Evaluation of AAL Platforms According to Architecture-Based Quality Attributes

Pablo Oliveira Antonino[1], Daniel Schneider[1], Cristian Hofmann[2], and Elisa Yumi Nakagawa[3]

[1] Fraunhofer IESE, Fraunhofer-Platz 1, 67663 Kaiserslautern, Germany
{pablo.antonino,daniel.schneider}@iese.fraunhofer.de
[2] Fraunhofer IGD, Fraunhoferstr. 5, 64283 Darmstadt, Germany
cristian.hofmann@igd.fraunhofer.de
[3] Dept. of Computer Systems, University of São Paulo, São Carlos, SP, Brazil
elisa@icmc.usp.br

Abstract. In the Ambient Assisted Living (AAL) domain, specific systems have been developed and applied to enable people with specific needs, such as elderly or disabled people, to live longer independently in their familiar residential environments. In order to support the development of such systems, a range of AAL platforms have been developed in recent years. However, there are considerable differences among these AAL platforms, particularly with respect to the treatment of important non-functional properties. This makes the selection of a suitable platform for a given AAL project very difficult. In order to support developers in this difficult task, we present an evaluation of relevant AAL platforms based on a selection of quality attributes that are important for AAL systems.

Keywords: Ambient Assisted Living, AAL Platform, System Architecture, Quality Attribute, AAL Platform Evaluation.

1 Introduction

Driven by demographical and societal changes in most industrialized countries, the development of Ambient Assisted Living (AAL) systems has emerged as a very promising application domain of Ambient Intelligence (AmI) systems. The main focus of AAL systems is to enable people with specific needs, e.g. elderly or disabled people, to live longer independently in their familiar residential environments [1]. From a technical perspective, an essential characteristic of AAL systems is their capability to react adaptively to dynamic changes in device/service availability, resource availability, system environment, or user requirements. Moreover, it is very important for the acceptance of such systems that they are able to assure both,functional and non-functional properties at any time. In order to tackle these challenges, it is indispensable to develop suitable AAL platforms that explicitly address openness, interoperability, adaptivity, and quality assurance. Recognizing these challenges, efforts have been aligned in national and European research projects in order to drive research into this direction [1,2]. As a result, a range of AAL platforms such as UniversAAL [13], OASIS [8], and OpenAAL [9] have been developed. However, each of these platforms has different foci and, correspondingly, different characteristics. In particular, there are considerable differences with respect to the treatment of relevant non-functional properties, such as security, maintainability, and safety. This obviously makes the selection of an adequate platform for a given AAL project very difficult. Even though the importance of non-functional properties has been widely discussed in the AAL research community (for instance, at the MonAMI project workshop in Passau, Germany [10]), there is no comprehensive survey on AAL platforms and their support of important non-functional properties.

Therefore, the main objective of this paper is to provide an evaluation of the currently most widely established and well-known AAL platforms with respect to their support of non-functional properties. To this end, we first identify and describe a set of relevant quality attributes for the context of AAL systems. Then we evaluate given AAL platforms by means of a qualitative analysis technique, describing to which degree the different quality attributes are fulfilled by each platform. The results are compiled into an evaluation of the platforms that should provide support for developers of AAL projects when faced with selecting an appropriate platform.

The remainder of this paper is organized as follows: In Section 2, we briefly discuss suitable quality attributes for our evaluation and provide a short overview of the selected attributes. Subsequently, we present the considered AAL platforms and their evaluation in Section 3. In Section 4, we conclude this paper with a discussion of our results, lessons learned, and limitations of this work.

