Context-Aware Integration of Smart Environments in Legacy Applications

Philipp Lehsten, Alexander Gladisch, and Djamshid Tavangarian

University of Rostock, Graduate School MuSAMA
{firstname.lastname}@uni-rostock.de
http://www.musama.de

Abstract. As opposed to conventional applications, smart environments are designed to offer transparent user assistance by decoupling users from devices. Apart from the lack of realised systems there are numerous applications that are strongly interwoven with the users' workflow and hard to replace, commonly called legacy applications. Instead of creating new applications, our approach is the loose integration of these both, the smart environment and the legacy application. In our work, we propose a generic architecture that is applicable to various kinds of environments and applications. The architecture comprises an intermediate layer that enables a loose coupling between smart environment and legacy application. Furthermore, we introduce a workflow to refine the generic architecture to fit the requirements of specific use cases. For our use case, we apply the vision of a pervasive university. Here, we integrate functionalities of smart lecture rooms into a learning management system that is commonly used in German universities and therefore hard to replace.

Keywords: smart environment, legacy application, pervasive computing, service-oriented architecture.

1 Introduction

Since pervasive computing was envisioned by Weiser [1], developing Smart Environments (SEs) is a major objective for researchers of various disciplines. SEs are designed to offer "anytime and anywhere computing" and moreover transparent user assistance by decoupling users from computing devices.

Assisting the user in his workflow is only partially realized by today's applications. However, workflows may comprise not only computing tasks, which can be facilitated by applications, but also interactions with the physical environment that can be improved by assistance of these SEs. Therefore, the consequence is the need to integrate smart capabilities in existing applications that already assist the user in his workflow in general and especially for applications that cannot be easily replaced, e.g. legacy applications.

Decoupling users from computing devices implies unobtrusiveness by hiding the complexity from the user. For that purpose, in the first visions on SEs, complete proactive user assistance was discussed [2]. However, undesired situations

D. Keyson et al. (Eds.): AmI 2011, LNCS 7040, pp. 126–135, 2011.

may occur because of error-prone intention recognition mechanisms. Moreover, if an explicit user interface is missing, the user is not able to adapt a new configuration. For these reasons, the users need options to interact, without getting overburdened by complexity of the SE.

To achieve these goals, we propose a generic architecture, which is applicable to various kinds of SEs and applications. The architecture comprises an intermediate layer between both systems and therefore enables the integration of the smart capabilities into an existing application. The layer includes an explicit but also context-aware user interface, which provides further assistance, e.g. by providing preset configurations. Thereby, the user can interact directly with the environment without being aware of the complexity. Furthermore, we describe a general workflow to analyse specific environments and applications. Based on the analysis results, the generic architecture can be refined to fit a specific use case. Finally, we present an enriched learning management system in the context of a pervasive university as implemented use case.

The rest of the paper is organized as follows: preliminary considerations are taken in section 2 to foster our approach. Section 3 introduces related work. Section 4 defines the generic architecture with the intermediate layer, while section 5 presents our workflow to refine our generic architecture for a specific use case. Then we present a use case according to the vision of a pervasive university in section 6. Here, we integrate functionalities of smart lecture rooms in a learning management system that is commonly used in German universities. Finally, we present our conclusion in section 7.

2 Preliminary Considerations

In general, SEs are considered as systems to support the users to achieve their goals. For this purpose, user intentions need to be recognized, e.g. by evaluating information that relates to users current situation. Such information is considered as context [3]. Additionally, devices and services, which are integrated into the environment, need to be configured and controlled in order to assist the users. Depending on the capabilities and available facilities of the environment, this task can be very complex.

SEs are applied for various fields of applications, smart meeting rooms [4], smart homes [5], smart university campus [6] or smart transportation systems (known as intelligent transportation systems) [7] are common examples. Accordingly, requirements on the infrastructure and therefore the context information differ significantly, not only in pure availability but also in granularity. For example smart homes for people, who suffer from dementia, have to recognize fine-grained subtasks to prevent injuries or to continue assistance after interruptions of users workflow. Therefore, user position and other fine-grained context data, as well as sensors to gather this information, is needed. On the other hand, assistance in smart lecture rooms can be realized by collecting coarse-grained context, such as: "Which person is giving a lecture at what time?". Here, among others identification and authorization of the user are necessary tasks of the workflow and can be realised utilising global repositories.

