An Architecture for Adaptive and Adaptable Mobile Applications for Physically Handicapped People

Matthias Betz[1], Mahmudul Huq[1], Volkmar Pipek[1], Markus Rohde[1], Gunnar Stevens[1], Roman Englert[2], and Volker Wulf[1]

[1] Faculty of Information System and New Media,
University of Siegen, Siegen, Germany
{matthias.betz,syed-m.huq,volkmar.pipek,
markus.rohde,gunnar.stevens,volker.wulf}@uni-siegen.de
[2] T-Laboratories at Ben-Gurion University (of the Negev), Israel
Beer-Sheva 84105, Israel
Roman.Englert@telekom.de

Abstract. Context-awareness is an important capability needed in devices in a ubiquitous computing environment. Ubiquitous computing devices use different types of sensors along with the user's interaction history in order to collect and store data. This data is then used to adapt the user's behavior to suit the current environment. In addition to the explicit modifications by user control, the behavior of these computing devices along with the interaction amongst one another depends on the continuously changing environment conditions. These characteristics require the development of systems that have both, adaptive and an adaptable nature. Context-awareness is particularly important for physically handicapped people. This is due to the fact that context-aware ubiquitous devices are able to help them detect changes in the surrounding, which handicapped people can not do for themselves. In this research paper we suggest a general architecture of Context-Aware Adaptable System (CAAS). We exemplify this architecture with an Ambient Service prototype that we have developed.

Keywords: Context-Aware Adaptable System (CAAS), Ambient Service (AS), End User Development (EUD), Adaptivity, Adaptability, Mediation.

1 Introduction

"Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and application themselves." [1].

Context-awareness is the most important feature of the ubiquitous and pervasive computing [1] [2]. It simplifies the interfaces we present to our mobile users, adapting them to the current situations [3]. Context-awareness is also the key characteristics of new smart and context-aware information services. It includes all kinds of situational properties like spatial and temporal contexts, task context, collocation context,

C. Stephanidis (Ed.): Universal Access in HCI, Part I, HCII 2007, LNCS 4554, pp. 335–344, 2007.

historical context etc. These systems also exploit their surrounding context to increase the suitability of a service or an application to the user's needs.

Context-aware systems support users in dealing with complex systems of technologies and devices. This is possible because they aim to understand the user's current task, location and time contexts and provide situation adequate support.

These characteristics make context-aware systems an interesting technology for physically handicapped people.

Physically handicapped users of communication technology face slow, labor intensive user interfaces, which make interactive conversations tedious and difficult [5]. Advancements in research on user interfaces and conversational prediction have made it feasible for devices to be developed that can assist users [6] [7] [8], particularly those with physical limitations, to communicate. Despite these advancements there are still opportunities for improvements. Users need decreased input load and significantly increased output speed without sacrificing intended conversational topics.

Designing context-aware systems for physically handicapped people is a complex task. Standard designs are based upon the active user's involvement for requirement capture and user testing to establish workable design [9]. Physically handicapped people can not participate actively during requirement capture session. They are also not able to take part in the extensive testing session.

Studying context-awareness systems in a real life setting demonstrates that the contexts sensed by these systems are often ambiguous. Although some of these ambiguities may be resolved using automated problem-solving techniques, many cases still require the user's involvement for the correct handling of the ambiguous context [4]. This leads to system approaches that have to have accurate and reliable adaptive services as well as adaptable concepts in order to allow users to intervene in a simple and transparent way whenever necessary.

Therefore new design challenges in context-aware systems are to interpret the context correctly not only to inform adaptive services, but also to empower the end users to understand and configure the behavior of the system [10] – to perform meta-design activities.

The research field of End User Development (EUD) has empowered the end users to adapt their own computer systems [11] [12] [13]. Some of the concepts already have been transferred to the construction of context-aware systems, for example, Programming By Demonstration (PBD) [14] [15]. Another interesting concept of EUD, namely Meta-Design Activity, can be applied in the context-aware systems. Meta-design activities empower end users by enabling them to act at designers at use time. It allows users to use domain oriented language to re-design and adapt current features to their own needs [11].

This paper focuses on the technical aspects which build a context-aware system that supports ubiquitous fitness, by presenting the architecture for mobile application [16]. The architecture explicitly addresses the idea of a combined approach for adaptable and adaptive services. By developing architecture with adaptable services, we allow end users to act as designers at use time. It is done by empowering them to redesign and adapt current features of the application, based on the architecture, to their own needs.

Thus our proposed architecture allows end users to perform meta- design activities. We will also exemplify this architecture with an Ambient Service prototype.

