BerlinTainment - An Agent-Based Serviceware Framework for Context-Aware Services

Jens Wohltorf, Richard Cissée, Andreas Rieger, Heiko Scheunemann
DAI-Labor
Technische Universität Berlin
Berlin, Germany
{Jens.Wohltorf, Richard.Cissee, Andreas.Rieger, Heiko.Scheunemann}@dai-labor.de

***Abstract*—In the near future, providers of mobile services will face increasing competition. Therefore, the ability to design, develop and deploy reliable context-aware services fast and easily will become essential. We introduce an agent-based Serviceware Framework assisting service providers in developing innovative services, thus reducing the time-to-market of the respective applications. The realized Serviceware Framework offers personalization, location awareness and device independence within each single service. We describe the utilization of the different modules of the framework as well as prototypical application services in the entertainment domain based on the framework developed within the BerlinTainment project.**

Serviceware Framework, multi-agent systems, mobile services, session management, device independence, personalization, context-awareness

I. Introduction

The ability to design, develop and deploy reliable context-aware services efficiently is becoming of central importance for providers of mobile services, who face increasing competition in the telecommunications market. In this paper, we introduce an agent-based Serviceware Framework supporting the development of context-aware services and describe prototypical services developed on the basis of the framework. There are several related projects in this area focusing on specific aspects of context-aware services, but our approach is novel in the sense that it combines functionality from different areas and provides additional services integrating information from various domains.

This paper is structured as follows: The following chapter describes the problems arising from developing context-aware services. Chapter III lists requirements of frameworks addressing these problems. Chapters IV and V describe our approach and implementation of an agent-based framework for ubiquitous context-aware services, with a special focus on reusability. Chapter VI describes related work. Chapter VII concludes the paper and outlines further work.

II. Problem Description

Currently, it is infeasible to develop context-aware services and applications efficiently, mainly because in addition to the service itself a large overhead of functionality is required, as shown in Figure 1:

- Context-aware services require solutions for three main issues in mobile computing: Ubiquitous and device-independent access, provision of personalized information and location-based services (LBS) functionality *(I)*;
- Additionally, infrastructure functionality for user management, AAA (authentication, authorization, accounting) and additional management tasks is required *(II)*;
- Finally, to increase the usability and value of the context-aware service, further secondary services offering assistance to the user e.g. in the form of calendar or community services may be required to add value to the overall application *(III)*.

Consider the following use case: The user wants to receive recommendations of restaurants, based on his current location as well as his user profile containing his preferences regarding cuisine, district, price range, etc. He intends to interact with the respective service via different technologies, such as mobile phones, PDAs, PCs and speech. Additionally, the user would prefer to use this service seamlessly on different devices, without having to restart the interaction on a new device.

Developing a service based on these requirements may be split up into developing the service itself (in this case, mainly

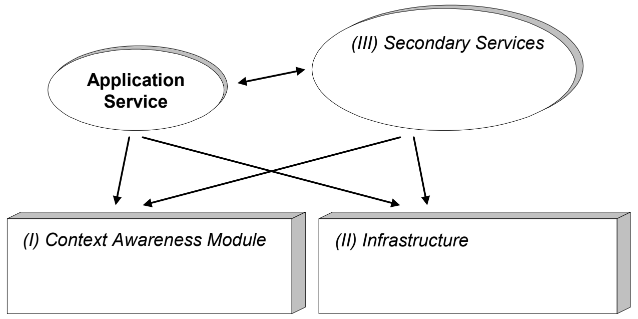


Figure 1. Functionality required in addition to a context-aware application service itself. The arrows indicate dependencies between the modules.

This work has been funded by the German Federal Ministry of Education and Research (BMBF).

the user interaction and the management of restaurant information) and developing the required overhead functionality, as described above.

In this scenario as well as in many other cases, developing all needed functionality is obviously not very efficient because of the time required and the costs involved. It may be noticed, however, that different scenarios require largely the same overhead functionality, i.e. the given use case may change, e.g. from recommendations of restaurants to a shopping guide, while the additional functionality and secondary services remain the same.

