traffic management system and the parking garage are selected on the basis of the destination and route of the salesman. Similarly, the smart space of the neighbor is selected dynamically on the basis of the medical data and his availability for handling an alert. As soon as the appropriate smart spaces are identified, the devices interact with each other and they may use each other's services. Thereby, it is important to note that not all devices may be equipped with direct Internet access. Clearly, the traffic management system and the management system for the parking garage may directly offer services via the Internet. However, in the alert handling application, it may be necessary to contact the neighbor through one of the devices embedded into the home environment such as a display attached to a refrigerator, for example. Thus, there may be a need for mediated interaction in such scenarios.

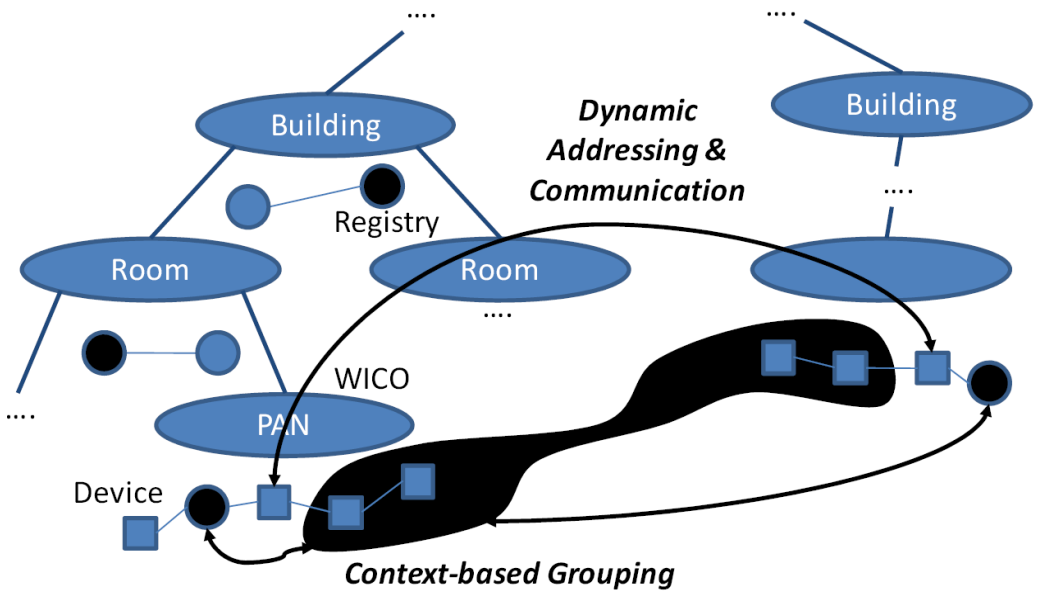

Figure 2 – Context-based Addressing and Grouping

Enabling this kind of interaction requires a suitable addressing and grouping scheme that allows devices to determine their interaction partners in other smart spaces on the basis of their context as shown in Figure 2. Since the interaction partners may be located in physically distant smart spaces the scheme must hide the dynamics of the underlying networks to facilitate scalability. As the scheme necessarily utilizes context information to enable the context-based selection of a device or service in a distant smart space, there is a need for a common representation and understanding of the relevant parts of the context. To establish such a representation and understanding, one may either rely on a comprehensive context ontology or a set of scenario-specific context ontologies that can be switched on demand and that facilitate meshing to support more complex applications. On the basis of these ontologies, the devices may then export parts of their context to different sets of devices. At the lowest level, they may export the available services together with their context to the devices that are present in the surrounding smart space. Beyond their smart space, they may export it to close-by and distant smart spaces. This requires an adequate distributed registry that enables devices to find the desired interaction partners and services. Once the devices and services are identified, they can start to interact with each other. Since

some devices may not be able to access the Internet directly, the interaction requires a gateway concept for mediation. In cases where legacy devices are present, the gateway concept must be flexible enough to support them as well.

Besides from solely supporting context-based interaction, the networking of different smart spaces also raises an extended set of challenges with respect to security and privacy. Within a single smart space, it is often possible to restrict the access to the smart space by means of low-level techniques such as link-layer encryption with a shared secret or access control, e.g. via 802.1x. Such techniques ensure that only authorized devices may join the smart space and thus, it is possible to freely share the services and the context information with the other devices in the smart space. Obviously, this approach cannot be applied when context and services shall be shared between different smart spaces. Thus, it is necessary to control the access at the middleware level. Thereby, it is important to note that the access control cannot be static but it must be based on context as well. This can be clarified by taking a closer look at the alert handling scenario. While it may be perfectly viable that the elderly person is able to determine the availability of a neighbor during alert handling, free access to this information would clearly raise a privacy concern.