The research paper is organized as follows: in the second section, the core concepts of context-aware adaptations will be discussed. The third section will suggest general software architecture for implementing Context-Aware Adaptable System (CAAS). In the fourth section we will present a research prototype based on this CAAS architecture. Finally, we will give a conclusion of our work.

2 Context-Aware Adaptation

The use of ambient services is a common way to demonstrate the behaviors and activities of context-aware systems. An ambient service represents a high level concept that allows us to construct complex services out of primitive ones by connecting them with each other via data flows.

Service applications, i. e., software components that are available over a network, promises both accelerated software development and a more flexible and less erroneous application.

In the course of time, the ambient service may change. Firstly, the user develops himself as he uses the service and then performs adaptations to the service on his own. Secondly, the ambient service automatically recognizes changes in the environment and usage behavior or performance of the user and makes the user aware of the potential improvements to his service. Thus, ambient services offer both user and context driven adaptation. In dealing with the adaptation of ambient services, two different aspects of contextualization of the application can be seen: adaptivity and adaptability [17].

The aim of the adaptivity is to have systems that adapt themselves to the context of use with respect to their functionality, context selection and presentation and user interpretation. Systems displaying such adaptive behaviors regarding the context of use are called context-aware systems. The objective of context-aware system is to assist the users by proactively supplying what is actually needed.

The aim of adaptability is to empower end users to customize or tailor computer systems according to their individual context specific requirements. Such approaches allow for adaptations to dynamically changing unanticipated requirements. Such approaches allows for the end user to customize their domain specific expertise to the system.

We believe that the potential of context-awareness systems for physically handicapped people can only exploit, when we take the special expertise of these people into account. Therefore the core concept of context-aware adaptation is to develop applications based on the shared initiative of human and computer intelligences.

The concept of context-aware adaptation is grounded on the work of FreEvolve, a component based tailorable system [13] [18] [19] [20], and a design of context-aware system [21] [22].

Component based tailorability is an approach to transfer the connectional ideas of EUD to the field of component based systems [23] [24]. According to O. Stimmerling [20], a software system owns the meta property of component based tailorability, if

the representation elements of its tailoring architecture are descriptions of its compositions.

The main challenge for ambient intelligent environment is that the context changes rapidly, which means that the system has to dynamically adapt to the servers and components that can abruptly appear and disappear at any time. Dealing with dynamic context creates a new field of EUD, whereas the use of the intelligence of the user to interpret the context creates a new field of context-aware systems.

A Context-Aware Adaptable System (CAAS) is defined as a system that enables the end user to adapt the application or application unit by taking the context of use into consideration.

The role of EUD in context-aware adaptable software is to provide a software design that is accountable to the users. The design also develops tools that make the adaptation of the system easy to the end users [11] [13]. With the help of EUD, end users will be empowered to configure and compose these information technologies according to their diverse and changing needs.

In component based architecture, context-awareness is used to reduce the burden of context adaptation. In context-aware systems, the context is used as a filter to determine which building blocks and available services are relevant in the context.

3 The General Architecture of CAAS

In the following section we would like to discuss the general software architecture for CAAS. The OSGi standard[1] has been used as the basis for setting up the context-aware adaptation.

The proposed software architecture for CAAS shows the three core concepts, i. e., Context-awareness, End User Development (EUD) and OSGi Service Management. The Mediated Adaptation Manager is the most important functionality in the CAAS architecture. It maintains the mediators and coordinates the adaptation process.

3.1 System Architecture

The following subsections present the main parts of the CAAS architecture.

- **Context Manager**

The context manager provides the CAAS application with a rich context model with an associated quality of information. It is responsible for acquiring context information from various sources and through a variety of communication protocols. A significant increase in the expressiveness, complexity and quality of the represented context can be achieved by transforming the acquired context data in several ways.

The context manager merges the various aspects of the acquired information into a coherent entity. The context derived by the context manager is any information that can be used to characterize the situation of a person or a computing entity and to identify the need for adaptation.

