# AN OSGi BASED MOBILE DEVELOPMENT OVERVIEW

Pascal Suetterlin, Olaf Thiele and Hartwig Knapp
*Chair in Information Systems III, University of Mannheim, L5,5 , 68131 Mannheim, Germany*

Keywords: OSGi, Mobile Architectures, Overview.

Abstract: This paper presents an overview of the three most promising platforms for developing mobile applications: Titan (Java), Google Android and the iPhone. While the Titan platform already implements most design goals, Google's Android platform is in an early development stage. The iPhone platform lacks most functionality, but will soon be enhanced by a new firmware. The main focus in our comparison lies on the deployment and management of applications through the Open Services Gateway Initiative (OSGi) framework.

## 1 INTRODUCTION

Following the immense success of mobile applications for the consumer market, much research in the recent years has focused on mobile applications to support the business side of mobility. Most projects assumed that processing power as well as the development environment on mobile devices are very limited. With the introduction of new mobile platforms these assumptions might have to be altered. In this paper we present the three most promising mobile platforms on the market: Sun Java, Google Android and Apple iPhone. All platforms offer numerous possibilities, but as one of the main aspects of past research consisted of dynamically deploying and managing applications, we focus on this area.

The OSGi specification described below has these features as its main building blocks. Both deployment and management proved to be hard to implement in the past due to the need to access many functions of the underlying operating system. OSGi aims at providing these basic functions for mobile applications across several device specifications. Taking the viewpoint of a potential developer for a dynamically loading application we present what modern platforms are capable of as well as how they conform to the OSGi standard. For both classical cell phones (formerly J2ME) and the new Google Android platform OSGi implementations are available and described below.

## 2 TECHNICAL CONTEXT

This section describes the technical context surrounding the OSGi approach.

### 2.1 Service Oriented Architecture

Due to the increased complexity and size of mobile applications there is a high motivation to implement a Service Oriented Architecture (SOA) in both personal and enterprise environments. Moreover, the growing number of mobile devices calls for more abstract architecture implementations like SOA. Simply speaking, SOA enables programmers to build software applications composed of loosely coupled components (Duda et al., 2005). Deployed applications need to be suited to the user's requirements, to the resources of his/her Input/Output device and to the surrounding environment (e.g. context). Deployment refers to all activities following the development of the software which make it available to the user/device (Ayed et al., 2006). These activities consist of installing and configuring of the software but can also include software reconfiguration, updates or even un-installing it. One of the main benefits of a SOA in networked systems is that it reduces significantly the overall complexity of building, managing and deploying applications. Many of the technical requirements for a SOA are covered by an OSGi implementation.



Suetterlin P., Thiele O. and Knapp H. (2008).
AN OSGi BASED MOBILE DEVELOPMENT OVERVIEW.
In *Proceedings of the International Conference on Wireless Information Networks and Systems*, pages 245-248
DOI: 10.5220/0002027502450248

### 2.2 OSGi

The Open Services Gateway Initiative (OSGi) has been actively developing its specifications for almost 10 years and aims at implementing a lightweight framework for deploying and managing applications in service oriented architectures (OSGiAlliance, 2007). The OSGi Service Platform was built for the consumer market and the alliance has already defined a specification for mobile devices such as smartphones and PDAs in 2006. Based on the newly created version OSGi 4 Nokia and others launched the JSR 232 which is implemented by the Sprint Titan Platform as described below. The OSGi framework creates a host environment for deploying and managing bundles and the services they provide (Hall and Cervantes, 2004). A Bundle is a plain jar-file with a special format manifest file which is specific to the framework. Bundles are organized hierarchically. It's possible that one application uses another bundle or exports its own packages.