By now, workflows for simple tasks are often facilitated by single applications where workflows for complex tasks are mostly encouraged by compound enterprise applications. These kinds of applications are often legacy applications and therefore replacement is laborious as well as time consuming. However, up to now, both kinds of application do not integrate assistance provided by SEs, whereas the integration of assisting capabilities into the user workflow is promising.

Despite the problems caused by heterogeneity and complexity of the SEs and existing applications, the coupling offers also multiple benefits. Legacy applications are often used as storage for large amounts of context information (e.g. resource management of rooms, time schedules or personal information) and they provide a graphical user interface as well as identification and authorization mechanisms. Therefore they are able to provide some information needed to evaluate the user context without using installed hardware sensors. Furthermore, they offer an interface, which the user is already used to and which is already mapped to the workflows. Using these applications as interface can help to accustom users to newly installed technologies. Therefore, we advocate the integration of an interface to the SE into the user interface of legacy applications. The generic architecture, we propose, comprises an intermediate layer that includes such an interface and allows a loose coupling of the legacy system with the SE. This allows among others a fast adoption to new interfaces and the support of a multitude of SEs.

3 Related Work

When examining user interfaces (UI) for SEs, physical UIs are highlighted in many cases. In [8], a toolkit for physical UIs for ubiquitous computing environments was developed. The focus of this work was on the integration of different I/O devices and their software proxies. In [9], Smartphones were considered as the default physical UI for ubiquitous computing applications. In their work, the authors examined and evaluated several interaction techniques where the capabilities of smartphones were used.

In [10] three different physical interfaces (PC, media terminal and smartphone) to control smart home environments were analysed. In their work the authors highlighted that users might not be ready to interact with their familiar environment by using new interaction techniques. Furthermore, they did a user study to evaluate the considered UIs. Here, the smartphone was also the most frequently used UI. Beyond the interfaces itself, various types of interactions with the UIs of SEs were identified in [10], among others speech, gesture and graphical UI (explicit UIs) as well as automatic interaction (implicit UIs). The subjects of the user study ranked automated interaction as the most relevant technique but they also highlighted that a full automation is undesirable due to missing possibilities of human intervention. The subjects wanted chains of functions (preset configurations) they could set up themselves.

In contrast to that, in [11] the authors postulate that (learning) activities are neither bound to a specific environment nor prestructured. For that reason,

the users (learners) should be able to create their own learning environment by configuring the available resources of the environment in ways they find most comfortable. Due to the multitude of available resources and functionalities in a SE, we doubt that this is possible for the user. Therefore, we want to support in our work the user by providing preset configurations, where the user is able to choose the environmental components, which fit best to his requirements.

4 Generic Architecture

Despite the heterogeneity of possible SEs and applications, we define a generic architecture, which is theoretically applicable to various combinations of SEs and applications. Afterwards, we introduce in section 5 a workflow, which allows the customization of the generic architecture for specific use cases. The proposed architecture comprises a transparent intermediate layer, which couples the SE and the legacy application. From the application point of view, the layer provides an extension of the UI to integrate additional functionalities. This requires a plug-in functionality or at least some kind of access to the source code of the application interface. From the other point of view, it provides commands to control the SE. Here again some kind of interface access is needed, like web services that can be used to control the SE. Thus, the intermediate layer allows a loose coupling without losing independent functionality of the SE and the application. Furthermore, the interfaces of the proposed intermediate layer can be exchanged to integrate other applications or SEs.