2 Quality Attributes as Evaluation Criteria

As a basis for the platform evaluation, we first had to identify an appropriate (i.e., small enough to be described in this paper yet meaningful) set of quality attributes for the AAL domain. A general starting point was provided by established standards for software quality such as ISO/IEC 9126, ISO/IEC 14598, and the new ISO/IEC 25000 SQuaRE. From these comprehensive sets of quality attributes we extracted a subset that was well suited for assessing the considered AAL platforms from a quality assurance perspective (cf. Table 1). As the main categories (or, according to the wording used in the standards, as the top-level quality characteristics) to be considered, we mostly adopted those proposed in the ISO/IEC 9126 and ISO/IEC 25000 SQuaRE standards. However, we made some slight changes owing to our task of evaluating AAL platforms. In correspondence with the standards, we considered maintainability and efficiency because both play important roles in AAL systems and depend on the platform. As for the former, it is common for AAL systems to be continuously maintained over their life-time as the needs of the assisted person usually change. Efficiency is important because AAL systems consist of heterogeneous distributed devices, which might only have scarce resources. In contrast to the standards, we further chose to include safety and security as top categories in addition to reliability. This is because we believe that AAL systems need to be particularly trustworthy, with trustworthiness being a composite of different quality characteristics including (but not necessarily limited to) safety, reliability, and security [3,12]. Safety takes up a special position when compared to the other quality attributes considered, because it is always a system property. Thus, we considered the general eligibility of the platforms for safety-critical applications. More precisely, we evaluated if there are weak points within the platforms (i.e., single points of failure) or if the platforms offer mechanisms that are directed at or can be utilized for safety measures. As for security, we mainly focused on the protection of sensible information, hence considering confidentiality as a main quality characteristic. We consequently evaluated if there are suitable security mechanisms in place in the platforms. Also in contrast to the standards, we did not consider usability and portability. Usability is not a property of a platform but rather of the applications that run on a platform. The same argument is true for portability, which is mostly a characteristic of (application) software to be deployed in different contexts such as different platforms.

3 Overview and Evaluation of AAL Platforms

Due to the large number of existing platforms, we do not aim to provide a complete overview in this paper. We consider the following set of more consolidated and well-known AAL platforms: Alhambra [5], Hydra [6], OASIS [8], OpenAAL [9], PERSONA [11], and UniversAAL [13]. It is important to point out that these platforms are based on OSGi[1]. However, this was not a criterion in the selection process; we only realized this during the analysis. In the following, we first give a short overview of the platforms considered. Then we describe the method applied to gather the required platform-specific information. Finally, we describe the evaluation results and provide a table that shows the platforms based on the set of quality attributes identified in Section 2.

3.1 Overview of Evaluated AAL Platforms

Alhambra: This platform provides a comprehensive architecture for accessing and integrating heterogeneous devices, providing interoperability with different communication protocols, and using a uniform functional interface. Alhambra is a service platform for developing modular, service-oriented, hardware-independent applications. In this perspective, Alhambra provides modularization, enabling an exchange of applications.

Hydra: Hydra is a service-oriented platform built for operating in environments with limited resources, such as energy, memory, and computational processing. It is a peer-to-peer based system that offers, among others, mechanisms for allowing service discovery and for ensuring high interoperability.

[1] OSGi has been considered one of the most appropriate frameworks to be used as a basis for the development of AAL platforms - http://www.osgi.org/

Table 1. Quality Attribute Framework for the AAL Domain

Quality Attribute	Description
RELIABILITY	
Recoverability	Recoverability can generally be defined as the ability of the system to recover if a failure does occur. In our evaluation we assessed if and how good recoverability is supported by mechanisms of the respective platforms.
SECURITY	
Encryption Mechanism	Encryption Mechanisms are essential for providing end-to-end message security in Web Services based environments. The use of digital certificates can simplify access control for elderly and demented persons (as they do not need to remember usernames and passwords).
User Roles & Security Profile Definition	The interface to core management services (i.e., user management, access management, role management, and token management) should be defined for the developer.
MAINTAINABILITY	
Changeability	Changeability is a property that allows software engineers to easily perform a change in the system design. In the case of the AAL platforms, we especially considered their facilities to maintain them after deployment.
Installability	Installability is the capability of a software product to be installed in a specified environment. For AAL platforms, we considered the presence of installation mechanisms that allow both people with and without technical knowledge to effectively add new services and devices to the system, as well as mechanisms for assuring that external dependencies will be automatically downloaded to assure proper (re)installation of the system.
EFFICIENCY	
Adequacy for Small Devices	Given the heterogeneous nature of AAL systems, it is important for a corresponding platform to soundly incorporate devices ranging from sensor nodes to PCs. This can be particularly difficult for devices like sensor nodes, as resource scarcity might require special concepts in order to integrate them into a platform.
Resource Consumption	Here we considered the general resource efficiency of the platform (e.g memory, CPU).
Communication Overhead	AAL systems usually rely on numerous distributed devices and platforms are required to realize their communication in an efficient way.
SAFETY	
Presence of Single Point of Failure	A single point of failure corresponds to a system component that, if it fails, will compromise the proper functioning of the system.
Safety Pattern Usage	Safety patterns are measures applied to the system architecture that will assure that the system will always be in a safe state. Examples of safety patterns are Homogeneous/Heterogeneous Redundancy, Watch Dog, and Triple Modular Redundancy.