III. REQUIREMENTS

To overcome the problems described above, a framework-oriented approach integrating a toolkit for context-aware services, a framework providing infrastructure functionality and services providing secondary functionality is required. Using such a framework, the developer may concentrate on the context aware-service itself, without having to go into many details of the overhead functionality.

A framework for context-aware services has the following additional requirements:

- *Modularity:* Typically, not all functionality is required in any given scenario. It should be possible to use only those parts of the framework that are actually required.
- *Scalability:* The framework should be usable for small, non-public systems as well as for applications with a large target audience.
- *Adaptability:* The functionality provided by the framework should not be static, or it would be outdated soon. Therefore it should be possible to add, remove, or replace parts of the framework, even within deployed systems during runtime, without requiring changes on the framework user's side.
- *Distributedness:* It should be possible to distribute the framework e.g. in order to enable load balancing, or to increase the overall security.
- *Security:* The framework should contain various security features to prevent unauthorized use of its functionality and resources, and to protect systems based on the framework against attacks.
- *Privacy:* In order to enhance the acceptance and trustworthiness of the respective systems from the intended users' point of view, privacy-enhancing technologies should be built into the framework.

All of these requirements should be met by a framework aiming at providing support not only for specific, but for any kind of context-aware services.

IV. APPROACH

Our approach of a Serviceware Framework for context-aware services is based on Multi-Agent System (MAS) technology. For an introduction to MAS technology, we refer to [1]. Basically, MAS architectures consist of agents encapsulating specific functionality and offering services to exchange information with other agents. All agents exist in a specific environments, the *agent platforms*. The interaction of agents is based on *ontologies* defining a common vocabulary. MAS architectures are especially suitable for realizing frameworks for context-aware services, because they fulfil the requirements given in the previous section:

- *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 which may be located on different servers, e.g. for reasons of load-balancing.
- *Security:* MAS architectures include security features preventing unauthorized service usage and prohibiting agents from attacking other agents or platforms.
- *Privacy:* Privacy aspects regarding personal information are addressed by encapsulating sensitive information within dedicated personal user agents.

There are other approaches for distributed systems addressing these requirements similarly. The OSA/ Parlay approach [8] has its main focus on supporting the design and development of mobile applications by offering standardized interfaces for many services offered by mobile networks. Jini network technology [9] is an open architecture that enables developers to create network-centric services. We have chosen MAS technology as the broadest approach, which is not restricted by a focus on any kind of networks.

The Serviceware Framework for context-aware services consists of several modules made up by agents providing functionality related to the areas described in Chapter II. The core of the framework consists of agents providing interfaces to functionality required for context-awareness, and of additional agents providing infrastructure functionality, in the form of the following modules:

- The *Personalization* module provides techniques for Information Filtering used to process large amounts of information, returning to users only individually relevant data. Depending on the character of the information to be recommended, different techniques may be utilized via this module, such as Collaborative Filtering, Knowledge-Based Filtering, Feature-Based Filtering, or combinations thereof.
- The *Location-Based Services* module supports the localization of users, and additionally provides ontologies for processing location information. It

provides functionality for mapping and routing between different locations, and suggests specific Points of Interest (e.g. tourist attractions or pharmacies) within a given region.

- The *Device-Independence* module provides services utilized to generate User Interfaces for different devices without having to change the underlying functionality of the respective services. This is mainly achieved by separating the device-dependent layout aspects from the user interaction aspects in each dialog. Layout aspects are specified in an abstract format by the service developer and mapped to the current device display by the framework.
- The *Infrastructure* module supports, among other aspects, the management of users, sessions, and services. With regard to ubiquitous computing, the user management is of special relevance: Each user is assigned a specific user agent managing all personal information. For privacy and security reasons, personal information is only accessible via this user agent. Thus the user may protect his personal information by removing the user agent from the system when he is not logged in. On the other hand, he may keep his user agent connected to the system even when he is logged out. In this case, the user agent decides, based on rules specified by the user, when to allow access to personal information, e.g. in order to notify the user via email, SMS or voice messages that new recommendations are available. By storing no information on the user's device, seamless session handover between different devices is enabled: When the user connects to the system via a new device, he is offered to continue previous sessions, or to start a new session.

