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

Jens Wohltorf, Richard Cissée, Andreas Rieger, and Heiko Scheunemann

DAI-Labor
Technische Universität Berlin
GOR 1-1, Franklinstraße 28/29
10587 Berlin, Germany
Phone: +49 (0)30 – 31421758
{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. Within the BerlinTainment project, we have utilized different modules of the framework to develop prototypical application services in the entertainment domain providing personalized, location aware, and device independent information.

1 Introduction

The ability to design, develop and deploy reliable context-aware services efficiently is becoming of central importance for providers of mobile services facing 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 we have developed within the BerlinTainment project, based on 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. Additionally, our approach supports ubiquitous context-aware services by introducing user agents as independent actors within the architecture.
Non-agent-based approaches, such as the DOM [1] and LoVEUS [2] projects, 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, such as the CRUMPET Project [3], our approach does not require any additional software (apart from a browser) on the user's device.
This paper is structured as follows: The following Chapter describes the problems arising from developing context-aware services. Chapter 3 lists requirements of frameworks addressing these problems. Chapters 4 and 5 describe our approach and implementation of an agent-based framework for ubiquitous context-aware services. Chapter 6 concludes the paper and outlines further work.

2 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:

- 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;
- Additionally, infrastructure functionality for user and session management, authentication/ authorization, accounting, notification of users, and additional management tasks is required;
- Finally, to increase the usability of the context-aware service itself, further *secondary services* offering the user assistance e.g. in the form of calendar or community services may be required to add value to the overall application.

Figure 1 shows the functionality required in addition to the application service itself.

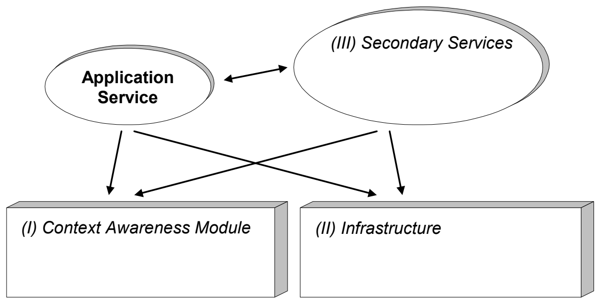


Fig. 1. Functionality required in addition to a context-aware application service itself. The arrows indicate dependencies between the modules.

Consider the following scenario: The user wants to receive recommendations of restaurants, based on his current location as well as his user profile containing his preferences regarding cuisine, price, etc. He intends to interact with the respective service via different devices, such as mobile phones, PDAs, and PCs. He may, for example, start the service via a PC to receive recommendations of restaurants, and continue the service seamlessly on his way to the restaurant via a mobile device, in order to receive routing information. Additionally, he is interested in receiving additional new recommendations even if he is not currently logged in to the service.

Developing a service based on these requirements may be split up into developing the service itself (in this case consisting mainly of the user interaction dialogs and the management of restaurant information) and developing the required overhead functionality, as described above. In this scenario as well as in similar other cases, developing all required 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: The given use case may require a different service, e.g. a shopping guide instead of a service recommending restaurants, while the additional functionality and secondary services remain largely the same.

3 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. Utilizing such a framework, the developer may concentrate his efforts on the context aware-service itself, without having to deal with a large amount of details regarding 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, if possible even within deployed systems, 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.

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.

4 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 [4]. Basically, MAS architectures consist of agents encapsulating specific functionality and offering services to exchange information with other agents.. All agents exist within 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* 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.

The Serviceware Framework for context-aware services consists of several modules made up by agents providing functionality related to the areas described in Chapter 2. 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 *Management* 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 the interface agents of the framework, e.g. filtering techniques for the provision of personalized information, or third-party software for routing and mapping, 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, and planning routes between different locations. Figure 2 shows the entire structure of the framework.