OASIS: OASIS is an ontology-driven, open reference architecture and platform that facilitates interoperability, seamless connectivity, and sharing of content between different services. Based on a service-oriented approach, it is open, modular, and standard-based. It includes a set of tools for content/services connection and management, for user interface creation and adaptation, and for service personalization and integration.

OpenAAL: The main goal is to enable an easy implementation and integration of flexible, context-aware, and personalized services. The OpenAAL middleware is a framework that supports integration and communication between AAL services. Furthermore, it provides generic platform services such as context management, workflow specifications of system behavior, and semantically enabled discovery of services. Both the framework and the platform services operate and communicate by means of a shared ontology.

PERSONA: The PERSONA project aims at developing a scalable, open-standard technological platform for building a range of AAL services. The relevant technical solutions include a middleware, a set of general-purpose components (forming the PERSONA platform), and a set of AAL services.

The middleware comprises a set of OSGi bundles organized in three logical layers: The Abstract Connection Layer handles the peer-to-peer connectivity between middleware instances, the Sodapop Layer realizes the peer and listener interfaces, and the PERSONA-specific Layer implements different busses, which are employed to enable the interaction between users and the general-purpose components.

UniversAAL: UniversAAL is based on a service-oriented architecture that reuses many components of PERSONA. The platform includes three main parts: (i) a runtime-support environment that provides core services for the execution of AAL services, (ii) a development support that provides documentation, tools, and development resources, and (iii) community support, including training and an online store, a one-stop shop for AAL services and applications.

3.2 Methodology for Collecting Information

Considering the characteristics of our work, we decided to conduct a survey in order to evaluate the AAL platforms. The conduction of surveys is an empirical strategy for a retrospective investigation about a topic of interest [14]. According to Wohlin et al. [14], a survey is a descriptive (to determine the distribution of attributes and characteristic), explanatory (to understand decisions that are made), and explorative (a preliminary study for a deeper future investigation) strategy. In particular, we adopted the interview technique, which is a qualitative analysis technique [7] widely used to conduct surveys [14] and is, in particular, sufficient to evaluate the selected quality attributes of software system architectures [4]. More precisely, we decided to conduct semi-structured interviews, guided by a script, but with interesting issues explored in more depth. The interviews were individually held more than one time so that bias could be minimized.

3.3 Result Evaluation

For each AAL platform, the quality attributes were discussed and analyzed. Table 2 summarizes the results of our analysis. In order to indicate if a platform addresses a specific attribute, we adopted these abbreviations: (i) HA (Highly Addressed), if the attribute is explicitly supported; (ii) A (Addressed), if the attribute is supported; (iii) PA (Partially Addressed), if the attribute is implicitly supported or limited to single features; (iv) NA (Not Addressed), if an attribute is insufficiently supported or is not addressed; and (v) INA (Information Not Available), if information about that attribute is not available. Bellow, we present more details about the platforms with regard to each quality attribute.

Recoverability: Since all platforms are based on the OSGi framework, on the level of single bundles, they are all protected by the recoverability mechanism provided by the hosting OSGi runtime environment. Besides the common protection, we identified that: (i) **Hydra** has a specific component for detecting

Table 2. Evaluation of the AAL Platforms

	Alhambra	Hydra	OASIS	OpenAAL	PERSONA	UniversAAL
RELIABILITY						
Recoverability	NA	A	NA	NA	HA	HA
SECURITY						
Encryption Mechanism	NA	A	A	NA	A	HA
User Roles & Security Profile Definition	A	A	PA	PA	A	HA
MAINTAINABILITY						
Changeability	PA	PA	PA	A	HA	HA
Installability	NA	A	A	NA	HA	HA
EFFICIENCY						
Adequacy for Small Devices	HA	HA	HA	NA	A	HA
Resource Consumption	A	NA	NA	INA	HA	HA
Communication Overhead	A	NA	NA	A	HA	HA
SAFETY						
Presence of Single Point of Failure	NA	NA	NA	INA	NA	HA
Safety Pattern Usage	NA	A	NA	NA	NA	A

failures in the system; (ii) **PERSONA** and **UniversAAL** address recoverability also at the hardware level; and (iii) **Alhambra**, **OASIS**, and **OpenAAL** do not provide additional mechanisms for recovery when failures occur.

