

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

Matthias Betz<sup>†</sup>, Mahmudul Huq<sup>†</sup>, Volkmar Pipek<sup>†</sup>, Markus Rohde<sup>†</sup>, Gunnar Stevens<sup>†</sup>, Roman Englert<sup>\*</sup>, Volker Wulf<sup>†</sup>

<sup>†</sup> 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}

<sup>\*</sup> 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 aspects, task context, collocation context, 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 the context-aware systems are often ambiguous. Although some of these ambiguities may be resolved using automated problem-solving techniques, many cases still requires 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.

For the many decisive aspects of ambiguous contexts it is the end user as a domain expert that has been identified as the key role to adapt context-aware behaviors. End users are also the key people to adapt necessary data sources and constraints. Thus, the end users and the domain experts should be able to define the appropriate adaptable behavior for their application.

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].

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].

This research paper is a part of a EUD research project started in 2004 to support ubiquitous fitness in the health/fitness area. In this project, we started with an ethnographical study of fitness and sports activities. This study lead to several application scenarios that should be addressed by a context-aware system [16].

This paper focus's on the technical aspects which builds a context-aware system that supports ubiquitous fitness, by presenting the architecture for mobile application.. The architecture explicitly addresses the idea of a combined approach for adaptable and adaptive services. 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.

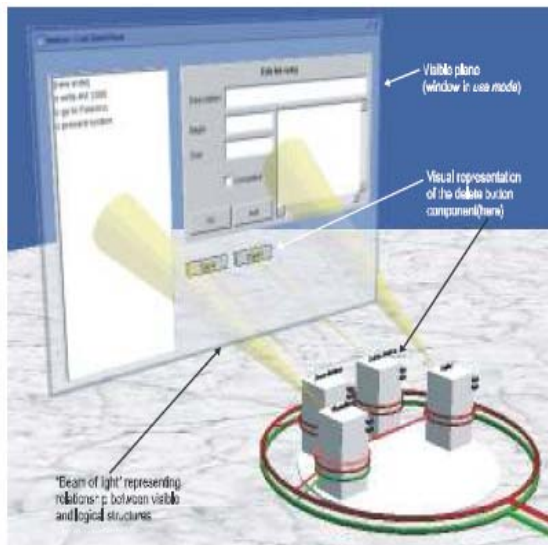
In the course of time, the ambient service may change. Firstly, the user develops himself as he uses the service and then performs adaptations to the service on his own. Secondly, the ambient service automatically recognizes changes in the environment and usage behavior or performance of the user and 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 change, 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].



**Fig. 1: 3D view of adapting components of a distributed application [23].**

Component based tailorability is an approach to transfer the connectional ideas of EUD to the field of component based systems [24]. According to O. Siemerling [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<sup>1</sup> 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 Basic assumptions

Assumptions take for granted that some information is fact. As a preposition of the software architecture, such assumptions need to be mentioned, in order to become aware of which topics are dealt with in this research paper and which topics are not parts of the discussion. In the following, five assumptions are described that are not regarded by this project.

- **Ubiquitous Computing Requirements**

The development of a CAAS application will take place in a ubiquitous computing environment. The CAAS software framework that will be used in the application development will address the requirements in ubiquitous computing.

The requirements are the following:

- Firstly, the environment must be a distributed environment. This type of distribution occurs in different levels. On a conceptual level the information is distributed and on an implementation level the system components are distributed.
- Secondly, the heterogeneity and interoperability of sensors, actuators, programming languages and operating systems exploited in a ubiquitous computing environment include various characteristics and capabilities that require consideration.
- Thirdly, a ubiquitous computing application needs to cope with many forms of spontaneity. The application must handle a dynamic environment and deal with breakdowns and errors. Robust systems continue to function in case of component malfunction, jam, disappear, restart, or inappropriate output.
- Fourthly, since information may refer to very personal and confidential aspects of people's

<sup>1</sup> Cf. <http://www.osgi.org>

lives, most people do not want this information to be entirely revealed to any other person. Embarrassing situations are avoided by the consideration of information security and privacy.

