An Agent-Based Architecture for Ubiquitous Multimodal User Interfaces

Andreas Rieger, Richard Cissée, Sebastian Feuerstack, Jens Wohltorf, Prof. Dr. Sahin Albayrak
DAI-Labor[1]
Technische Universität Berlin
Berlin, Germany
{andreas.rieger, richard.cissee, sebastian.feuerstack, jens.wohltorf, sahin.albayrak}@dai-labor.de

Abstract

Today's mobile world is made up of heterogeneous networks combined with various devices with different characteristics. Progressive users have the desire to utilize the same services and access the same information content on all their available devices. Therefore, an efficient way of providing user interfaces for multiple devices and their different modalities is needed. Current service development architectures, however, do not provide an easy approach for creating multimodal interfaces on different devices. Our agent-based architecture, the Multi-Access Service Platform, offers a flexible solution for ubiquitous service access. Using an abstract description of content and interactions, it is capable of generating user interfaces in languages such as HTML, WML, and VoiceXML. Therefore, our solution allows a heterogeneous landscape of devices to access an application server's mobile services. The paper concludes with the description of a mobile service we have implemented based on the Multi-Access Service Platform.

1. Introduction

The emerging availability of mobile services offers great new possibilities and challenges for service users and service providers: The service users get the chance to access a new category of services offering new possibilities. These services are accessed and used differently than traditional services. The service providers creating value-added mobile services face high development efforts because they have to adapt services to new devices and modalities.

Novel technology is required for adding value to and improving existing mobile services. Today's users are already networked over a broad range of second to third generation mobile phones that exhibit a wide spectrum of capabilities and basic qualities. While PDAs, for example, have relatively large color displays and powerful processors, mobile phones are constrained to significantly less display area, storage space and processing power. It is therefore a challenge for service providers to support as many different devices as possible, but a task necessary for service providers who want to remain competitive. Consequently, the cost of maintaining and adjusting content and developing user interfaces for a mobile portal is increasing significantly. Contrary to interfaces designed for stationary desktop PCs, graphics, text and other media must be adapted for each class of devices. As device independency is only the minimum requirement for mobile services, users soon will expect voice interfaces as well as the ability to switch between arbitrary devices and modalities that are most comfortable to them in a specific situation.

The growing availability of new devices with improved input and output capabilities will allow a more transparent, human-behavior oriented way of human-computer interactions. Multimodal user interfaces allow services to be used via different modalities. Based on his preferences, the user can chose the most suitable modality for his current context. Multimodal user interfaces are expected to be easier to use and may be preferred by many users over existing graphical user interfaces. Additionally, they have the potential to address new usage scenarios.

Many future applications will feature both multimodal interaction and device independent access in combination. Mobile multimedia, in-car information and entertainment as well as home automation and control are only a few scenarios where multimodality and device-independence complement each other beneficially. To simplify the service usage, mechanisms for single sign on and session management should also be integrated. Our work is motivated by the vision that future user interface technologies must therefore support all these aspects in a single, integrated solution.

The paper is structured as follows: The following chapter describes the problems arising from making services accessible through different devices and modalities. Chapter 3 lists requirements for architectures addressing these problems. Chapter 4 describes our realized architecture for solving these problems. Chapter 5 describes an application, currently deployed as a prototype, which has been realized based

[1] This project has been funded by the German Federal Ministry of Research and Education (BMBF) under Grant No. 01AK037.

on our architecture. Chapter 6 describes related work. Finally, Chapter 7 concludes the paper with a short evaluation of our approach and outlines future work.

2. Problem Description

Today's mobile world consists of heterogeneous networks, ranging from high-speed Wireless-LAN networks, to slower GPRS connection and serveral networks in-between. Not every network offers a full coverage, so in many situations it may be necessary to switch devices in order to continue the service usage.

Supporting different devices and modalities is a huge challenge for service providers. Unfortunately, there are big differences between the capabilities of mobile devices. For example, a notebook with WLAN connection offers greater audio and video capabilities than a mobile phone running with a GPRS connection.

