Platform architecture portfolio
Comparison of 3 platforms (Android, iOS, mobile Ubuntu)

Marlo Häring

Fachbereich Informatik
Universität Hamburg
Vogt-Kölln-Straße 30
22527 Hamburg
marlohaering@googlemail.com

Abstract: In this paper I will introduce three different mobile operating systems and their related developing approaches. Each of these systems requires divergent technical infrastructures to develop mobile applications.

1 Introduction

Android and iOS are currently the leading operating systems because of their popularity. Figure 1 shows the dominance of these two systems. On the 3rd of January Mark Shuttleworth, who provides leadership for the Ubuntu operating system, announced a new smartphone interface for its popular operating system Ubuntu.

Although all operating systems have the same tasks, they differ in numerous areas and fulfill them with diverse approaches. This is particularly evident in the area of user experience. Mobile Ubuntu for example tries to transfer their popular unity environment besides desktop PCs and television also to smartphones. So it stands out from existing operating systems. In particular the developing progress of a mobile application requires different technical infrastructures and platforms.

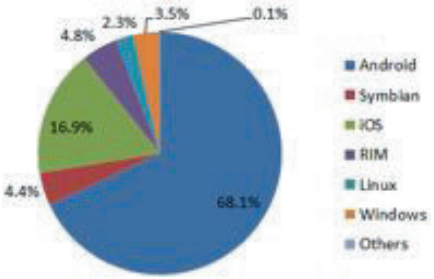

Figure 1: IDC looked at global smartphone shipments in Q2 2012, placing Android well out in front in terms of operating system shareSource: IDC Worldwide Mobile Phone Tracker, August 8, 2012

2 Developing for Android

Android apps can be developed on any common platform (Windows, Linux, MacOS) since *Eclipse* is the most popular IDE and it is mainly written in Java. Furthermore *Eclipse* brings a very mighty debugging engine as well as a plug-in infrastructure, which allows great flexibility. The Android Development Tools (ADT) are also installed as a plug-in. They include an emulator, which behaves like a real Android device and allows you to test your application without having a real device. It is even possible to create multiple Android Virtual Machine instances (AVM) with different customizations like display size, sd-card capacity or API Level. The emulator can run several AVMs in parallel. The whole backend code is written in Java. Android provides a subset of the Java 7 libraries adapted to the UI widgets and touch events that occur through the touch interface. The interface is defined in XML. The UI elements are structured hierarchically. *ViewGroups* are container. They define the layout structure of their containing *View* elements and can include further *ViewGroup* elements. In this way the GUI layout is designed. Eclipse brings a simple "*what you see is what you get*" GUI builder which parses and generates XML code. *DroidDraw*[1] is an alternative GUI builder also usable as an online applet version. To access UI elements in Java each element has a unique ID, which is defined in the XML structure. Subsequently action listener can be registered so that events are handled. When the app is built it is first compiled into *.class* files (Java-Byte code) and thereafter converted to *.dex* (Dalvik executable) files which can be executed by the Dalvik Virtual machine (VM). This VM creates an instance for each app running.

3 Developing for iOS

The developing of apps for Apple devices requires a Macintosh (MacOS). The developing environment consists basically of *Xcode*. It provides a very efficient simulator for IPhone and IPad as well as powerful debugging tools just like Microsoft's *Visual Studio*, *Eclipse* or Oracle's *Netbeans*. The sizes of apps developed are mostly very small so they can run on relative weak processors. For this reason the computer does not need to be powerful. It should have enough power for *Xcode* and especially for the simulator to run smoothly. Using the simulator the whole developing process can be accomplished exclusively on the mac so that no mobile devices are needed. The iOS version, display resolution and several other settings can be customized. User experience is the only field where the simulator is not suitable. Controlling the simulated device display by mouse doesn't feel as natural as by touch. The most inexpensive way to fill this gap is to buy an IPhone touch (roughly $300). Unfortunately a leak of sensors goes along with it but it is sufficient to feel the usage by finger swipes and accessibility of the UI widgets. The app is built with the *Model View Controller* (MVC) concept. The UI is designed by the *Xcode* integrated GUI builder and the backend parts (*Model* and *Controller*) are written in ObjectiveC. C and Smalltalk influenced this programming language so it brings object-oriented elements to C. Similar to C++ each class consists of

[1] http://www.droiddraw.org/

an interface file (.h) and an implementation file (.m). Properties are used to access the GUI within the ObjectiveC code. They can be created automatically with *Xcode* by dragging the respective element into the interface file. An outlet is created and can be used to interact with the UI element. Events are handled by delegate methods, which are methods that can be connected to a specific event. When the event occurs the connected method is invoked (delegate pattern).

4 Developing for mobile Ubuntu

Nowadays smartphones are produced, providing performance characteristics from former desktop PCs. This enables mobile operating systems to use modern multi core technology. As a result current graphic chips can render pictures for desktop PCs. With this technical development mobile Ubuntu forms in combination with a docking station, keyboard and monitor, a fully-fledged mobile workstation, which is expected to be very attractive to enterprise customers. Additionally mobile Ubuntu tries to attract smartphone beginners with a simple access to basic functionalities, used by the majority. Since mobile Ubuntu is a Linux system and has one of the best web service integration there are plenty of possibilities to develop an app. One option is the QML toolkit, which is built upon the newest version of the QT framework. The user interface consists of a tree of declared objects with properties. Similar to Android's UI elements QML objects have a unique ID, which is used to refer to. For dimensions QML uses grid units, which are device dependent. This is a helpful concept to build apps platform agnostic because layouts can be dimensioned once and used on multiple devices. A stricter JavaScript web browser variant can be embedded in several ways as a scripting language in QML. This framework provides a quick way to create interactive apps for Ubuntu on all devices.

5 Trends

Nowadays it becomes more and more important that web services are able to use system APIs to provide deep integration into the interface. The different kind of SDKs and frameworks make it difficult to develop cross-platform mobile apps, which appear same on different systems. As an intermediate step developer build their web services using HTML5, CSS and JavaScript and provide thin clients customized for each mobile system to access the programmed web service. By this way main parts of the app can be programmed generically as a website thus costs are saved because system independence is ensured. This development makes it obvious that web apps turn into first class citizens in mobile operating systems.

References

[And12] Android. Glossary. http://developer.android.com/guide/appendix/glossary.html, 2012. [Online; accessed 27-Jan-2013].

[Can12a] Canonical Ltd. Ubuntu app developer. http://developer.ubuntu.com/get-started/gomobile/, 2012. [Online; accessed 27-Jan-2013].

[Can12b] Canonical Ltd. Ubuntu for phones. http://www.ubuntu.com/devices/phone, 2012. [Online; accessed 27-Jan-2013].

[GR11] Mark H. Goadrich and Michael P. Rogers. Smart smartphone development - ios versus android.

[Qt12] Qt Project Hosting. JavaScript Expressions in QML Documents. http://qt-project.org/doc/qt-5.0/qtqml/qtqml-javascript-expressions.html, 2012. [Online; accessed 27-Jan-2013].