Besides the bundle-mechanism the OSGi standard describes a layer of four levels which provide the different functions as described in the specifications. These are the "L0 - Execution Environment", "L1 - Module", "L2 - Life-cycle" and "L3 - Service Model". The first layer L0 consists of the minimum execution environment so that almost every Java-enabled device could implement the framework. This layer can even be executed on older J2ME devices. The next layer L1 implements the concept of bundles that uses classes from each other in a controlled way according to system and bundle constraints. The management mechanism provided by the framework in layer L2 allows installation, activation/deactivation, update and removal of the bundles without restarting the virtual machine. The final layer L3 describes the part which provides the publish/find/bind services.

## 3 COMPETING ARCHITECTURES

In this section we highlight the specificities of the three platforms covered in this paper: Titan as a typical Java implementation, Google Android and the relatively new Apple iPhone. We start off with a short introduction into the Java VM. Figure 1 gives an overview of the OSGi implementations:

### 3.1 Mobile Java

Java 2 Micro Edition, which is meant for low memory consumer devices, provides a run-time environment for applications implemented in a stripped-down version of the Java language (Lund and Norum, 2004). The software implementations on the mobile devices differ from the reference implementation for desktop computers(J2SE) of Sun Microsystems and result in incompatibilities. Details can be found in the specification (J2ME, 2005). In addition to the minimal J2ME components called "Connected Limited Device Configuration" (CLDC), the "Mobile Information Device Profile" (MIDP) offers in version 2.0 new features such as an enhanced user interface, multimedia and game functionality, greater connectivity, over-the-air (OTA) provisioning, and end-to-end security (Sun, 2006). Some vendors implement more functionality defined in Java Specification Requests (JSR). This could be Bluetooth support (JSR-82) or access to the locally stored contacts (JSR-75). Nokia, for example, took the so called lead in several JSRs to standardize their implementation efforts.

In this context we make use of the "Connected Device Configuration" (CDC), that runs on the most recent cell phones as full implementations of a Java virtual machine. It already utilizes the JSR-232 which is the Java specification of the OSGi framework core and mobile specifications. In contrast to existing MIDP applications, this implementation can make use of servlets, allows for better graphical programming (games) and has direct access to the phone's hardware. The older CLDC and MIDP standards will probably give way to the fully J2SE compliant CDC. However, it has recently been stated that "the integration of OSGi with a Java EE environment is challenging as both technologies want to take control of the Java runtime" (Kaegi and Deugo, 2008). This might hold true for mobile applications as well.

### 3.2 Sprint Titan Platform

The Titan platform by Sprint (developed together with Nokia) offers built in support for OSGi through a JSR-232 specification and is therefore an existing implementation of the JSR. Furthermore, it is already included in the Windows Mobile SDK for Smartphones. Another advantage is, that the architecture is openly accessible. Legacy applications are run as classical MIDP applications, whereas OSGi bundles are run by the CDC VM. Generally, bundles are run in the background. The present Titan version supports the following application models: plain old MIDlets (POM), OSG bundles and deployment packages (DPs) and Eclipse embedded Rich Client Platform applications (eRCP). Future application models include OAMs (OSGi aware MIDlets) or widget based web applications.