Because of the large number of different devices used by mobile users, service providers should strive to make their services accessible for as many mobile devices as possible; otherwise, they might lose their customers. Integrating different modalities pushes the number of service access methods to a new level. However, supporting many different devices and modalities usually leads to a huge development effort, because user interfaces for each device or class of devices have to be developed separately. This procedure leads to an explosion of costs, which is obviously to be avoided by service providers.

Therefore, an architecture is needed that upgrades existing mobile services for new devices and modalities, with as little effort for the service provider as possible.

3. Requirements

To overcome the problems described above, an architecture for creating multimodal user interfaces should fulfill the following basic requirements.

- ***Model-Based Approach***: User interfaces are represented by a single abstract model. The actual graphical, voice or multimodal user interfaces are derived from this abstract model.
- ***Device dependent UI generation***: User interfaces are generated during runtime. Users should be able to change their modalities of interaction as well as their devices on the fly, i.e. during an ongoing session. The generation process should be adaptive, based on the device profile.
- ***Simplified UI creation process***: Most user interface designers are not used to thinking in an abstract user interface language, because they are more used to thinking in terms of the look and feel of concrete user interfaces. Therefore, powerful tools should support the developers in every stage of development, supporting the new concept of creating UIs.
- ***Provision of seamless services:*** Users should be allowed to switch between the devices and modalities as they like. Ongoing service sessions can be stopped, stored persistently and continued seamlessly exactly at the same point on an arbitrary modality or device.
- ***Automated sign-on:*** To increase the usability, authentication information should be provided by the user's personal devices. Thus, the user is only required to sign on once and join different sessions later on.

4. Approach

Our approach complies with all requirements stated in Chapter 3. We have successfully built a Multi-Access Service Platform (MASP), which allows the creation of user interfaces for different modalities. Figure 1 gives a basic overview of the MASP-related layers, which are described in detail in the following sections.

4.1. AIDL

Each service's content has to be described in an abstract model using a language called *Abstract Interaction Description Language* (AIDL). While the MASP itself implements the Controller layer of the Model-View-Controller design pattern, the AIDL language realizes the View layer, as depicted in Figure 1.

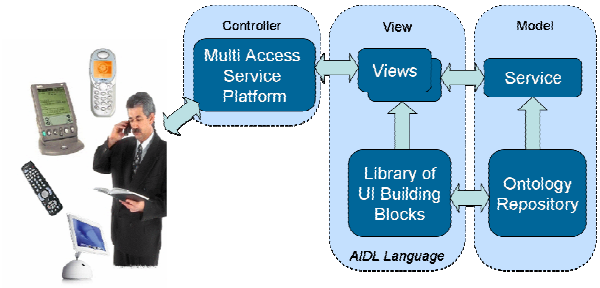

Figure 1 - Layer Model

Concrete graphical, voice or multimodal user interfaces are derived from this single model. It is necessary to mark the service's input and output information (i.e. the content) most relevant to the user. This process can be done for every modality or device class to distinguish the amount and order of information the MASP presents to the user. Learning yet another new language is not necessary, since AIDL is completely encapsulated by a tool we provide to add these information.

A fast and efficient way of user interface generation is supported by the platform: AIDL is an XML-based language describing all relevant input and output parameters of a user interface. As soon as an AIDL description exists, each service is immediately usable via all devices. Intelligent general renderings based on

XSL transformations provide first mockups for each device. Thus, the user interface designer's task is no longer to design user interfaces from scratch, but to continuously refine the automatically generated interfaces.

The AIDL language supports linking user interface elements by referring to categories of ontologies. Expressing semantics with the help of ontologies is an established approach in the Semantic Web community [1]. This approach enables the MASP to organize a library of user interface building blocks and utilize them within the automatic mockup creation.

The three basic language elements of AIDL are "data", "group", and "scenario". Data elements describe primitive types such as character strings, numbers or links to another service (URLs). The group type is used to make statements about a list of elements. Using groups allows setting the order of content elements, which is very important when using a voice interface. A "scenario" joins a list of groups and describes the relationships between them, such as ordering, orientation, and constraints regarding size and format of group and data instances. Finally, a scenario results in a user interface that is rendered on the user's device.