To enable context-based access control at the middleware level, it is possible to reuse the context ontologies that are also used for context-based addressing. However, in addition, it is necessary to equip the middleware with a flexible but lightweight mechanism for specifying and enforcing access rights to services and context information. In order to keep the middleware minimal, it is desirable to reuse the addressing and grouping scheme to specify privileged devices and groups. Yet, as the context information may be distributed via insecure networks, it is necessary to ensure that the information used during access control is authentic. This requires additional mechanisms and protocols to ensure that context information cannot be forged easily. Furthermore, in cases where the context information itself is a sensitive piece of information it is necessary to ensure secrecy during the transmission. This requires adequate key distribution mechanisms and encryption protocols. However, since the interacting devices may be resource-poor, these mechanisms and protocols must be lightweight. Alternatively, it may be possible to rely on a gateway concept to offload the resource intensive tasks to a more powerful device. To do this, it is necessary to model and enforce (simple) trust relationships between the devices.

4. Middleware Architecture

In the following, we briefly outline the resulting middleware architecture and the software stack that will be developed within the PECES project. The components of the architecture are depicted in Figure 3. As indicated in this figure, the overall system can be split into different layers. At the bottom, the hardware layer represents heterogeneous hardware ranging from specialized embedded systems to general purpose computers. As discussed previously, these systems will be equipped with different communication technologies. Above the device layer, a hardware abstraction layer takes care of abstracting from the different device capabilities and communication technologies. On top of the hardware abstraction layer, the PECES middleware layer provides common functionality to the application objects and services that reside at the application layer.

As indicated by the colors of the building blocks, the PECES project will not develop the necessary software completely from scratch in order to focus on the new and innovative middleware features. Specifically, the project will reuse software systems that have been developed by third parties as well as existing research prototypes that have been developed by members of the consortium. A Java Virtual Machine (JVM) will be employed at the hardware abstraction layer to abstract from some of the specifics of the underlying hardware. This will enable the utilization of the same software on a broad range of different devices, including personal and industrial PCs, smart phones, PDAs, wireless access points

and embedded systems such as the JStamp or TINI processor as well as embedded sensor platforms such as SunSPOT, for example. To abstract from the networking technology, the project will rely on the BASE micro-broker [2]. By defining a flexible plug-in architecture, BASE can effectively provide a homogeneous but efficient communication API to higher layers that can be used to support different communication abstractions. However, due to the great variety of devices utilized by the targeted applications, it will be necessary to extend the existing system with additional communication plug-ins.

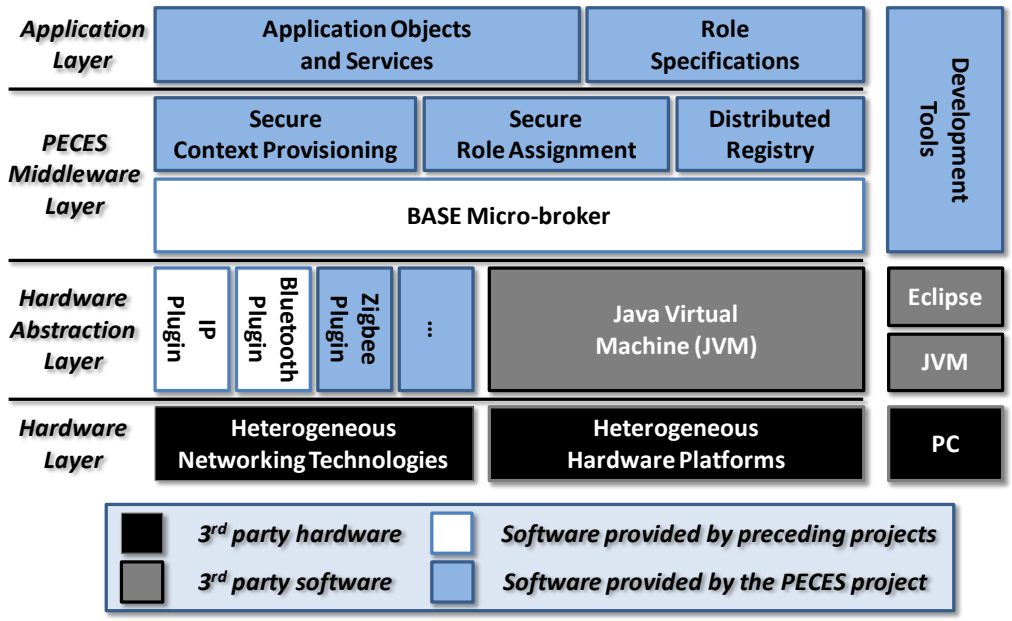

Figure 3 – PECES Middleware Architecture

On top of the abstractions provided by the BASE micro-broker, the PECES consortium will develop three additional middleware services that realize the new core functionality – that is the establishment of smart spaces as well as the dynamic networking of distant smart spaces in an application-specific and secure manner. A service for context provisioning will manage the available context information for the devices. In order to be able to use the context information as basis for security or privacy sensitive applications, the service for context provisioning will not only manage the actual context but also relevant meta information such as the source, precision and timeliness of the information. To model the context information, the PECES consortium will employ a set of flexible and extensible context ontologies that will be instanciated for the scenarios described previously.