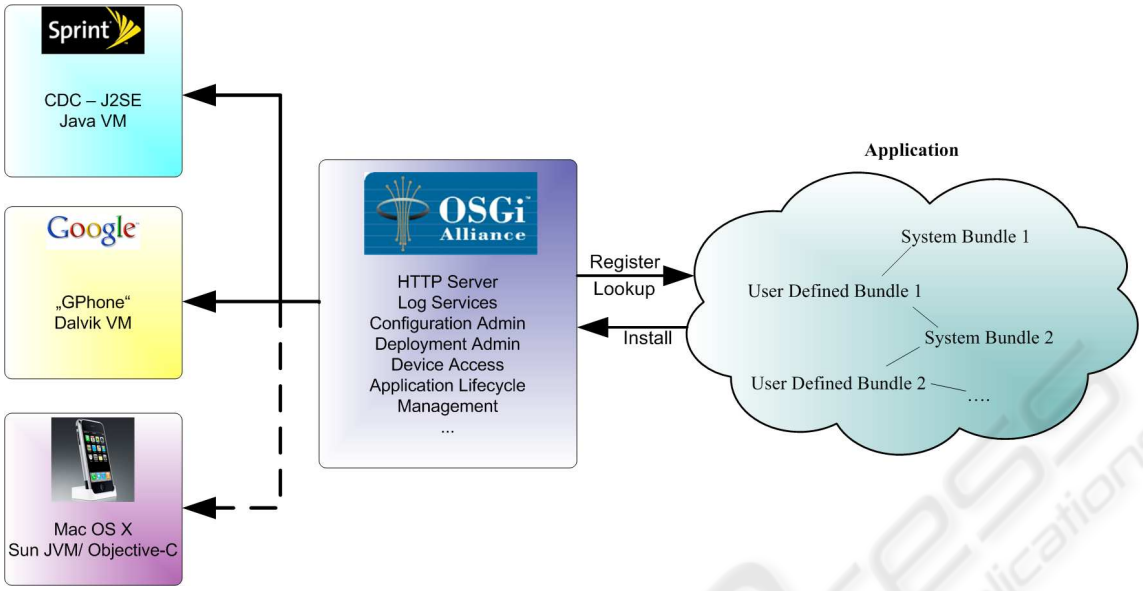


Figure 1: Competing Architectures and their OSGi Implementation.

A key OSGi concept of Titan is the possibility to manage applications at run time. This can be done remotely through the OMA-DM protocol. All necessary development files are supplied as Eclipse plug-ins. The plug-in aids in launching the VM, deploying applications and bundles as well as using a built-in profiler for debugging. As Titan is a true OSGi implementation, most software written beforehand can easily be ported to the platform. Effort for deployment and management is therefore minimal.

In a business context the Sprint platform can be seen as a full OSGi implementation. It is justified to say that other vendors will follow. Especially given the fact that CDC allows for standard Java programming instead of building specialized (cut down) versions with Sun's mobile J2ME platform. Two competitors in this area are the relatively new Android platform by Google and the very popular Apple iPhone. Both are described in the following sections.

## 3.3 Android

Originally the company Android Inc. worked on a novel (supposedly simpler) platform for mobile application development. While Sun tries to overcome the limitations of J2ME by introducing J2SE (Java Standard Edition) features with CDC, Android offers a simpler unifying approach. Google bought Android in 2005 and thus drew public attention to the project. In 2007 Google founded the Open Handset Alliance together with LG, Motorola, Samsung and others in order to develop an open mobile platform. This Alliance focused on the development of the so called "GPhone". Even though no mobile device is yet available, it can be seen as a competitor to Nokia's market power.

The Android platform builds upon a Linux kernel, uses Apache Harmony, offers several APIs and runs through a newly developed virtual machine name Dalvik VM. Through this setup, Google avoids paying the Java licensing costs for mobile devices. Even though the Dalvik VM draws heavily from the Java world. Dalvik source code follows the Java syntax, but offers different mobile APIs. The Android dx tool offers to port existing Java byte code to executable Dalvik byte code. Thus, existing Java source code can easily be ported to Dalvik source code. General Dalvik APIs include SQLite (one of the smallest database systems), OpenGL, SAX-Handler, Bluetooth (BLUEZ), JSON and a WebKit to run a browser (e.g. Safari).

Apache Felix is the successor of the Oscar open source project and implements the OSGi release 4 core framework specification (Felix, 2008) for general use on Java platforms. Its goal is to provide a completely compliant implementation of the OSGi framework specification. Through its almost J2SE-Standard JVM it's possible to run Apache Felix with only a few alterations (Offermans, 2007) on Android. Thus, Android implements OSGi. In the next section we present the iPhone platform.