- **Static Communication Interfaces**

It is assumed that the interfaces for inter-service communication are static. In particular this means that the adaptation leaves the interface of the service unchanged. Only its internal functionality and behavior are adapted and the external appearance remains the same.

- **Expressive Service Description**

The expressiveness of the service includes the forming of precise, clean and general characterizations of service compositions and their representation in all the relevant aspects. Furthermore, it is assumed that formal context description has been made.

The situation description is the specification of certain conditions that need to be fulfilled in the specific situation. Basic specifications involve simple checks of values of the context data. To cope with the increasing complexity the formal specification is used. It allows defining levels of abstraction for basic values.

Based on the expressive situation descriptions it is possible to analyze the current context for triggering automatic adaptation of the system. It is also possible to inform the user about the actual situation and to provide context specific tailoring options.

- **Standard Context Ontology**

In order to maintain consistent context knowledge and to share that information among other systems and applications, a standard ontology is required. This context ontology determines how to represent context, so that it can be processed and reasoned by different implementations. Formally, ontology defines a shared vocabulary and associated semantic relations in a given language. It is an explicit specification of a domain for further development.

- **Hierarchal context specification**

In our work we follow a hierarchal context concept. This concept distinguishes between a situation and a context that characterize this situation. On a conceptual level, a situation is characterized by the following definition:

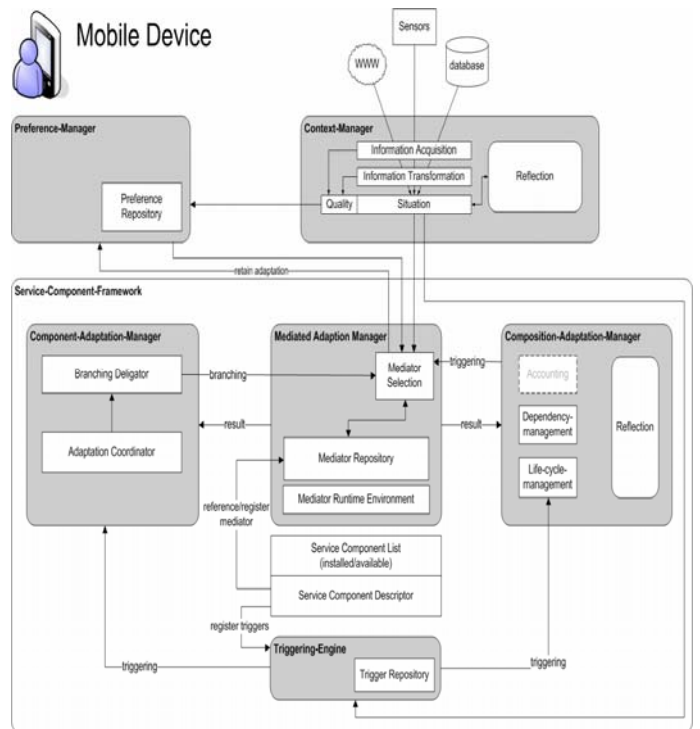
“A situation is defined by the set of all context specifications that hold in this actual situation.”

In a technical sense, the actual situation is described by more general or more fine grained context specification (e.g. being in a general fitness situation

and doing some bench pressing). As a result, an actual situation is defined by a dynamic graph of sub- and super-contexts. This dynamic graph is a sub-graph of the super graph defined by the standard context ontology.

### 3.2 System architecture

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



**Fig. 2: Software architecture for Context-Aware Adaptable System (CAAS)**

- **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.

Common context transformation techniques include fusion, derivation, aggregation and interpretation. 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.

- **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.

The reflection and transparency components provide feedback to the user in two ways, namely proactive and reactive. The proactive feedback is given when the user's expected event has occurred. The set of possible events are determined by the application developer. However, the user can specify which kind of events should trigger a feedback event.

- **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 adaptation manager component.