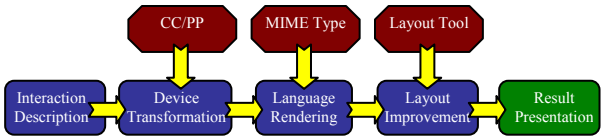

Figure 2 – Rendering Process

A fundamental benefit of using AIDL is the adoption of ontologies to describe user interface building blocks. Groups are used to select relevant attributes of a category and are automatically associated to user interface building blocks to improve the generic rendering. This helps the MASP to identify the information that is most important to the user in a specific situation and to hide irrelevant information. AIDL groups are similar to the classical views concept of a database and ontologies are used in a way similar to tables, because they are used to store the domain knowledge (the content) of a service.

4.2. Architecture

The Implementation of the Multi-Access Service Platform [4] has been carried out based on the Foundation for Intelligent Physical Agents (FIPA)-compliant [5] Multi-Agent System (MAS) architecture "Java Intelligent Agent Componentware" (JIAC) [1][3]. JIAC integrates fundamental aspects of autonomous agents regarding pro-activeness, intelligence, communication capabilities and mobility by providing a scalable component-based architecture. Additionally, JIAC offers components realizing management and security functionality, and provides a methodology for Agent-Oriented Software Engineering (AOSE). The main features of applications realized with the JIAC MAS architecture are:

- ***Modularity:*** MAS-based applications are mainly configured by selecting and defining the participating agents. Therefore, different modules made up by groups of agents may be changed easily.
- ***Scalability:*** Scalability is mainly achieved by duplicated the agents responsible for critical tasks, thus distributing the load between multiple identical agents and removing bottlenecks.
- ***Adaptability:*** MAS-based applications may be reconfigured at runtime, i.e. agents may be added or removed to adapt the functionality provided. The newly offered services may be used immediately.
- ***Distributedness:*** Mobile agents have the ability to migrate between platforms that may be located on different servers.

The Multi-Access Service Platform itself is distributed in two different versions: The first version is an enterprise- ready distribution that uses a web archive file allows the deployment within J2EE-based application servers. The second option is a micro edition that uses an internal web server to answer service requests and is small enough to run in embedded environments such as home gateways.

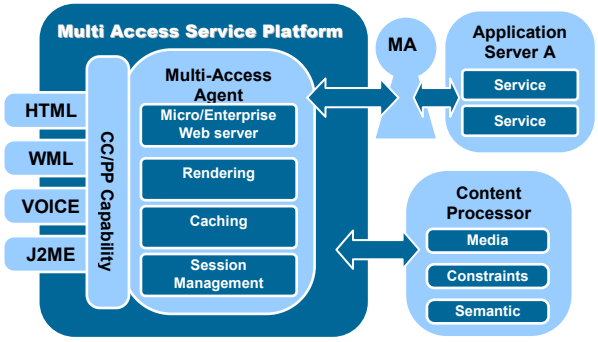

Figure 3 – MASP Architecture

The architecture of the platform is divided into three main roles, as shown in Figure 3:

- The Multi-Access Agent (MAA) receives and answers service requests from the user's device. The MAA is the main access point for all user connections to the system.
- The Content Processor is responsible for processing all multimedia content. Under the precondition that all multimedia content contains semantic description attributes, the Content Processor can deliver the optimal presentation according to the user's device.
- The Mediative Agent (MA) is a broker connecting the MAA and the services offered by the application server.

The MASP uses the Java Authentication and Authorization Service (JAAS) to enable services to authenticate users and to enforce access controls mechanisms. Currently an operating system login module as well as a device independent login module

is provided. The latter is directly connected to a user management directory of an LDAP server to authenticate user information. Since JAAS implements the pluggable authentication module (PAM) framework, other authentication modules can easily be added. As shown in Figure 4, each PAM module can directly connect to an LDAP server to authenticate user information. A module that connects to the Sun Java Systems Access Manager in order to provide single sign-on features is currently being developed.