## 3.4 iPhone

The iPhone platform has been developed for the Apple iPhone. The iPhone combines functions like me-

dia player, mobile smartphone, digital camera and Internet access in one device. Thus we speak of the iPhone as both the physical device as well as the mobile platform architecture. Comparing to Android, the iPhone platform currently renounces the Java technology due to licence fees which are around 5 - 10 Dollar per device. The operating system is an reduced Mac OS X (Leopard) which needs about 700MB disk space. The scope of operations is limited compared to the desktop version but uses an Linux-kernel (Darwin) with an modified development framework (Cocoa) that supports touchscreen abilities.

Among a lot of helpful features the iPhone has main limitations in application development comparing to other devices. First, the iPhone has no Java virtual machine available, all applications are written in Objective-C. Second, there is currently no possibility to deploy an application to an iPhone. Facing this drawback, Apple announced launching the SDK 2.0 in 2008. Developers are then able to send their applications to Apple and distribute it via the AppStore software which will cause costs for both, developers and users. Another option to bring an application on the iPhone today is to ”jailbreak” it. During this process some security key functions are disabled so that downloading iPhone applications from third parties like regular developers becomes possible. Of course, the claim for warranty expires immediately. Hence, this is not an acceptable option to deploy applications in a business context, but it might be alright for research purposes. Apple will provide some more key features for the iPhone which are absolutely necessary for the application development process.

## 4 CONCLUSIONS

We presented the three most promising mobile platforms for dynamically deploying and managing applications available today. At first we looked at Sprint's Titan platform which has a built-in OSGi implementation according to JSR 232. One of the design goals of the platform was the abililty of an SOA-oriented framework. In our opinion Titan is a powerful platform for dynamic distribution and collaboration of software components and applications. There is also support for the Eclipse embedded Rich Client (eRCP) application model which means that rich GUI applications can run across a broad range of devices. A major drawback is that there is only a Windows Mobile implementation which means a limited device selection.

The second platform presented, Google Android, has no built-in OSGi-Framework. We considered some OSGi-implementations such as Eclipse Equinox, Apache Felix and Knopflerfish. Due to the lack of available hardware, we could only test Apache Felix in the emulator of Android SDK.

The iPhone SDK as our last platform has no official mechanism for application distribution yet. Through the business model of the iPhone Apple wants to take a ”small” amount of the sales from third-party applications. Apple created a so called App-Store to make application delivery as simple as possible and wants to add it in the next iPhone firmware. After completion, it should be possible to bring a JVM to the iPhone (as announced from Sun). The next step would be to port a OSGi-implementation to the iPhone. When that task is completed then it would be possible to use the OSGi-framework on the iPhone. From hardware aspects Android could also run on the iPhone. We hope that our insights on present OSGi implementations will help further research.

## REFERENCES

Ayed, Taconet, Bernard, and Berbers (2006). An adaptation methodology for the deployment of mobile component-based applications. In *ACS/IEEE International Conference on Pervasive Services*.

Duda, I., Aleksy, M., and Butter, T. (2005). Architectures for mobile device integration into service-oriented architectures. In *ICMB 2005: Proceedings of the International Conference on Mobile Business*.

Felix (2008). Apache felix project: http://felix.apache.org/.

Hall, R. and Cervantes, H. (2004). An osgi implementation and experience report. *CCNC, Consumer Communications and Networking Conference*.

J2ME (2005). Java 2 micro edition: http://java.sun.com/j2me/.

Kaegi, S. R. and Deugo, D. (2008). Modular java web applications. In *SAC '08: Proceedings of the 2008 ACM symposium on Applied computing*, pages 688–693, New York, NY, USA. ACM.

Lund, C.-H. W. and Norum, M. S. (2004). A framework for mobile collaborative applications on mobile phones. Technical report, Norwegian University of Science and Technology.

Offermans, M. (2007). Osgi, google android, apache felix: http://blog.luminis.nl/roller/luminis/entry/osgi_on_google_android_using.

OSGiAlliance (2007). Osgi service platform release 4: http://www.osgi.org/release4/.

Sun (2006). Mobile information device profile (midp); jsr 37, jsr 118:http://java.sun.com/products/midp/.