To ease the utilization of the context information at the application-level, the PECES middleware will encompass a role assignment service. As described in greater detail in [5], this service will enable application developers to automate various configuration tasks in a context-dependent manner. This includes the formation of smart spaces as well as context-dependent interaction and access control. To support this in a generic manner, the application developer provides a role specification that is automatically (re-)evaluated by the service at runtime. Technically, a role specification consists of a set of roles that are associated with a set of constraints over the context. Using constraints, a developer may, for example, specify that a certain role should be assigned to all devices in the vicinity that belong to a certain user or that are within a certain room. When passing a specification to the service, the service automatically computes the set of devices that match the specified constraints and it assigns the corresponding roles to them. The dynamically computed role assignments can then be used as basis for context-dependent communication. For example, a device may simply send a message to all devices that have been assigned a certain role or

set of roles. Depending on the security requirements, the role assignment itself can be secured appropriately, e.g. by cryptographically signing the association between the role and the target device. Such a secure role assignment enables the usage of an assigned, signed role as cryptographic token during access control.

In addition, the role assignment can also be exported to a distributed registry. This enables the remote discovery and reuse of the assignment. To support this, a role specification may reference role assignments stored in the registry as part of a constraint. By assigning a new role to roles that represent multiple smart spaces, a developer can easily "join" two or more remote smart spaces by assigning a new role to them. The assigned roles can then be used for context-based communication, for example. In order to manage the role assignments in a scalable manner, the distributed registry will be organized hierarchically with registries at various levels, i.e. device registry, smart space registry and global meta-registry.

5. Conclusions

Existing middleware systems for pervasive computing only provide limited support for the simultaneous interaction of devices across the boundaries of a smart space. However, a broad range of pervasive computing scenarios cannot be supported without such an interaction. Enabling this raises a new class of middleware challenges. The goal of the PECES project is to address these challenges with appropriate concepts. Towards this end, the PECES consortium is developing a middleware that enables the interaction across the boundaries of a single smart space in a context-dependent and secure manner. This middleware will provide three novel services to ease application development, namely a service for secure context provisioning, a service for automatic and secure role assignment as well as a distributed registry to manage dynamic role assignments. Together with the remaining middleware components, these services will ensure that future pervasive applications can leverage a dynamic set of smart spaces to bridge the islands of integration.

References

[1] Aitenbichler, E., Kangasharju, J., & Mühlhäuser, M. (2005). Experiences with MundoCore. 3rd IEEE International Conference on Pervasive Computing and Communications Workshops, (pp. 168-172). USA.

[2] Becker, C., Schiele, G., Gubbels, H., & Rothermel, K. (2003). BASE - A Micro-broker-based Middleware for Pervasive Computing. 1st IEEE International Conference on Pervasive Computing and Communications, (pp. 443-451). Fort Worth, USA.

[3] Garlan, D., Siewiorek, D., Smailagic, A., & Steenkiste, P. (April-June 2002). Toward Distraction-Free Pervasive Computing. IEEE Pervasive Computing , 1 (2), pp. 22-31.

[4] Grimm, R. (July-September 2004). One.world: Experiences with a Pervasive Computing Architecture. IEEE Pervasive Computing , 3 (3), pp. 22-30.

[5] Haroon, M., Handte, M., Marron, P. (March 2009) Generic Role Assignment – A Uniform Abstraction for Configuration of Pervasive Systems. Workshop on Middleware Support for Pervasive Computing at Percom'09, Galveston, Texas, USA

[6] Paluska, J., Pham, H., Saif, U., Chau, G., & Ward, S. (March 2008). Structured Decomposition of Adaptive Applications. 6th Annual IEEE International Conference on Pervasive Computing and Communications , S. 1-10.

[7] Ponnekanti, S., Johanson, B., Kiciman, E., & Fox, A. (March 2003). Portability, Extensibility and Robustness in iROS. 1st IEEE Int. Conference on Pervasive Computing and Communications, pp. 11-19.

[8] Roman, M., & Campbell, R. (September 2000). Gaia: Enabling Active Spaces. 9th ACM SIGOPS European Workshop , S. 229-234.

[9] Roman, M., Ho, H. & Campbell, R. (December 2002). Application Mobility in Active Spaces. 1st International Conference on Mobile and Ubiquitous Multimedia, Oulu, Finland

[10] Sousa, J. P. & Garlan, D. (August 2002). Aura: an Architectural Framework for User Mobility in Ubiquitous Computing Environments. 3rd IEEE/IFIP Conference on Software Architecture, pp. 29-43.

[11] Weiser, M. (February 1991). The computer for the 21st century. Scientific American , 265 (3), S. 66-75.