The intermediate layer consists of three components as shown in figure 1: (1) adaptive GUI, (2) rules engine and (3) context storage with importers. The core

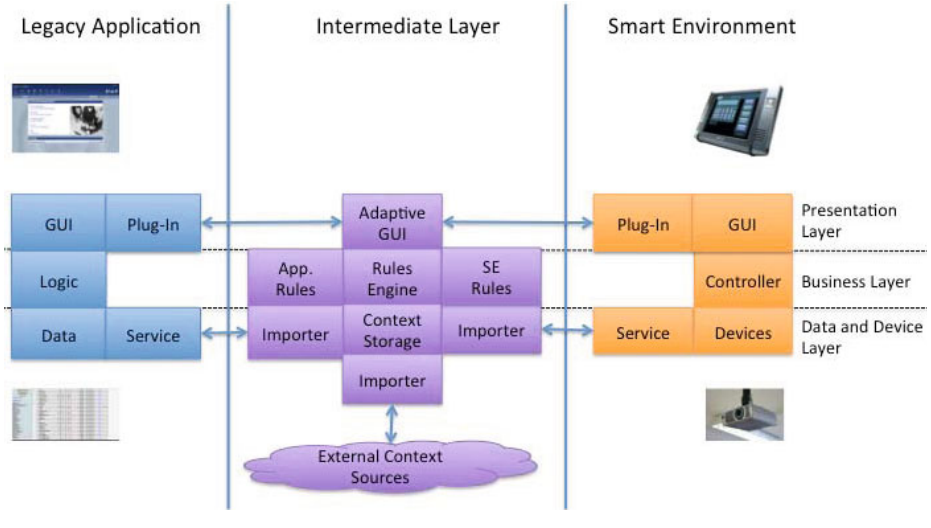

Fig. 1. Proposed architecture with intermediate layer to interconnect smart environment and legacy application

of the intermediate layer is the rules engine, which does the context processing. This engine integrates the logic of intended assistance functionalities into the system by providing and executing rules. The rules are based on context information, which is provided by a context storage. The context storage receives context information from the importers, stores and manages it and aggregates the context information if needed. To realize the rules engine, e.g. a knowledge-based system can be applied. The output of the rules engine are context-based preset configurations for the adaptive GUI, which allow to configure and to control assistance of the SE easily. Therefore, the complexity of the configuration remains widely hidden from the user. Moreover, on basis of the context, only preset configurations, which are most properly applicable to assist the user in his current situation, are produced by the rules engine. Thus, the interaction of the user with the SE is further simplified. The set of possible preset configurations is filtered and therefore the (manual) choice of the preset configurations, which fulfil the users demands best, is less complex for the user.

The context information itself is gathered by the importers. Every importer consists of a data converter and a communication handler. The data converter allows the conversion of gathered context information into a target format, which can be used by the rules engine. The context information is gathered by the communication handlers, which encapsulate the communication with the context information source. Therefore, they act as gateway between intermediate layer and SE or legacy application respectively. The realisation of these components can be done with web services or agents.

The preset configurations are provided to the user via (explicit) UI. The UI is customized to the presentation layer of the application. Here, it is possible to generate add-ons for a GUI (e.g. an additional website for an web interface) or any other kind of UI if needed. Due to the interconnection to the adaptive GUI, the UI receives the generated preset configurations. The manual choice of a specific preset configuration is easy for the user. The context-aware generation of preset configuration limits the number of options that are presented to the user. Therefore, complexity of the systems remains mostly hidden to the user. The interconnection of the user interface to the SE enables the direct application and execution of preset configurations. Thus, the SE is configured according to the preset chosen by the user and assists the user as desired.

The entire architecture is developed as modular as possible to allow extensions in any direction easily. An extension of the architecture allows the seamless integration of various SEs into various applications while using the same rules engine and context repository.

5 Workflow for Pervasive Service Integration

Due to heterogeneity of SEs and legacy applications, we propose a workflow to refine our generic architecture to fit the requirements of specific use cases. The workflow consists of three phases: (1) analysis phase to identify possible interactions and interfaces between both as well as available context data; (2)

concept phase to refine the generic architecture for the specific use case on basis of the gathered data in analysis phase; (3) implementation phase which includes, in addition to implementation, basic functional tests.