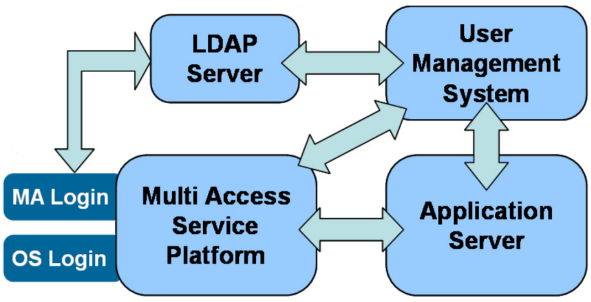

Figure 4 - MASP Authentification

After the login process both the MASP and the Application Server may ask the user management for the user's authorization to use the service. Authorization is supported by attaching a service control list (SCL) to each individual service offered by a service provider. A SCL contains rules for allowing or denying access to a service. These rules can be defined freely for any information available at the time a service is requested. Currently, predefined rules for JAAS subjects created during the login and rules for X.509 certificates that are exchanged in SSL connections are available. A SCL may be created or modified by using a security policy editor.

4.3. Session Management

The MASP further enhances multimodal interactions by providing mechanisms for intelligent session management. It allows the freezing of a running service session that may subsequently be resumed seamlessly via arbitrary other modalities. Thus, the user may switch seamlessly between devices without any data loss or synchronization needs. A running service can be stopped immediately for any reason and continued later on. The session management integrates well with the authentication and authorization layer of the MASP because a list of sessions is kept for each user, and mechanisms to switch between these sessions are provided immediately after the user has logged on.

We distinguish between access sessions and service sessions: The access session is established between the device and the MASP as soon as the user connects and logs in, whereas the service session identifies a service used by the user.

The access session provides information about the device used by the user and information about the context of the service usage. It is associated with a user object, which also stores additional (persistent) information about the user, such as a list of known devices or disabilities, within a user profile. This enables the MASP to render user interfaces for specific devices and to consider additional information during this rendering process. A user can have several active access sessions at the same time, which enables him to use different services at the same time.

The service session provides information about the service usage state. It enables the MASP to control a user's service usages, allowing the user to freeze and resume services, switch between active sessions or even change the device during service usage. Service sessions can also be continued if a user logs off and logs back on later.

The session management is based on the user management and the authentication mechanisms described above which enable the MASP to manage a list of users and to store additional data for each of these users. The user management also provides the prerequisites needed to store a list of access sessions for each user. Each of the stored access sessions is itself associated with a list of service sessions stored in a service history. The combination of service and access sessions allows control of the presence of a user using the access session, user information management and the service usage control.

4.4. User Interface Adaptation

The MASP uses CC/PP-based [6] device capability detection by using a framework specified by the W3C organization for the detection of device capabilities at runtime. Using CC/PP enables us to optimize the automatic user interface generation by taking screen size, supported media formats and user preferences into consideration.

The following example explains the basic features of the process adapting user interfaces by the MASP: Figure 5 shows the runtime optimization of a service portal user interface. The scenario "Select Service" has been defined in AIDL, consisting of a primitive data type describing the current state of the interaction ("I can offer you the following services:") and a group ("available services"). This group is constrained to be rendered in a circular layout.

If the user connects to the portal service, the MASP detects the device capabilities with the help of CC/PP. In our example, the screen width is 800 pixels, whereas the AIDL-based scenario asks for an optimal size of 600 pixels (indicated by the "Media Group" parameters in the figure). The preferred attribute indicates to the MASP whether the rendered user interface should be as near as possible to the optimum value, or as far away as possible from a minimal value.

Based on the screen and group constraints, the parameters for single images ("Multimedia data") are adjusted: Due to the circular layout of the group, the width of the single images is preferred to be close to

100 pixels. The maximal width is limited by the maximal width of the superior element, i.e. the group.