Specific functionality used by these interface agents, e.g. filtering techniques for the provision of personalized information, is provided via *External Utilities* which are easily exchangeable. The main advantage for the developer in using interface agents for accessing the External Utilities lies in the fact that the functionality may be accessed uniformly and transparently without the developer having to deal with technical details of the underlying functionality.

Additionally, various secondary services are provided supporting the user in managing his personal information, scheduling his appointments, or planning routes between different locations.

Using the Serviceware Framework, different application services can easily be build and deployed. Figure 2 shows the architecture of the Serviceware Framework. Developers of application services may use functionality provided by the Context Awareness Module. This module combines the three modules for Personalization, Location-Based Services (LBS) and Device Independence described above. Together with the Management Module and the secondary services, the Context awareness Module forms the basis of the Serviceware Framework.

V. Implementation

The Implementation of the framework has been carried out based on the FIPA-compliant MAS-architecture *Java Intelligent Agent Componentware* (JIAC) [2][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).

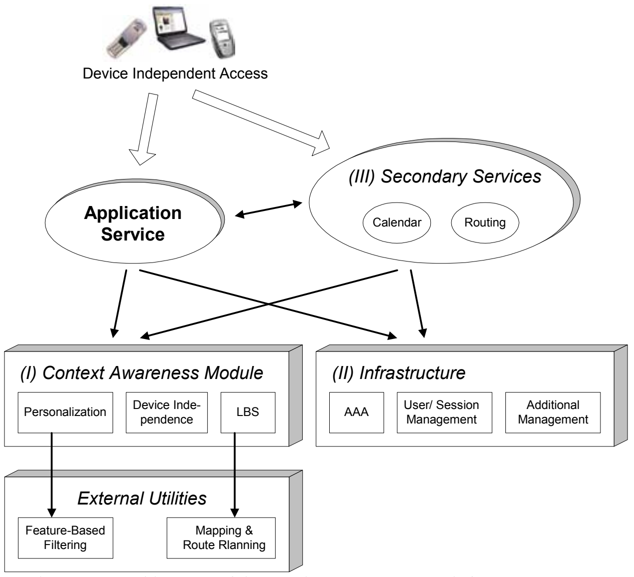
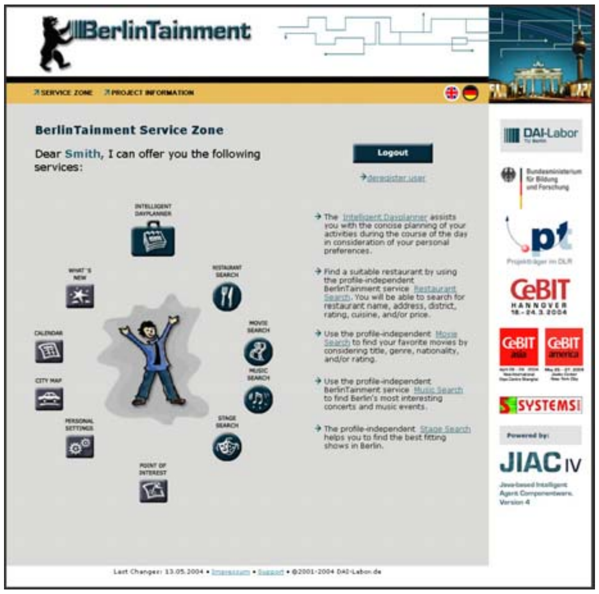


Figure 2. Architecture of the Serviceware Framework for context-aware services.