5.1 Analysis Phase

In this phase, the SE as well as the legacy application need to be examined. The analysis consists of two parts: (1) technical analysis and (2) logical analysis. Table 1 provides an overview on examined attributes.

In the technical analysis, technical possible integration points for interfaces (to establish a connection and to integrate the UI) and available functionalities need to be identified. Thus, the infrastructure of the environment as well as the software architecture need to be unveiled. Furthermore, used technologies (e. g. web services) need to be examined. While using information that was gathered in this part of the analysis, it is possible to design the technical integration.

Table 1. Analysis phase: Overview on examined attributes

Analysis	Smart Environment	Legacy Application
Technical	- Accessible services - Software interfaces - User interfaces	- Software architecture - Software technology - Interfaces (plug-ins, UI)
Logical	- User assistance - AAA - Context data - Rules and decision models	- Use case - AAA - Context data

In the logical analysis, we examine the overall context and possible use cases of user assistance. Thus, factors such as use case of the application, assistance functionalities provided by the SE, AAA (authentication, authorization, accounting), and available context data need to be examined. Additionally, the users workflow needs to be examined to identify concrete actions that can be supported. A user workflow consists of actions the user executes on application and / or environment to perform a task. Here, we include tasks that are achieved by using the application (such as navigating, authenticating or downloading of files) as well as manual tasks (such as interconnecting or adjusting devices). Thus, the logical analysis allows the identification of integration points on a semantic level.

5.2 Concept Phase

In this phase, a concept to refine the generic architecture on basis of the gathered data in analysis phase will be developed. The phase is divided into three parts: (1) process definition; (2) definition of context data flow; (3) integration of data converters.

In the process definition, the potential user workflows are identified, which can be assisted by applying capabilities of the SE. Therefore, the workflows are

reproduced, including the tasks that can be supported by the SE. The workflows with high benefits are chosen while those where the added functionality is only marginal can be neglected. This ensures, that the support by SE is integrated only into relevant workflows, which reduces the work and maintenance effort.

The next step identifies the required context data and sources. The processing of this information is done by applying rules stored in the rules engine. To design these rules, every action of the SE in the workflow is enriched with conditions, which need to be met. These rules are evaluated by a rules engine, which checks the facts of the conditions in the background. Facts are provided by importers which are specific for every context source and therefore every context source needs an importer. Now the context information and rules are added as well as possible interfaces to local and global information sources. This ensures that the system can access every information, which is necessary to meet the requirements in the workflow as well as possible security considerations.

In the last part the converters for the importers are defined. This makes sure that the data provided by the importers match the format used by the rules engine. Additionally, the rules are refined to be as precise as possible to handle every possible condition. At this point, the developed workflow should be as fine grained as possible. If there are no inconsistencies left, then the last phase with the implementation can be approached.

5.3 Implementation Phase

The implementation phase is divided into realisation of the refined architecture and following functional tests. The implementation is processed according to results of the first and second phase. To execute functional tests is possibly complex, especially due to context-awareness of the system. Here, in addition to the functionality of the specified processes also the influences of different context information needs to be taken into account.

6 Use Case

For our use case, we apply the vision of a pervasive university [6]. Especially for the deployment and test of SEs, a university campus is a good environment. Usually, it is well equipped with communication and service infrastructure. In addition, there are many young people with a high grade of affinity to new technologies available within a university. Thus, several proposals that apply the ubiquitous and pervasive computing paradigm for campus environments can be found in literature [12], [13], [14], [15], [6] and reflect this advantages.