Encryption mechanism: We observed that: (i) **Hydra** has an encryption mechanism addressed by the Trust Manager component; (ii) **OASIS** has an encryption mechanism in the Trust and Security Framework; (iii) **PERSONA** has an encryption mechanism where the necessary authorization for enabling a middleware instance (representing a PERSONA-aware node) to take part in a certain AAL Space requires manual installation of the AAL Space shared key on that node. The presence of such an encryption mechanism additionally ensures end-to-end security and integrity of messages exchanged among the middleware instances; (iv) **UniversAAL**'s encryption mechanism extends Persona's with additional public & private key pairs for communication beyond a single AAL Space (e.g., with another AAL Space or with Web services outside the AAL Space); and (v) **Alhambra** and **OpenAAL** do not offer any encryption mechanism.

User roles & security profile definition: We observed that: (i) **Alhambra** offers mechanisms for defining user roles and profiles; (ii) **Hydra** has a policy framework where user roles and security profiles can be defined. Overall confidentiality is assured by asymmetric encryption as mentioned before, combined with this policy framework; (iii) **OASIS**'s approach to linking user roles and security profiles can lead to user role conflicts (two roles with conflicting properties can be associated with the same user profile). Moreover, there is no mechanism to prevent user information (which should be strictly confidential) to be seen and manipulated by other users, which implies notable confidentiality issues; (iv) **OpenAAL** offers a possibility to define user profiles for different users that might also contain security information. However, such security-related profiles are currently not available; (v) **PERSONA**'s approach with respect to user roles and security profile definition is basically structured as an extensive ontology of different classes of users with the appropriate profile models and a profiling component for accessing profile data. PERSONA offers mechanisms for user and component authentication which, in addition to the two security points above, contribute to confidentiality; and (vi) **UniversAAL**'s approach was extended from PERSONA by adding policy-based security mechanisms integrated into matchmaking between offers and requests. With this extension, the confidentiality of UniversAAL is improved from PERSONA's solution, since the policy-based extension provides generalized mechanisms that might even eliminate the need for application-level access control.

Changeability: In general, because of the good modularity offered by OSGi, it is not difficult to perform changes in systems that are based on or use these platforms. Besides that, we identified that: (i) **Alhambra**, **Hydra**, and **OASIS** do not offer any additional mechanism for improving changeability; (ii) **OpenAAL** has a clear and quite simple architecture for a very specific set of applications that allow several extension possibilities. Additionally, OpenAAL's architecture exploits semantic technologies, which enhance changeability even further. On the other hand, documentation of these possibilities is rather sparse; (iii) **PERSONA** offers mechanisms for replacing components on the fly. The distributed implementation of the PERSONA middleware provides dynamic plug-and-play of hardware and software artifacts. Actually, the PERSONA middleware can be regarded as Communication Middleware. i.e., it is distributed on nodes that are PERSONA-aware. The distribution is hidden by the middleware, so that any building block is regarded in the same manner on the platform level; and (iv) **UniversAAL**'s changeability mechanisms are basically the same as PERSONA's, with improvements on the modularity of the PERSONA middleware as a result of enhancing the distribution function.

Installability: The main points are: (i) **Alhambra and OpenAAL** have no special mechanism for supporting installability. Service installations are done via simple copy and paste of bundles ; (ii) **Hydra** offers a wizard for guiding the user through the installation process; (iii) **OASIS** does not offer barriers for installing new devices and services in the existing systems. However, it is important to know the ontology of what the service is about to conduct a smooth installation; (iv) **PERSONA**'s approach regarding installability is based on dynamic dependencies checking. The installation of dependencies and other external components is facilitated by reusable and predefined OSGi configuration files that allow communication with external repositories where the newest versions of software artifacts are located; and (v) **UniversAAL**'s approach was improved from PERSONA. In particular, improvements were made to tools for the creation of an initial dataset as well as for facilitating the download and installation of applications from online stores, along with personalization tools.

