A Flexible Architecture for Mobile Collaboration Services

Thomas Springer, Daniel Schuster, Iris Braun & Jordan Janeiro
TU Dresden, Germany
thomas.springer@tu-dresden.de

Markus Endler
Departamento de Informatica
PUC-Rio
Rio de Janeiro, Brazil
endler@inf.puc-rio.br

Antonio A.F. Loureiro
DCC/UFMG
Belo Horizonte, Brazil
loureiro@dcc.ufmg.br

1. INTRODUCTION

Collaboration services like Instant Messengers, audio and video conferencing, shared presentations and applications, have developed into important tools in business and private live. The availability of more powerful, sensor equipped mobile devices, higher data rates provided by urban WLAN infrastructures and the UMTS extension HSDPA as well as cheap data communication enable a wide use of collaboration applications also in mobile environments.

There is already a multitude of collaborative applications available on the market. Although they share a lot of common functionality, most of them are built from scratch, or are tailored to a specific device platform using proprietary libraries. An open and customizable environment for mobile collaborative applications is still missing.

An analysis of the top 50 proposals of Google's Android Developers Challenge should illustrate the need for platform support for mobile collaboration. These applications can be considered as a representative sample regarding user needs and market relevance. According to our analysis 41 of 50 applications (i.e. 82%) provide collaborative functionality, i.e. could use at least one of the services we propose in section 2. The collaboration features comprise shared editing of images, chatting, or discussing music or prices. The most important aspect is location-awareness, 52% use it, followed by communication features (26 %), geo-tagging (22 %) and content-sharing (20 %). 30 % of the applications combine at least 3 collaborative features.

In the following sections we present a conceptual architecture for mobile collaborative systems. We identify common functionality that may be helpful for a number of mobile collaborative applications. This include map-based visualization, multimedia geo-tagging, communication functions and comprehensive context management. We evaluated the architecture by implementing it under three different mobile platforms using free collaboration frameworks like XMPP or NaradaBrokering [2].

2. MOBILIS ARCHITECTURE

Our Mobilis conceptual architecture for mobile collaboration consists of four layers, namely the operating system of the (mobile) device, the basic services layer, the Mobilis services layer and the application layer (see figure 1). The *Device OS layer* represents the functionality provided by the operating system of the device (e.g. Windows Mobile, Symbian OS or the OS layer in Android)[1]. The *Basic Services layer* contains functionality which is implemented dependent on the operating system functionality. The *Mobilis Services layer* contains the services for supporting mobile collaboration. At this layer all services can share their functionality with all other services, e.g. the group chat service uses the group management service, and most of the services are adaptive using context. The *Application layer* is the layer where the applications reside. In the following we will focus on the Mobilis layer. We describe all services and their interdependencies with Mobilis as well as Basic services.

The **Context Management Service** is responsible for sharing context between different participants of a collaboration session (e.g. device and connectivity, physical context like location, noise or light level, and current situation like user task, or willingness to communicate). It uses basic services for request/response and pub/sub communication and Mobilis services for group management and security to deal with privacy concerns of context distribution.

The **Geo-location Service** provides access to location information and answers proximity and range queries. This includes the access of geographic information systems and lower layer positioning functionality (e.g. GPS or WLAN). The service builds on Basic communication functionality and access to local hardware (e.g. a GPS device).

The **Multimedia Tagging Service** supports to attach locations with multimedia content and for accessing arbitrary tags. The service uses the basic event detection and pub/sub service and Geo-location, Media Sharing and Context Management for representing tags by overlaying maps.

The **Group Chat Service** allows the creation of chat rooms and the exchange of asynchronous chat messages in the scope of particular rooms. It builds on the session management and Group and Context Management Service.

The **Group Management Service** enable the creation of groups and the control of adding and removing/leaving members to and from groups. The service uses the basic

[1]Because we focus on Java programming the Virtual Machine and additional Java libraries reside partially on that layer, partially on higher layers.

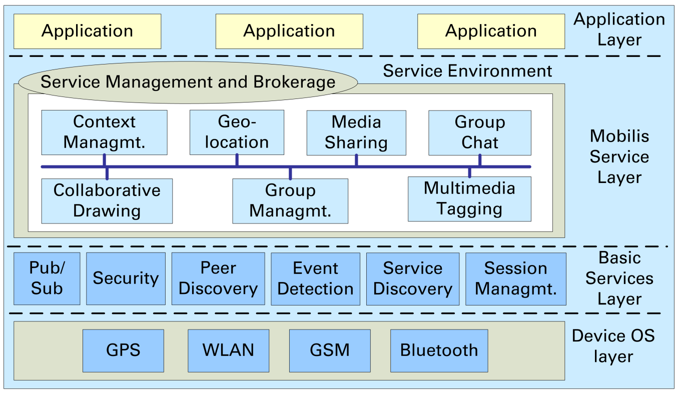

Figure 1: The Mobilis conceptual architecture.

pub/sub, event detection and session management service.