BerlinTainment Service Zone

Dear Smith, I can offer you the following services:

- The Intelligent Dayplanner assists you with the concise planning of your activities during the course of the day in consideration of your personal preferences.
- Find a suitable restaurant by using the profile-independent BerlinTainment service Restaurant Search. You will be able to search for restaurant name, address, district, rating, cuisine, and/or price.
- Use the profile-independent Movie Search to find your favorite movies by considering title, genre, nationality, and/or rating.
- Use the profile-independent BerlinTainment service Music Search to find Berlin's most interesting concerts and music events.
- The profile-independent Stage Search helps you to find the best fitting shows in Berlin.

Last Changes: 13.05.2004 • Impressum • Support • ©2001-2004 DAI-Labor.de

Figure 3. Portal of the BerlinTainment Demonstrator.

Certain aspects of the framework are addressed by JIAC itself: Regarding e.g. device-independent access to services, JIAC provides *Multi-Access Agents* (MAA) capable of adapting user interfaces, based on the user's current device. Basically, the CC/PP profile [4] of a device is used to determine the language the abstract user interfaces are to be transformed into (such as HTML/WML for browser-based interfaces, or VXML for voice-based interfaces), and to determine the way graphics are to be presented, based on the device's display size. Privacy and security aspects regarding personal information are addressed by encapsulating sensitive information, such as user profiles, within dedicated user agents. These agents are under control of the respective users and protected from unauthorized access by mechanisms inherent to JIAC.

Based on the implemented framework, we have developed several prototypical context-aware services comprising the BerlinTainment demonstrator, an application for entertainment planning in Berlin [10]. Figure 3 shows a screenshot of the portal of the BerlinTainment demonstrator. The application is accessible via different devices and channels, such as voice, mobile phones, PDAs and PCs. An additional secondary service, the *Intelligent Day Planner*, integrates the different context-aware services, allowing the user to schedule his activities for a given day and receiving personalized and location-based recommendations for each activity, such as restaurant, cinema or theatre 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.

The quality of a framework is mainly dependent on its reusability. The implemented framework consists of several specific agents implementing functionality which can be reused in other frameworks or in order to speed up the development of application services.

For the management of services and users, functionality is provided by several agents, which form a special secondary service, the *Basic Service*. This service handles for example registration and login procedures for users and is also responsible for registration of application services and secondary services. Additional secondary services are available to enhance the functionality of application services. Examples for functionalities of secondary services which can be used within application services are calendar services as well as routing and mapping. These functionalities from secondary services are immediately available for new application services once they register to the Serviceware Framework.

The implemented prototype is deployed as a distributed system containing several agent platforms and a large number of single agents. The administration of the system is supported by several tools, such as the agent platform monitor shown in Figure 4, allowing the supervision, configuration and migration of agents. Thus, the system may be adapted by adding additional agents, and re-distributed on a different server configuration, if necessary even at run-time.

VI. Related Projects

Several projects providing services to end-users have been developed that are similar to BerlinTainment. One of them is the German DOM project, which aims at providing personalized and location-based services for end-users [5]. The approach of DOM consists of four layers: Content, Basic Functions, Transactions and General Service layer. The DOM approach is further structured in functional components but lacks in contrast to agent-based approaches in terms of scalability and flexibility.

The same disadvantage applies to the LoVEUS project [6]. LoVEUS aims to provide ubiquitous services for personalized and tourism-oriented multimedia information. It follows a client-server approach incorporating a server with numerous clients.

The CRUMPET approach deals with the creation of user-friendly, personalized and mobile tourism services [7]. The architecture is based on a multi-agent system. The user or terminal agents are hosted on the end-users' terminal devices and provide service GUIs. A brokerage function enables the user agents to declare interest in particular services and receive information about services that meet special criteria, such as proximity constraints. For example, when the users' location changes, local services may then meet the specified service constraints and are offered to the user. While personal agents are generally suitable for context-aware services, hosting agents on users' devices has the drawback that special software is required on the device. Thus the range of usable devices is limited and device independence cannot be achieved.