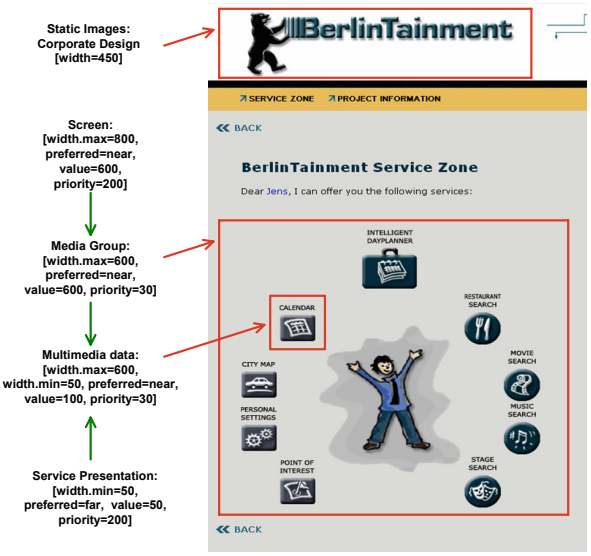

Figure 5 – User Interface Adoption

Apart from constraints that are derived from the user's device capabilities, there is another option for constraint definition from the service's point of view. In our example, a minimum size constraint is given by the "Service Representation" parameters, requiring each single service image to be at least 50 pixels in width. Specifying constraints on the service level makes sense if a service is targeting specific audiences such as people with disabilities who prefer larger text and images.

5. BerlinTainment

Based on the MASP architecture, we have developed an application called BerlinTainment, which utilizes all features of the MASP to create a device independent and multimodal mobile service.

The BerlinTainment project [7][8] focuses on the realization of a scalable Serviceware Framework based on multi-agent technology. The framework is utilized for the development of context-aware services, i.e. services providing personalized, location-based information. As a showcase for the functionality of the Serviceware Framework, different services in the entertainment domain have been developed and integrated into the BerlinTainment demonstrator.

Using the MASP developers were able to focus on creating the different services comprising the BerlinTainment demonstrator, rather than creating user interfaces for each different device. It turned out that the first mockup of user interfaces on different devices created by the MASP were usable and had only to be reworked slightly with layout and graphical information.

The implemented system benefits from using the Multi-Access Service Platform, which enables it to extend its services so that they can be used in a ubiquitous and seamless way. Examples of supported devices and modalities include mobile phones via WML-based user interfaces and telephones via voice-based interfaces. In each case, the MASP adds value to BerlinTainment for mobile users by letting them switch between modalities, as they like. Figure 6 shows two WML screenshots of the BerlinTainment demonstrator. The left screenshot shows the portal service, the HTML interface of which is shown in Figure 5. Note that the images of the HTML version have been reduced to textual links due to display size restrictions of the device. The right screenshot shows a dialog in which the image size has been automatically adapted to the constraints of the device.

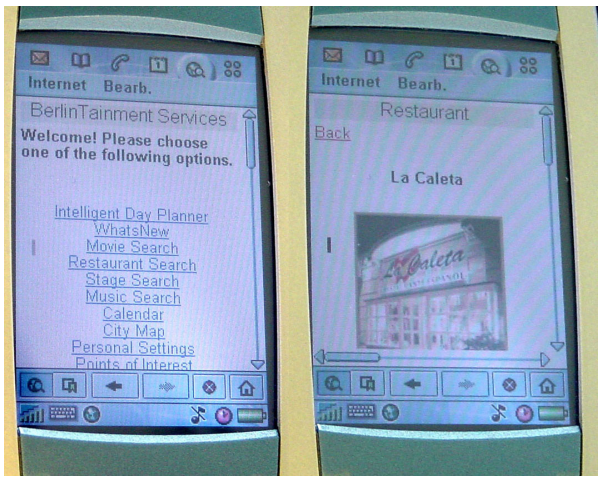

Figure 6 - BerlinTainment Screen in WML

Using BerlinTainment, a tourist visiting Berlin is assisted by a set of services including a restaurant, theater, concert and movie finder, a calendar and a routing service. An additional service, the Intelligent Day Planner, integrates the different services, allowing the user to schedule his activities for a given day and to receive personalized and location-based recommendations for each activity, such as restaurant, cinema or theater visits. Based on these recommendations, the user is given the possibility to make reservations for the various activities and plan his route between the different locations.

A frequent usage scenario highlighting the MASP-related features of BerlinTainment is using the "Intelligent Dayplanner" to generate suggestions on how to spend the leisure time. Most users prefer to start BerlinTainment on their stationary personal computer, because using a browser-based interface is more comfortable for editing user preferences. After BerlinTainment has made a suitable recommendation, the user may freeze the service and leave for the location of the recommended event. While traveling to the location, the user may want to obtain precise directions. This may be achieved by dialing the phone

number of the BerlinTainment system with an in-car or mobile phone and selecting the location-based routing services by voice control. Reaching the location, the BerlinTainment user may finally continue the frozen service session in order to look at the details of the recommendation, reschedule suggestions or review the visited location by using a WML-enabled smart phone.