To implement the proposed architecture and to evaluate the practicability of our workflow, we decided to enrich the Stud.IP [16] learning management system (LMS) by integrating functionalities of a smart lecture room, which is available in our university. Stud.IP can be considered as a legacy application, which is used by numerous German universities and educational institutions. It provides an all-purpose web 2.0 platform to support students in their studies, lecturers

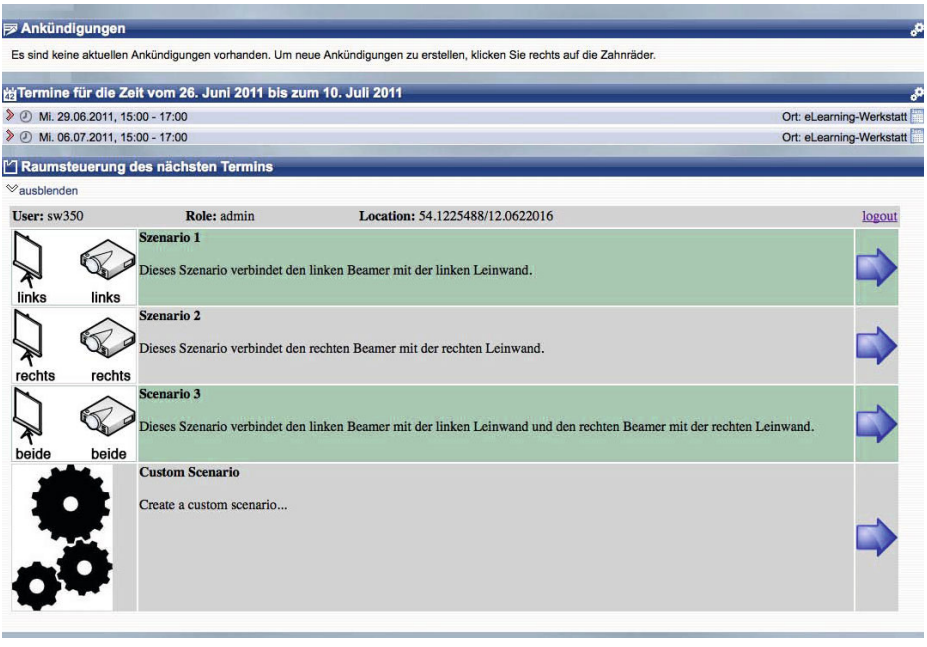

Fig. 2. Screenshot of the presented use case

in giving lectures and administration as well as campus management tasks. The smart lecture room provides several sensors (e.g. SensFloor) and other integrated devices (multiple digital projectors, monitors, cameras, microphones, visualizer, smartboard, etc.) to assist the lecturer. Currently, the room is controlled by an AMX Board with proprietary software. By implementing our architecture for this scenario, we want to achieve two goals: (1) the simplification of configuration and control of the smart lecture room as well as (2) a loose coupling of smart lecture room and LMS with a familiar UI for the lecturer. Furthermore, we want the ability to exchange the smart learning environment easily. Following the proposed workflow, our analysis revealed results as shown in table 2.

On basis of the analysis we decided to realize the UI for the integration of smart lecture room functionalities as an extension of the Stud.IP web page, which is available for every lecture course and which is accessible only by lecturers. The importer for the Stud.IP was realized using a web service, which interconnects the Stud.IP DB via MySQL. The importer for the smart lecture room is realized by using the available web services. To interconnect the university LDAP a web service is used as importer as well. The knowledge gathered by the web services is transferred via message bus to a knowledge-based system, which was implemented using JBoss Drools. Here, predefined rules are applied on the gathered knowledge; preset configurations are generated and provided to the GUI. Thus, suitable preset configurations are displayed as web page, which is embedded in Stud.IP as shown in fig. 2.

Table 2. Analysis results of Stud.IP LMS and smart lecture room

Analysis	Smart lecture room	Stud.IP LMS
Technical	- Device control - Web services - GUI (via AMX-Board)	- 3-tier web application - PHP, Java Script, MySQL - Web services, plugins, DB, GUI
Logical	- Presentation assistance - Only physical access restriction - Indoor position via SensFloor - Predefined preset configurations	- Lecture management, course schedules, lecture supporting media files - University LDAP, Stud.IP user groups - (Web 2.0) user profiles, time and room schedules, university calendar

7 Conclusions

In this paper, we introduced a generic architecture to integrate SEs into existing, especially legacy, applications. The architecture acts as intermediate layer between SE and application and allows the loose coupling of both independent systems. For that purpose, we introduced a workflow which includes analysis of current situation, refinement of the proposed architecture and implementation. On basis of our concept, we implemented a use case referred to the vision of the pervasive university. Here, we integrated functionalities of a smart lecture room into the learning management system Stud.IP.