The composition adaptation manager is responsible for maintaining the state of each service component in the CAAS framework. It has to control the installation, starting, stopping and uninstalling of service components. A triggering engine activates these state transitions through a triggering mechanism. The composition management contains a life cycle manager

which maintains a list of available service components and their descriptions. Through this triggered composition management, it is possible to realize a component based adaptation initiated by either the system or by the end user in a shared way. The service composition can change in regards to the actual situation. The end user or the system must be enabled to trigger such state transitions to achieve composition based adaptation. The composition adaptation manager also contains a dependency manager for maintaining the interactions between the service components, which are connected by a dependency graph.

The component adaptation follows the breakpoint principle. Breakpoints can be placed anywhere in the code or sequence of a service component indicating the possibility or need for adaptation. At these points the execution is stopped and the mediator must delegate an execution. The mediated adaptation manager takes over the execution and fills the gap with appropriate mediators. After the completion of the mediation process, the result is given back to the main execution sequence.

- **Mediated Adaptation Manager**

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

During the mediated adaptation process, respective mediators are instantiated from the mediator repository and placed into the runtime environment. After the process is completed, the mediated adaptation manager returns the result to the initiating component. The mediated adaptation manager sends the results of the adaptation process to the preference manager. The preference manager then stores the results in the user preference repository. While sending the result to the preference manager, the mediated adaptation manager assigns the weighting values to the generated results, which shows the user preference to the result.

- **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. All context driven adaptations are registered on a rule base (a rule base structure?). The triggering engine constantly compares the current situation provided by the context manager with the registered situation abstractions. The triggering engine can trigger component based adaptations and state transitions according to the current composition and therefore enable a rule based starting and stopping of functionality represented by service components.

- **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<sup>2</sup>. It contains a view, presenting the main functionality to the user.

---

<sup>2</sup> 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>).

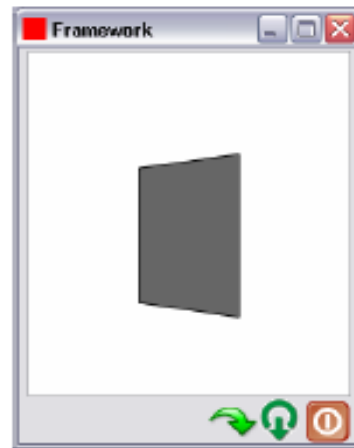


**Fig. 3: The action bar of the Service Component Framework**

An omnipresent action bar shows the main menu of an AS application. Several icons are available to control the framework (cf. fig. 3).

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.



**Fig. 4: Animation for the “Flip to backside” metaphor**

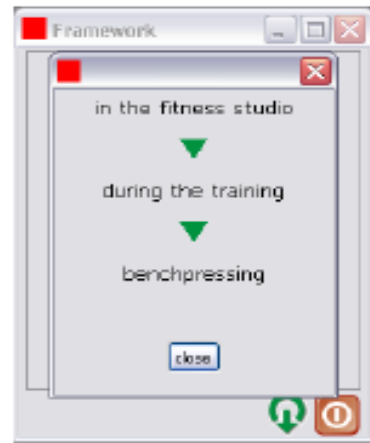
The third button allows one to switch the user-interface to the “backside” presenting the actual adaptation opportunities of the AS application. This solution is based on a mechanical clock metaphor, where the clock face is on the front side (*the use mode*) and the gears of the clock are on the backside (*the repair mode*). Therefore the backside stands for the access to the “maintenance hatch” of the application, which means access to the adaptation and configuration of the application.

In order to support the mental switch from using a function to adapt this function the flip to the “backside” is smoothly animated. Fig. 4 sketches the animation when the graphical-user-interface flips to the “backside”.



**Fig. 5: The desktop with active services in the current context**

Fig. 5 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. 7: Information about the current context**

Fig. 7 shows the dialog which reflects on the actual context. As mentioned above, the current context has a hierarchical structure. When the user is in the context “during the training”, he is also in the more general super-context “in the fitness studio” (fig. 7). However, at the same time the user is also in a more concrete sub-context “benchpressing” (fig. 7) mode.