6. Related Work

Driven by the objective of device independency, UIML [9] proposes an XML language for device independent user interface design. UIML is not meant for on-the-fly generation of UIs but for a consistent UI development across different platforms. The UI description can be converted into other languages like Java or HTML.

Recent projects in the HCI community focus on generation of UI for different devices. The Personal Universal Controller (PUC) [10] is confined to the remote control domain and can generate UIs to control home appliances like VCRs, stereo equipment, ovens etc. that often comes with a remote control. The PUC employs an abstract interaction description that describes manipulable elements of appliances as state variables.

In [11] the UI generation in PUC is enhanced with additional semantic information about the controlled device to improve the quality of the UI. This is the only related approach we know of where semantic information about the service usage is used to improve usability.

7. Conclusion and Future Work

We have presented an architecture that allows the fast and easy creation of multimodal user interfaces based on an abstract description language. Additionally, an application, which made use of this architecture, has been described. By realizing this application based on the MASP, we have shown that it is in fact possible to create a multimodal and device independent mobile service with less development effort compared to approaches based on traditional UI design methods.

The emergence of multimodal interfaces, as presented in this paper, represents only the beginning of a progression toward computational interfaces capable of human-like interactions. Such interfaces might use context information from a large number of different input devices and sensors in order to support intelligent adaptation to the user's current situation and usage environment. Integration of context information could lead to an automated and intelligent selection of input and output devices and methods. Future adaptive multimodal context aware services have the potential to provide new functionality, to be easier to use, and to react flexibly as a multifunctional and personalized mobile system.

8. References

[1] James Hendler, Tim Berners-Lee and Eric Miller: Integrating Applications on the Semantic Web. Journal of the Institute of Electrical Engineers of Japan, Vol 122(10), October, 2002, p. 676-680.

[2] Stefan Fricke, Karsten Bsufka, Jan Keiser, Torge Schmidt, Ralf Sesseler, Sahin Albayrak: Agent-based Telematic Services and Telecom Applications. Communications of the ACM, April 2001

[3] Ralf Sesseler, Sahin Albayrak: Service-ware Framework for Developing 3G Mobile Services. The Sixteenth International Symposium on Computer and Information Sciences, ICSIS XVI, 2001

[4] Joos-Hendrik Böse, Sebastian Feuerstack: Adaptive User Interfaces for Ubiquitous Access To Agent-based Services. Workshop on Human-Agent Interaction, Agentcities ID3, Barcelona, Spain February 2003

[5] FIPA Agent Management Specification, FIPA00023, 2002, http://www.fipa.org/specs/fipa00023

[6] W3C Recommendation: Composite Capability/Preference Profiles (CC/PP): Structure and Vocabularies 1.0, 15 January 2004, http://www.w3.org/TR/2004/REC-CCPP-struct-vocab-20040115/ (28 June 2004)

[7] Jens Wohltorf, Richard Cissée, Andreas Rieger, Heiko Scheunemann: BerlinTainment - An Agent-Based Serviceware Framework for Context-Aware Services. Proceedings of 1st International Symposium on Wireless Communication Systems, ISWCS 2004

[8] Jens Wohltof, Richard Cissée, Andreas Rieger, Heiko Scheunemann: BerlinTainment - An Agent-Based Serviceware Framework for Ubiquitous Context-Aware Services, AAMAS 2004 demonstration, New York, USA

[9] Constantinos Phanouriou: Uiml: A device-independent user interface markup language, Ph.D. thesis, Virginia Polytechnic Institute ans State University, September 2000.

[10] Jeffrey Nichols, Brad A. Myers, Michael Higgins, Joseph Hughes, Thomas K. Harris, Roni Rosenfeld, and Mathilde Pignol: Generating remote control interfaces for complex appliances, Proceedings of the 15th annual ACM symposium on User interface software and technology, ACM Press, 2002, pp. 161–170.

[11] Jeffrey Nichols, Brad A. Myers, and Kevin Litwack, Improving automatic interface generation with smart templates, Proceedings of the 9th international conference on Intelligent user interface, ACM Press, 2004, pp. 286–288.