[1] Cf. http://www.osgi.org

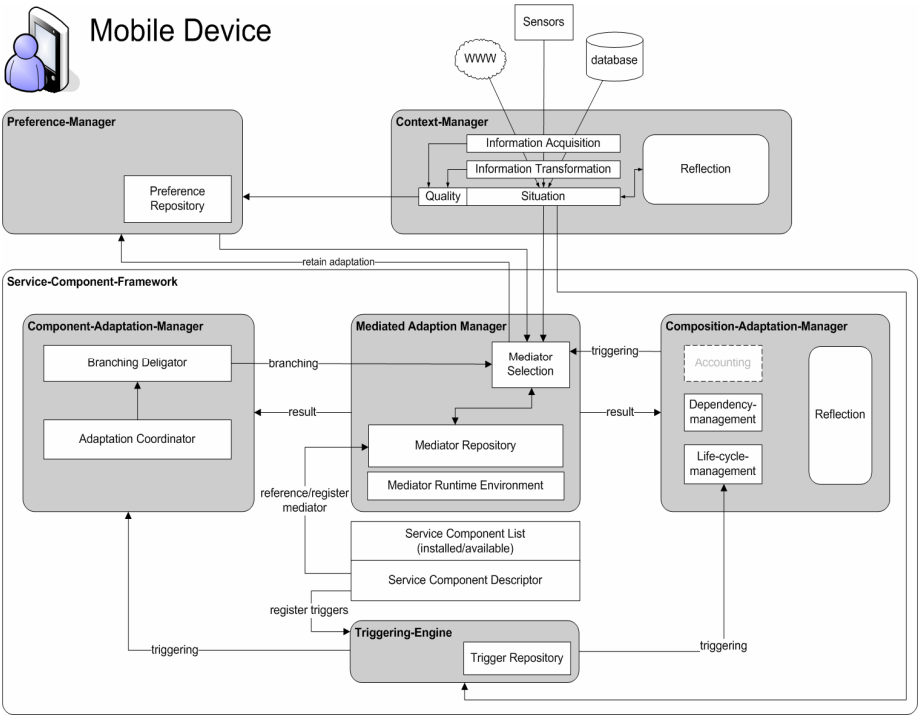

Fig. 1. Software architecture for Context-Aware Adaptable System (CAAS)

- **Reflection and Transparency Components**

The users need to understand and investigate the system's mode of operation in order to decide which modifications are needed to carry out. Inspection requires the application to externalize its current state, as well as its composition. Externalization or reflection means the provision of a human readable description of components which are plugged into the system, information about the current state of these components, and configuration settings needed for launching and operating the system.

- **Preference Manager**

The preference manager is responsible for storing the completed adaptation processes. It maintains a repository of user preferences. Each preference takes the form of a named pair that consists of a scope and a scoring expression. The scope describes the situation in which the preference is applied. A situation is described in predicate logic and may be evaluated as true or false. A preference is considered applicable within a given context only if the scope expression is true. The scoring expression assigns a score to a choice, which is a numerical value in the range [0 to 1].

- **Composition Adaptation Manager and Component Adaptation Manager**

In the proposed software architecture, adaptation works on two different levels of granularity. The coarse grained level adaptation is comprised of addition and removal

of entire blocks of functionality and is realized by the composition adaptation manager. The fine grained adaptation is the adaptation of a component itself, which is achieved through the component adaptation manager.

- **Mediated Adaptation Manager**

The mediated adaptation manager is responsible for delivering appropriate mediator functions in both adaptation phases, namely the composition and component adaptation phases.

Mediated adaptation describes an adaptation technique which separates the adaptation process from the operating software artifacts. The start of a mediated adaptation can be triggered by either the system or the end user. The completion of a mediated adaptation results in an accomplished and retained adaptation of the system. A mediated adaptation may be accomplished automatically, driven by the end user, or may be a sequence of both can occur in combination.

Every adaptation service component lists its required mediator in its own descriptor. For more specific mediators, it is also possible for service components to contribute component specific mediators in the mediated adaptation manager.

- **Triggering Engine**

The CAAS application developers create a set of rules, each consisting of an event and a corresponding action. These sets of rules are kept in a triggering repository. The triggering engine detects the occurrence of significant events and invokes actions in accordance with the rules. In some cases, the rules are associated with constraints as additional preconditions for the execution of actions that are evaluated following the events.

- **Service Component Description**

The service component description defines the behavior of a service component and specifies state transitions in the life cycle of the component in one or more situations. This enables the framework to build rules for the triggering engine to install, start, stop and uninstall a service component with respect to the actual situation. For the component adaptation, the triggering engine can use the same information for building rules for a situation dependent adaptation of the internal behavior of a component. The descriptor also defines the data required by the service component in order to reference the appropriate mediators from the mediator repositories or to register its own specific mediators.

4 Ambient Service Prototype

This section presents a prototypical implementation of the Ambient Service (AS) architecture. The prototype is part of a EUD research project in the health/fitness area.

We will describe the main parts of the architecture and their interactions. The heart of the architecture is the AS container. It is realized as an OSGi bundle[2]. It contains a view, presenting the main functionality to the user.