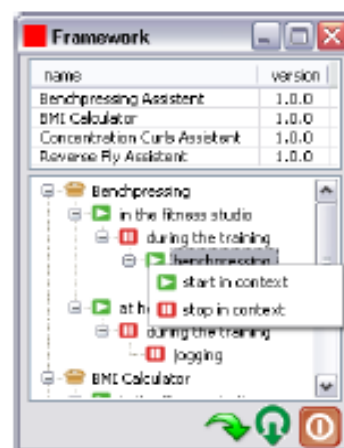
The most concrete context is printed at the leaf of the context hierarchy. The whole hierarchy is shown to let the user know how the most concrete context is derived as a sub-context of other contexts. This feedback supports the user to correct the context interpretation by the computer.

Fig. 8 shows the two different parts of the user interface after the complete flip to “backside” is done.



**Fig. 6: Usage of the personal digital trainer service**

Selecting “benchpressing” makes the GUI of the bench pressing service component available for the user. As shown in the fig. 6, the chosen service supports the user to perform effective bench pressing exercises.



**Fig. 8: 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, service components are displayed which are currently installed in the Ambient Service application. The spanning tree underneath each Service-Component represents the contextual behaviour. It is the tree-representation of the content of the XML deployment descriptor.

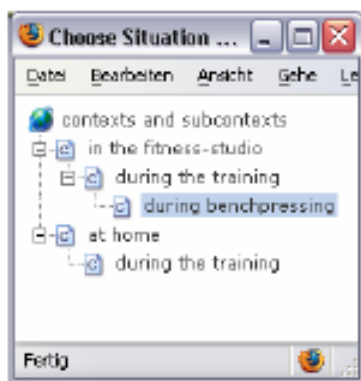
These two parts realize the Composition Adaptation Management. It is possible to drag and drop service components from the repository into the AS container in order to download and install the service component on the mobile device. It is also possible to drag and drop installed service components from the framework to the repository to perform the un-installation of the service components. After uninstalling all the service components, they will not be available for accessing in the AS application, even in the right context.

If a service component is installed in the framework the user can use this service in specific situation. For this he has to specify the contextual behaviour. He can add or remove add new contexts to a service component via drag and drop the service in the actual context graph.

These personal adaptations are retained on the device so that it is possible to restore the personal settings, even after a complete reinstallation of a service component. The vendor settings are kept in the descriptor so that they are available for resetting a service component's contextual behaviour.

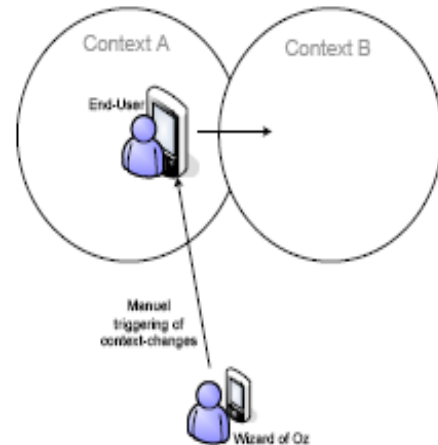
We tested our prototype on a *context simulator*. We developed a small web application to substitute the context manager. With this application, we can simulate context changes, which generate output like the real context manager.

Fig. 9 shows a screenshot of this browser based web application.



**Fig. 9: Screenshot of the dummy context manager**

As the dummy context manager can not interpret sensor data we use a “Wizard of Oz technique”.



**Fig. 10: “Wizard of Oz” concept for substitution of the Context Manager**

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. Figure 10 is illustrates this principle.

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.

The majority of the research on context-aware systems focuses only on the aspect improving the context capturing algorithm. They do not try to empower the users to understand such systems. However, to cope with problems such as unreliable or ambiguous context information, it is necessary to put users in the control of service selection, composition and execution.

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 an architecture for Context-aware Adaptable System (CAAS) to address adaptivity and adaptability. In this architecture, the concept of mediation is the core issue. Mediation is extremely necessary in order to build a suitable system for a ubiquitous and pervasive computing environment that can be controlled by users.

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 to 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.