Acknowledgements. Philipp Lehsten's and Alexander Gladisch's work are both supported by the German National Research Foundation (DFG), Graduate School 1424 MuSAMA. The authors would also like to thank Martin Hölzel and Stefan Wendt for implementing the system.

References

1. Weiser, M.: The Computer for the 21st Century. Scientific American, (Communications, Computers, and Network) (September 1991)
2. Satyanarayanan, M.: Pervasive computing: vision and challenges. IEEE Personal Communications 8(4), 10–17 (2001)
3. Abowd, G.D., Dey, A.K., Brown, P.J., Davies, N., Smith, M., Steggles, P.: Towards a Better Understanding of Context and Context-Awareness. In: Gellersen, H.-W. (ed.) HUC 1999. LNCS, vol. 1707, pp. 304–307. Springer, Heidelberg (1999)
4. Giersich, M., Heider, T., Kirste, T.: Ai methods for smart environments: A case study on team assistance in smart meeting rooms. In: Proceedings of Scientific Workshop 1: Artificial Intelligence Methods for Ambient Intelligence on European Conference on Ambient Intelligence (AmI 2007), Darmstadt, Germany (November 2007)
5. Orpwood, R., Adlam, T., Evans, N., Chadd, J., Self, D.: Evaluation of an assisted-living smart home for someone with dementia. Journal of Assistive Technologies 2, 13–21 (2008)

6. Lucke, U., Tavangarian, D.: Eine Service- und Kontext-basierte Infrastruktur für die Pervasive University. In: GI Jahrestagung, pp. 1935–1949 (2009)
7. Wang, F.-Y., Zeng, D., Yang, L.: Smart cars on smart roads: An IEEE intelligent transportation systems society update. IEEE Pervasive Computing 5(4), 68–69 (2006)
8. Ballagas, R., Ringel, M., Stone, M., Borchers, J.: Istuff: a physical user interface toolkit for ubiquitous computing environments. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI 2003, pp. 537–544. ACM, New York (2003)
9. Ballagas, R., Borchers, J., Rohs, M., Sheridan, J.G.: The smart phone: a ubiquitous input device. IEEE Pervasive Computing 5(1), 70–77 (2006)
10. Koskela, T., Väänänen-Vainio-Mattila, K.: Evolution towards smart home environments: empirical evaluation of three user interfaces. Personal and Ubiquitous Computing 8, 234–240 (2004), doi:10.1007/s00779-004-0283-x
11. Syvanen, A., Beale, R., Sharples, M., Ahonen, M., Lonsdale, P.: Supporting pervasive learning environments: adaptability and context awareness in mobile learning. In: IEEE International Workshop on Wireless and Mobile Technologies in Education, WMTE 2005, page 3 (November 2005)
12. Weiser, M.: The future of ubiquitous computing on campus. Commun. ACM 41(1), 41–42 (1998)
13. Griswold, W.G., Shanahan, P., Brown, S.W., Boyer, R., Ratto, M., Shapiro, R.B., Truong, T.M.: Activecampus: experiments in community-oriented ubiquitous computing. Computer 37(10), 73–81 (2004)
14. Barkhuus, L., Dourish, P.: Everyday Encounters with Context-Aware Computing in a Campus Environment. In: Davies, N., Mynatt, E.D., Siio, I. (eds.) UbiComp 2004. LNCS, vol. 3205, pp. 232–249. Springer, Heidelberg (2004)
15. Al Takrouri, B., Canonico, A., Gongora, L., Janiszewski, M., Toader, C., Schrader, A.: Eyejot - a ubiquitous context-aware campus information system. In: 2nd International Conference on Pervasive Computing and Applications, ICPCA 2007, pp. 122–127 (2007)
16. Stud.IP developer group. Stud.ip - Studienbegleitender Internetsupport von Praesenzlehre, `http://www.studip.de/home/` (last access: May 2011)