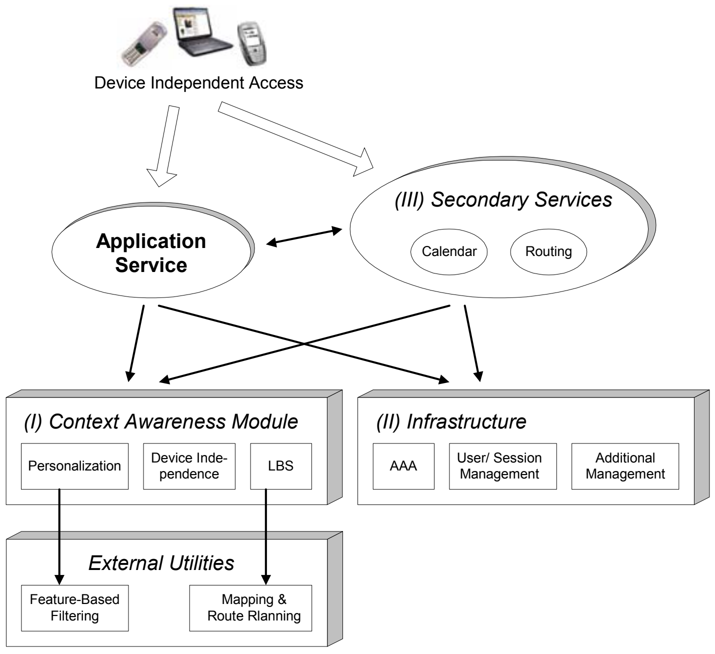


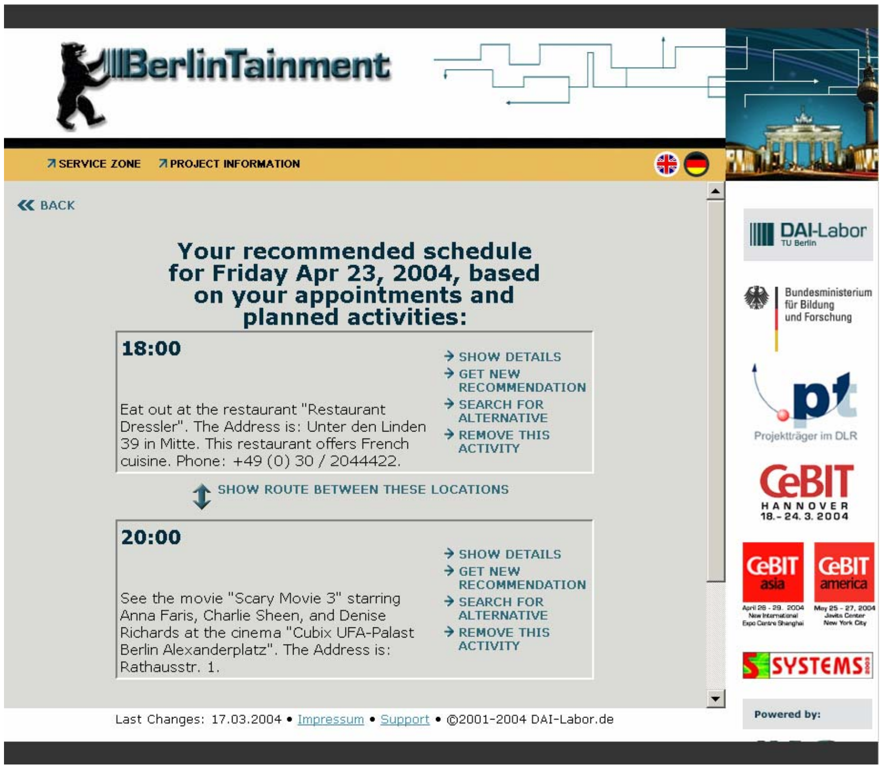
Fig. 2. Architecture of the framework for ubiquitous context-aware services.

5 Implementation

The Implementation of the framework has been carried out based on the FIPA-compliant MAS-architecture *Java Intelligent Agent Componentware* (JIAC) [5, 6]. 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).

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 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[1]. The application is accessible via different devices, such as 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, receiving 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. Figure 3 shows a screenshot of the Intelligent Day Planner presenting recommended activities.



Your recommended schedule for Friday Apr 23, 2004, based on your appointments and planned activities:

18:00
SHOW DETAILS
GET NEW RECOMMENDATION
SEARCH FOR ALTERNATIVE
REMOVE THIS ACTIVITY
Eat out at the restaurant "Restaurant Dressler". The Address is: Unter den Linden 39 in Mitte. This restaurant offers French cuisine. Phone: +49 (0) 30 / 2044422.

SHOW ROUTE BETWEEN THESE LOCATIONS

20:00
SHOW DETAILS
GET NEW RECOMMENDATION
SEARCH FOR ALTERNATIVE
REMOVE THIS ACTIVITY
See the movie "Scary Movie 3" starring Anna Faris, Charlie Sheen, and Denise Richards at the cinema "Cubix UFA-Palast Berlin Alexanderplatz". The Address is: Rathausstr. 1.

Last Changes: 17.03.2004 • Impressum • Support • ©2001-2004 DAI-Labor.de

Fig. 3. Screenshot of the BerlinTainment Intelligent Day Planner, presenting recommendations for activities based on this user's preferences.

6 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.
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. To increase the

[1] The demonstrator is accessible via www.BerlinTainment.de.

privacy of the user's information, mechanisms for privacy-preserving information filtering as suggested in [7] may be added to the framework.

References

1. DOM – Der Orientierte Mensch. Online at http://www.der-orientierte-mensch.de/.
2. Michalis Karagiozidis et al.: Location Aware Visually Enhanced Ubiquitous Services. 2003. Online at http://loveus.intranet.gr/.
3. B. Schmidt-Belz, M. Makelainen, A. Nick, S. Poslad: Intelligent Brokering of Tourism Services for Mobile Users. ENTER 2002, Innsbruck.
4. S. Albayrak: Introduction to Agent Oriented Technology for Telecommunications. In: S. Albayrak (ed.): Intelligent Agents for Telecommunications Applications, IOS Press, p. 1 – 18, 1998.
5. 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.
6. Ralf Sesseler, Sahin Albayrak: Service-ware Framework for Developing 3G Mobile Services. The Sixteenth International Symposium on Computer and Information Sciences, ICSIS XVI, 2001.
7. Richard Cissée: An Architecture for Agent-Based Privacy-Preserving Information Filtering. Sixth International Workshop on Trust, Privacy, Deception and Fraud in Agent Systems 2003.