## 6. References

- [1] Dey, A. K. (2001). *Understanding and Using Context*. Personal and Ubiquitous Computing Journal. 5(1).
- [2] Dourish, P. (2001). *Seeking a Foundation for Context-Aware Computing*. Human Computer Interaction. 16.
- [3] Want, R., Pering, T. (2005). *System Challenges for Ubiquitous & Pervasive Computing*. ICSE'05. St. Louis, Missouri, USA.
- [4] Dey, A. K., Mankoff, J. (2005). *Designing Mediation for Context-Aware Applications*. ACM Transactions on Computer-Human Interaction (TOCHI). 12(1).
- [5] Davis, A. B., Moore, M. M., Storey, V. C. (2003). *Context-Aware Communication for Severely Disabled Users*. CUU'03. Vancouver, British Columbia, Canada.
- [6] Alm, N., Arnott, J. L., Newell, A. F. (1992). *Prediction and Conversational Momentum in an Augmentative Communication System*. Communication System. 35.
- [7] Copestake, A. (1996). *Applying Natural Language Processing Techniques to Speech Prostheses*. AAI Fall Symposium on Developing Assistive Technology for People with Disabilities.
- [8] Adams, L., Hunt, L., Moore, M. (2003). *The “Aware System”-Prototyping an Augmentative Communication Interface*. RESNA 2003.
- [9] Zajicek, M. (2004). *A Special Design Approach for Special People*. K. Miesenberger et al. (Eds.): ICCHP 2004, LNCS 3118.
- [10] Bellotti, V., Edwards, K. (2001). *Intelligibility and Accountability: Human Considerations in Context –Aware Systems*. Human-Computer Interaction. 16.
- [11] Fischer, G. (2002). *Beyond ‘Couch Potatoes’: from Consumers to Designers and Active Contributors*. [http://firstmonday.org/issues/issue7\\_12/fischer/](http://firstmonday.org/issues/issue7_12/fischer/)
- [12] Lieberman, H., Paterno, F., Wulf, V. (Eds.). (2005). *End User Development*. Springer. Germany.
- [13] Wulf, V. (1994). *Anpassbarkeit im Prozess Evolutionäre Systementwicklung*. GMD-Spiegel. 3.
- [14] Lieberman, H. (Eds.). (2000). *Your Wish is My Command: Giving Users the Power to Instruct their Software*. Morgen Kaufmann.
- [15] Dey, A. K., Hamid, R., Beckmann, C., Li, I, Hsu, D. (2004). *A CAPpella: Programming By Demonstration of Context-Aware Applications*. SIGCHI Conference on Human Factors in Computing Systems, CHI.
- [16] Stevens, G., Wulf, V., Rohde, M.,

- Zimmermann, A. (2006). *Ubiquitous Fitness Support Starts in Everyday's Context*. The 6<sup>th</sup> World Conference "The Engineering of Sport".
- [17] Krogsæter, M., Oppermann, R., and Thomas, C. (1994). *A user interface integrating adaptability and adaptivity*. R. Oppermann (Ed.): Adaptive User Support: Ergonomic Design of Manually and Automatically Adaptable Software. Lawrence Erlbaum Associates, Hillsdale, NJ, USA.
- [18] Morch, A., Stevens, G., Won, M., Klann, M., Dittreich, Y., Wulf, V. (2004). *Component-Based Technologies for End-User Development*. Communication of the ACM. 47(9).
- [19] Stevens, G., Wulf, V. (2002). *A New Dimension in Access Control: Studying Maintenance Engineering Across Organizational Boundaries*. CSCW 2002.
- [20] Stimmerling, O. (2000). *Component-Based Tailorability*. PhD thesis, Department of Computer Science, University of Bonn.
- [21] Zimmermann, A., Lorentz, A., Specht, M. (2005). *Applications of a Context-Management System*. CONTEXT 2005.
- [22] Zimmermann, A., Specht, M., Lorentz, A. (2000). *Personalization and Context-Management User Modeling and User-Adapted Interaction*. Journal of Personalization Research (UMUUI).
- [23] Hallenberger, M. (2000). *Eine 3D Benutzerschnittstelle für komponentenbasierte Anpassbarkeit*. Diplomarbeit, University of Bonn.
- [24] Szyperski, C. (1998). *Component Software: Beyond Object-Oriented Programming*. Addison-Wesley.