[2] We use Eclipse eRCP to realize our concept using some features of Eclipse which are not part of the official OSGi standard, e.g. the Eclipse workbench features (cf. http://www.eclipse.org/ercp).

Fig. 2. The action bar of the Service Component Framework

An omnipresent action bar shows the main menu of an AS application. Several icons are available to control the framework (cf. fig. 2).

The first icon from the right side is the power-off button. This icon closes the AS application.

The second button allows the user to inspect the machine captured context. It opens a dialog which shows the actual hierarchical graph of the captured sub- and super-contexts of the current situation.

The third button allows one to switch the user-interface to the "backside" presenting the actual adaptation opportunities of the AS application. The backside stands for the access to the "maintenance hatch" of the application, which means access to the adaptation and configuration of the application.

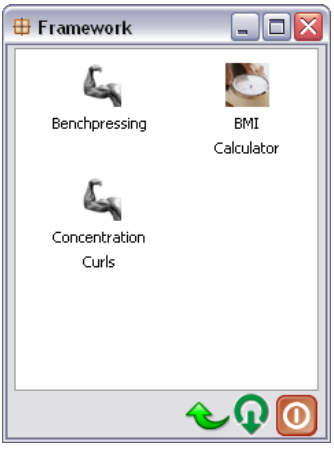

Fig. 3. The desktop with active services in the current context

Fig. 3 shows the initial graphical user interface of the Ambient Service application. It shows the available services for the actual contexts represented by the desktop symbols. If the context of the user changes the desktop will change accordingly. New available services will appear on the desktop and services, which got unavailable, will disappear. To consume a service, the user simply has to tip on the related symbol.

Fig. 4 shows the two different parts of the user interface after the complete flip to "backside" is done.

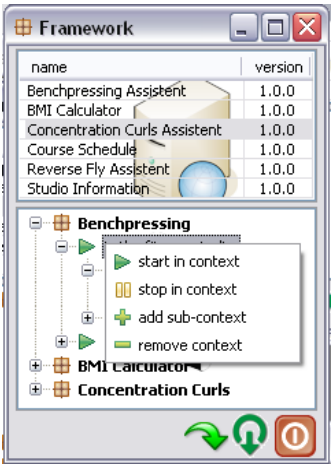

Fig. 4. The "backside" of the AS application, which is used for composition issues

The upper part of the "backside" shows a table representing all available AS components in the service component repository. In the lower half of the adaptation mode interface, those service components are displayed which are currently installed in the Ambient Service application. In the spanning tree underneath each Service-Component represents the contextual behaviour. It is the tree-representation of the content of the XML deployment descriptor.

We tested our prototype on a *context simulator*. We developed a small web application to substitute the context manager. With this application, we can simulate context changes, which generate output like the real context manager.

As the dummy context manager can not interpret sensor data we use a "Wizard of Oz" technique. A user will use the AS component framework as if there is a context manager. The human Wizard interprets the actual situation and uses this application to switch the context.

We use this technique to simulate a context change. i.e., moving from context A (e.g. benchpressing) to context B (e.g. going into refreshing area).

The Wizard of Oz recognizes this change in context and chooses context B in his web application. The information about change in context is immediately informed to the trigger engine. The Ambient Service component framework now has the necessary input to trigger context driven behavior of the installed service components.

5 Conclusion

Context-aware systems offer a great opportunity for handicapped people to perform those activities that they could not do under usual circumstances. . However, it is difficult for the software developers who are not in the same life situation like the handicapped people, to anticipate all the necessities of the physically handicapped people.

We put forward the idea of Shared Initiative, which brings the concepts of context-awareness and end user development together.

We have given a first outline of on architecture for Context-aware Adaptable System (CAAS) to address adaptivity and adaptability. In this architecture, the concept of mediation is the core issue.

In the EUD research project we have developed three prototypes based on the CAAS architecture. In this research paper we have described one of them, namely, Ambient Service. The prototype can adjust itself automatically to different contexts, and it allows for the end users to tailor the software according their needs.

We have primarily focused on the technical concepts of a Context-aware Adaptable System (CAAS). However, we believe that a real life testing is necessary to evaluate the context-awareness. Based on the experiences and comments made from the end users, such an evaluation will produce reliable results on the appropriation of context-awareness technology.

To conduct these tests, we have to realize a "Wizard of Oz" suite in order to simulate sensor behavior in an early stage of the design process with low cost (in respect to a proprietary sensor solution)

As a next step of our research, we will perform a formative evaluation on the Ambient Service prototype in a real world fitness setting. From there, we want to use the flexibility of a "Wizard of Oz" technique to experiment on different design concepts in cooperation with users in a Participatory Design manner.