Adequacy for small devices: With regard to this attribute, we observed that: (i) **Alhambra** can be easily integrated with small devices and sensor nodes in existing systems, since there are well-defined interfaces that allow smart integration of such elements; (ii) **Hydra** was explicitly designed to interoperate with small devices and sensor nodes. It has a hybrid approach for supporting the following scenarios: If the small device contains a Hydra implementation, it communicates directly via service invocation. If the small device does not have a Hydra implementation on it, communication is done via a proxy called Hydra Proxy; (iii) **OASIS**'s adequacy for small devices is also addressed using a proxy based approach consisting of the integration of two OSGi implementations: the one on which the whole system is structured and the other one that is dedicated to supporting integration of small devices; (iv) for **PERSONA**, a design pattern was specified in order to allow the integration of small device and sensor nodes. Based on this specification, PERSONA provides concrete implementation for well-known home automation standards such as KNX as well as the home automation and health profiles of ZigBee; (v) **UniversAAL**'s adequacy regarding the support of small devices and sensor nodes was improved from PERSONA by offering additional support for the IEEE-11073 standard, automatic generation of code for new device wrappers, and commissioning tools. This means that for each class of protocols (known to the system), UniversAAL offers the possibility to generate "virtual representations" compliant with the internal data/device model. In PERSONA, these internal representations had to be manually generated; and (vi) **OpenAAL** does not address this attribute.

Resource consumption: All of the analyzed platforms require a minimum resource for running OSGi, the Java Virtual Machine (JVM), and a database which, in general, is a lightweight one. In more detail, we observed that: (i) **Alhambra** has well-structured resource consumption management modules for ensuring that unnecessary resources will not be consumed. In general, resource consumption is very low; (ii) **Hydra**: Besides the resources needed for OSGi, JVM, and database, the Network Manager of Hydra consumes a considerable amount of physical memory; (iii) **OpenAAL** resource consumption was never evaluated in detail by the OpenAAL team. Nevertheless, the platform designers assume that the core parts are small enough to consume very few resources. The biggest resource consumers are memory and processor task; (iv) **PERSONA** and **UniversAAL** require 4MB of space and Java 1.3 running. It is important to point out that around 2MB are for OSGi. In order to achieve an optimal load balance, platform components can be distributed without restriction to the different available nodes in the AAL Space. Runtime measurements on memory and CPU consumption have not been performed so far; and (v) **OASIS** does not offer any management for resource consumption.

Communication overhead: Regarding this attribute, we observed that: (i) in **Alhambra**, on the Bus communication level, a Queue mechanism is used. The overhead depends on the size of the Queue. On the Service level, the OSGi mechanism takes care of this aspect and, in general, has very low overhead. For communication with external devices, there is a specific dedicated communication protocol for avoiding high communication overhead; (ii) in **Hydra**, the main cause of communication overhead is the Network Manager, which is based on JXTA; (iii) in **OASIS**, an the service level there is an overhead caused by a proxy-based approach to orchestrating the services. Another overhead is caused by the hybrid approach used in the data model: The most important information of the user profile is replicated in the central server and in the local node; (iv) in **OpenAAL**, communication is highly dependent on the OSGi communication framework, which results in little communication overhead; and (v) **PERSONA** and **UniversAAL** have a dedicated mechanism for dealing with communication overhead that is realized in two ways: Persona communicates through 4 busses. For 2 busses, n-1 (n = number of nodes) messages are broadcasted ONCE when a node throws a new event. The rest of the procedure is executed locally at each node. For the other two busses, communication is centralized, that are always optimized with respect to overhead). They introduce a "Coordinator", so that CONSTANTLY 0, 2, or 4 messages, in the worst case, are received. With other approaches, usually 2n messages are received.

Presence of single point of failure: We observed that: (i) in **Alhambra**, the Residential Gateway is a single point of failure for the whole platform; (ii) in **Hydra**, all communication is done via the Network Manager, which communicates via a single server called Super Node that, in turn, becomes a single point of failure; (iii) in **OASIS**, a single point of failure is the service registration component; if this component fails, service calls or registration will not be possible; (iv) **PERSONA**'s basic architecture and the distributed implementation of the middleware were explicitly designed to avoid single points of failure. Nevertheless, there are few mandatory platform components on top of the middleware that make the hosting node a critical one. The failure of such a node leads to the loss of certain functionalities. For instance, if the node hosting the PERSONA Dialog Manager fails, explicit interactions with the user will not work anymore; (v) in **UniversAAL**, there is no single point of failures; and (vi) regarding **OpenAAL**, no information is available.

Use of safety patterns: Regarding this attribute, (i) **Hydra** presents Redundancy as a safety pattern, through the use of different services with the same goal; (ii) for **UniversAAL**, explicit mechanisms are under development to support redundant installation of critical components; and (iii) the other platforms do not use any safety patterns.