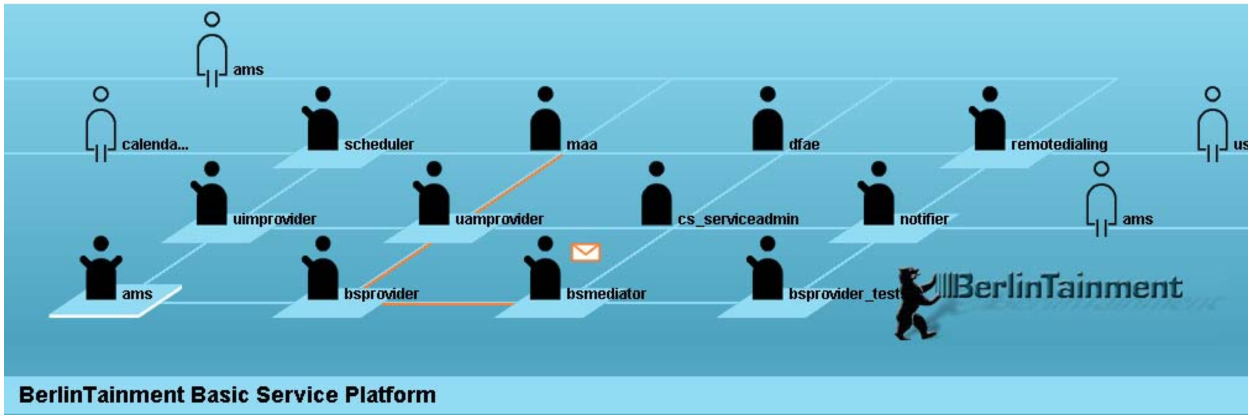


Figure 4. Agent Platform Monitor of a platform of the deployed prototype.

VII. Conclusion and Further Work

We have developed a Serviceware Framework for context-aware services based on MAS technology. By developing prototypical applications based on the framework, we have shown that it may in fact be utilized to develop context-aware services efficiently. Various modules for different tasks and services allowed the development of the entertainment prototype within a short time.

As shown, non-agent-based approaches, lack the capabilities inherent to agent-based approaches with regard to requirements such as modularity and adaptability. In contrast to approaches requiring specific software (e.g. for hosting user agents) on the user's device, thus limiting the set of possible devices, our approach does not require any additional software (apart from a browser) on the user's device.

Further work will focus mainly on providing additional functionality via External Utilities, such as knowledge-based filtering techniques for improved personalization, and on providing additional interfaces, e.g. for location tracking. Eventually, an application based on the framework might be deployed as part of a commercial service.

References

[1] S. Albayrak: Introduction to Agent Oriented Technology for Telecommunications. In: S. Albayrak (ed.): Intelligent Agents for Telecommunications Applications, IOS Press, p. 1 – 18, 1998

[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] 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)

[5] DOM – Der Orientierte Mensch, project funded by German Federal Ministry of Education and Research (2002), http://www.der-orientierte-mensch.de/ (28 June 2004)

[6] LoVEUS - Location aware Visually Enhanced Ubiquitous Services (2003), http://loveus.intranet.gr/ (28 June 2004)

[7] CRUMPET - Creation of User-friendly Mobile services Personalised for Tourism (2003), http://www.ist-crumpet.org/ (28 June 2004)

[8] ETSI ES 202 915-1: Open Service Access (OSA); Application Programming Interface (API); Part 1: Overview (Parlay 4), August 2003

[9] Sun Microsystems: Jini Architecture Specification, Version 2.0, June 2003

[10] J. Wohltorf, R. Cissee, A. Rieger, H. Scheunemann: An Agent-Based Serviceware Framework for Ubiquitous Context-Aware Services. AAMAS 2004 Demonstration Session, New York, USA