The **Media Sharing Service** offers the functionality to search for, access and offer multimedia content in the scope of groups. Therefore, the service builds on the pub/sub and session management basic services and the Mobilis services for Group and Context Management. Context is used to adapt content to connectivity and device capabilities.

The **Collaborative Drawing Service** allows to edit a common canvas/screen in a group. Especially an overlay with other content is supported, e.g. for drawing the shortest route between two locations on a map provided by the Geo-location Service. The service builds on the Event Detection , Session Management, Group Management, and Context Management.

3. IMPLEMENTATION EXPERIENCES

We implemented our middleware concept using the three platforms Java ME, Java SE and Android [1]. For providing collaboration functionality we adopted XMPP functionality, encapsulated into the specified services. XMPP is a family of protocols for collaborative environments based on a client/server architecture. A set of XEPs (XMPP Extension Protocols) provide further functionality, e.g. presence exchange, pub/sub, group management and group chat.

We implemented a collaborative location-aware tourist guide as a first application. One major requirement was the use of Java, because it is supported by the majority of device platforms. For XMPP/XEPs a set of libraries exists which implement the protocols on client site. The most comprehensive one is the Smack API, the API we choose for the implementation. It is the only library with support for XEPs for pub/sub and Group Management.

Our first target platform was *Java ME* as it is able to run on different mobile operating systems and is supported by a large number of mobile devices. But especially designing an appropriate GUI and integrating the XMPP libraries turned out to be very difficult under JME. We spent a lot of time trying different combinations of Virtual machines, versions of JME and XMPP APIs. The final solution uses a modified version of the Beep XMPP API on top of the Java Cre-ME virtual machine. Due to the difficulties with the JME approach, we implemented a *Java SE* version in parallel. There we were able to use the actual Java version with full functionality and the comprehensive Smack XMPP library. It was also easier to develop a GUI. The major drawback is the limited possibility to run Java SE on mobile devices. In a third parallel approach, we implemented a prototype based on the *Android* framework. With Android we were able to combine the good points of JSE-based development with a good GUI framework and abstraction for OS services like GPS. The Mobilis services were implemented as Android activities and services. We also used the Smack API for XMPP functionality.

In the context of our tourist application the Mobilis services are used and combined to support several features. **Geo-location Service** uses the *Google Maps API* to access and display the map around a position, and supports the common map navigation functions. **Geo-tagging Service** supports the creation and management of so called *Fun flags*, the selective display of them on the map displayed by *Geo-location Service*, and the binding of a multimedia object to a Fun flag, e.g. a photo or a text based on the *Media Sharing Service*;**Group Management Service** is used to create and manage tourist groups, and authenticate the group members; it exports any changes of the group membership to the other services; **Context Management Service** provide local and remote Mobilis services with different sorts of system context information, e.g. battery level, or connectivity status; and *Media Sharing Service* supports the share of any sort of content object either locally or remotely in a transparent way. For remote pub/sub communication,the NaradaBrokering system was adopted.

4. SUMMARY

Our market analysis has illustrated the need for a middleware platform for mobile collaboration. We have presented the Mobilis platform as our solution and discussed the services as well as their interrelations. While the Mobilis platform is a conceptual one it can be implemented on top of various operating systems. We have presented our experiences with the implementation of the Mobilis platform based on Java ME, Java SE and Android. As a result we discovered that our assumption about the lack of support for collaborative features in the platforms is correct. All platforms provide just basic features more or less related to collaboration functionality. The capabilities of the platforms are quite different. With Java ME we failed to implement the entire functionality because of version problems of Java and XMPP libraries. Some of the functionality had to be implemented again. The Java SE version was fully implemented but introduces limitations with regard to the supported device platforms. Just Laptops and powerful PDAs are supported. The Android platform provided the best support for our implementation. Especially the concepts for creating UIs and the easy to use APIs for Google Maps access have arisen to be helpful. In future we want to extend the services of our platform and plan to find new and innovative application scenarios with new requirements for our platform such as knowledge sharing within a group or P2P location-aware product search on mobile devices.

5. REFERENCES

[1] Google. Android - An Open Handset Alliance Project. Web page, 2008. http://code.google.com/android/.

[2] S. Pallickara and G. Fox. NaradaBrokering: A Distributed Middleware Framework and Architecture for Enabling Durable Peer-to-Peer Grids. In *in Proceedings of Middleware Conference 2003, Rio de Janeiro*, pages 41–61, 2003.