References

1. Dey, A.K.: Understanding and Using Context. Personal and Ubiquitous Computing Journal. 5(1) (2001)
2. Dourish, P.: Seeking a Foundation for Context-Aware Computing. Human Computer Interaction. 16 (2001)
3. Want, R., Pering, T.: System Challenges for Ubiquitous & Pervasive Computing. In: ICSE 2005. LNCS, vol. 4309, Springer, Heidelberg (2006)
4. Dey, A.K., Mankoff, J.: Designing Mediation for Context-Aware Applications. ACM Transactions on Computer-Human Interaction (TOCHI). 12(1) (2005)
5. Davis, A.B., Moore, M.M., Storey, V.C.: Context-Aware Communication for Severely Disabled Users. CUU'03. Vancouver, British Columbia, Canada (2003)
6. Alm, N., Arnott, J.L., Newell, A.F.: Prediction and Conversational Momentum in an Augmentative Communication System. Communication System. 35 (1992)
7. Copestake, A.: Applying Natural Language Processing Techniques to Speech Prostheses. In: AAAI Fall Symposium on Developing Assistive Technology for People with Diabilities (1996)
8. Adams, L., Hunt, L., Moore, M. (2003). The "Aware System"-Prototyping an Augmentative Communication Interface. RESNA (2003)
9. Zajicek, M.: A Special Design Approach for Special People. In: Miesenberger, K., Klaus, J., Zagler, W., Burger, D. (eds.) ICCHP 2004. LNCS, vol. 3118, Springer, Heidelberg (2004)
10. Bellotti, V., Edwards, K.: Intelligibility and Accountability: Human Considerations in Context –Aware Systems. Human-Computer Interaction. 16 (2001)

11. Fischer, G.: Beyond 'Couch Potatoes': from Consumers to Designers and Active Contributors (2002), http://firstmonday.org/issues/issue7_12/fischer
12. Lieberman, H., Paterno, F., Wulf, V. (eds.): End User Development. Springer, Germany (2005)
13. Wulf, V.: Anpassbarkeit im Prozess Evolutionäre Systementwicklung. GMD-Spiegel. 3 (1994)
14. Lieberman, H. (ed.): Your Wish is My Command: Giving Users the Power to Instruct their Software. Morgen Kaufmann, San Francisco (2000)
15. Dey, A.K., Hamid, R., Beckmann, C., Li, I., Hsu, D.: A CAPpella: Programming By Demonstration of Context-Aware Applications. In: SIGCHI Conference on Human Factors in Computing Systems, CHI (2004)
16. Stevens, G., Wulf, V., Rohde, M., Zimmermann, A.: Ubiquitous Fitness Support Starts in Everyday's Context. In: The 6th World Conference The Engineering of Sport (2006)
17. Krogsæter, M., Oppermann, R., Thomas, C.: A user interface integrating adaptability and adaptivity. In: Oppermann, R. (ed.) Adaptive User Support: Ergonomic Design of Manually and Automatically Adaptable Software, Hillsdale, NJ, USA, Lawrence Erlbaum Associates, Mahwah (1994)
18. Morch, A., Stevens, G., Won, M., Klann, M., Dittreich, Y., Wulf, V.: Component-Based Technologies for End-User Development. Communication of the ACM. 47(9) (2004)
19. Stevens, G., Wulf, V.: A New Dimension in Access Control: Studying Maintenance Engineering Across Organizational Boundaries. (CSCW 2002) (2002)
20. Stiemerling, O.: Component-Based Tailorability. PhD thesis, Department of Computer Science, University of Bonn. (2000)
21. Zimmermann, A., Lorentz, A., Specht, M.: Applications of a Context-Management System. In: Dey, A.K., Kokinov, B., Leake, D.B., Turner, R. (eds.) CONTEXT 2005. LNCS (LNAI), vol. 3554, Springer, Heidelberg (2005)
22. Zimmermann, A., Specht, M., Lorentz, A.: Personalization and Context-Management User Modeling and User-Adapted Interaction. Journal of Personalization Research (UMUUAI) (2000)
23. Hallenberger, M.: Eine 3D Benutzerschnittstelle für komponentenbasierte Anpassbarkeit. Diplomarbeit, University of Bonn. (2000)
24. Szyperski, C.: Component Software: Beyond Object-Oriented Programming. Addison-Wesley, Reading (1998)