Summarizing our analysis, we observed that the AAL platforms considered in this work are relevant and present different advantages in various contexts of use. Considering the analyzed set of quality attributes, the results point out that none of them fully addresses the quality attributes; however, overall, UniversAAL presented the best evaluation. Considering other quality attributes, the analysis might have a completely different result. With respect to our quality attribute set, it is important to point out that the use of safety patterns, in general, has not been taken in consideration, even knowing that their use could avoid failures leading to serious injuries to the assisted persons. It is worth highlighting that this type of analysis is not trivial, since it involves a huge amount of information from different sources and, beyond that, information analysis and summarization require considerable efforts and are time consuming. Thus, the performed evaluation per se demonstrates the necessity of a well-structured evaluation approach.

4 Conclusions

Selecting an adequate AAL platform is essential for the success of AAL projects. The main contribution of this paper is to present an evaluation of well-known AAL platforms, based on quality attributes analyzed on the architectural level and providing information that could provide guidance in the selection of the appropriate platform for a new AAL project. As future work, we intend to consolidate our analysis by performing scenario-based evaluations, also involving other AAL platforms and aiming at contributing to the effective development of AAL projects.

Acknowledgements. This work is supported by the OptimAAL and ProAssist4Life projects, and by the Brazilian funding agencies FAPESP and CNPq. We would also like to thank Saied Tazari from Fraunhofer IGD, Peter Wolf from FZI Karlsruhe, Marius Ofgen from TU Kaiserslautern, Mario Schmidt from Fraunhofer IESE, Lohrasb Jalali and Patrick Lukat from Fraunhofer IMS and Marc Jentsch from Fraunhofer FIT for serving as interview partners. We would also like to thank Sonnhild Namingha from Fraunhofer IESE for linguistic support.

References

1. AAL Joint Programme: Ambient Assisted Living (AAL) joint programme, World Wide Web (2011), `http://www.aal-europe.eu/` (acessed May 16, 2011)
2. AAL Open Association: AAl Open Association - AALOA, World Wide Web (2011), `http://www.aaloa.org/` (acessed May 16, 2011)
3. Avižienis, A., Laprie, J., Randell, B., Landwehr, C.: Basic concepts and taxonomy of dependable and secure computing. IEEE Trans. on Dependable and Secure Computing 1(1), 11–33 (2004)
4. Clements, P., Kazman, R., Klein, M.: Evaluating Software Architecture. The SEI Series in Software Engineering, Boston, MA (2002)
5. Dimitrov, T.: Design and Implementation of a Home Automation Service Gateway based on OSGi. Master's thesis, University of Duisburg-Essen, Düsseldorf, Germany (December 2005)
6. Hydra Project: Hydra open source middleware, World Wide Web (2011), `http://www.hydramiddleware.eu/` (acessed May 17/2011)
7. Miles, M.B., Huberman, M.: Qualitative Data Analysis: An Expanded Sourcebook, 2nd edn. Sage Publications (1994)
8. OASIS Project: OASIS: quality of life for the elderly, World Wide Web (2011), `http://www.oasis-project.eu/` (acessed May 11, 2011)
9. OpenAAL: OpenAAL: The open source middleware for ambient-assisted living, World Wide Web (2011), `http://openaal.org/` (acessed May 16, 2011)

10. Passau Workshop on ICT & Ageing: Announcing the European Initiative for an AAL Platform: Which Features Should Be In AAL Platforms, World Wide Web (2010), `http://www.hi.se/Global/monami/05PanelWhichFeaturesShouldBeInAAL1Platforms.pdf` (acessed May 13, 2011)
11. PERSONA Project: PERceptive Spaces prOmoting iNdependent Aging, World Wide Web (2011), `http://www.aal-persona.org/` (acessed May 13, 2011)
12. Schneider, D., Becker, M., Trapp, M.: Approaching runtime trust assurance in open adaptive systems. In: SEAMS 2011 at ICSE 2011, Hawaii, USA (2011)
13. UniversAAL Project: The UniversAAL Reference Architecture, World Wide Web (2011), `http://www.universaal.org/images/stories/deliverables/D1.3-B.pdf` (access in March 25, 2011)
14. Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B., Wesslén, A.: Experimentation in Software Engineering. Kluwer Academic Publishers (2000)