УДК 004.41

Ch. Schneider, H. Hermanni, K. Hänßgen

CROSS-PLATFORM NATIVE SOFTWARE DEVELOPMENT FOR MOBILE OPERATING SYSTEMS (ANDROID, IOS, WINDOWS 8 RT)

In this article you will find overview of most useful technologies for software engineering of mobile systems. The main problem is existence of group of mobile operation systems that has different paradigms and it is hard to develop software for all of them (Android, iOS, Windows 8RT). In this work we try to provide approach for software development for all mobile operation systems.

Keywords: Android, iOS, Windows 8RT, software development for mobile operation systems.

Introduction

Since 2011, for the first time, more mobile devices have been manufactured and sold than fixed-line computers [1]. The market for smartphones and tablets is visibly growing, as is the internet activity generated by users of mobiles and mobile applications. Software is no longer sold only on a carrier medium by way of the retail trade, but digitally through online stores. The biggest of these are iTunes AppStore and Google Play, with a total turnover of 2.2 billion in the first quarter of 2013 [2]. Such stores enable software designers to publicise their products directly, dispensing with the services of middlemen and so saving on distribution costs. Google's and Apple's commission amounts to 30 % as against their retailing of around 50 %. In contrast to the closed Apple landscape, Android additionally subdivides into independent or provider-specific suppliers, including AndroidPit, Samsung Apps and Amazon AppStore. The maintenance of all these channels costs a great deal of time and money.

The number of apps in iTunes AppStore has risen steadily since 2001, from 800 (July 2008 Release) to over 700,000 (October 2012 [3]). Google Play presents a similar picture, reaching the 600,000 apps mark in June 2012 [4]. More and more companies are developing apps for mobile operating systems. From newspapers (e.g. BILD, FAZ) through games developers and publishers (e.g. EA, Disney) to producers of navigation software (e.g. TomTom, Navigon) and Augmented Reality, the possibilities are many, while the billion users worldwide [5] form a correspondingly large and economically tempting market.

However, the differences between the mobile operating systems pose a major problem to software developers. An app created for iOS is incompatible with Android and vice versa. Only an extravagant degree of porting can cater for both markets at once: depending on the scope of the app, this may be expected to cost 50,000 euros and take a year to develop. Furthermore, updating of the apps will occur every year if regular.

To make matters worse, there are serious differences even within a single system landscape. Until 2008, the iPod and iPhone shared the same solution, which kept the expense of development down. Today, together with the iPad, there are five distinguishable solutions, which in turn exhibit three distinguishable ratios (4:3, 5:3 and 16:9) and display sizes. With Android, the number is significantly greater on account of its multiplicity of manufacturers (400 makers and 2,400 devices [6]). It is also worth offering landscape and portrait mode and keeping in mind the sundry specifications of devices (ARM, Intel, single core, quad core, multi-touch facility).

On account of this diversity, few companies manage to cover two or three different systems, and so most of them concentrate on iOS and/or Android. Apple, on the other hand, despite its smaller number of users, can generally expect revenues as much as three times the amount of Android's [7]. The reason for this is its better facilities for payment (Google for a long time issued only credit cards, in contrast to Apple's supplementary gift cards) combined with the higher social status (reflected in superior purchasing power) of its customers.

Planned projects

It follows from the content of Section I that, for the past few years, the right theoretical and technical conditions have existed for the use of tablet computers as an educational aid in schools. In a pilot project conducted jointly by the University of Leipzig, the Hochschule für Technik, Wirtschaft und Kultur and the Freie Gymnasium Borsdorf as a research project under the title

"Bildung serviert auf dem Table(t)t" ("Education served up on a tablet", with a pun on the German *Tablett* (tray)), an IT-based teaching system is being implemented, to be tried out in practice at the schools and in the partner organisations that are involved, with the aim of making work and communication "paper-free".

The various elements of the project programme are being worked out in interdisciplinary tandem by teachers and computer specialists. Research and development of new types of app's have to regard following areas:

- What tablet operating systems can be used for and how;
- Possible applications of various types of equipment and various software implementations;
- Possible applications of system management systems;
- Terminal system solutions;
- Questions regarding the network technologies and transmission protocols employed;
- Band-width and QoS management in computer networks;
- Questions of system security and backups;
- Remote control of system components;
- Presentation of electronically transmitted teaching materials;
- Systems for simplified editing of lesson preparations;
- Software for whole-class activities;
- Software and e-books for specialist instruction;
- Multimedia communication for groups and individuals;
- Pedagogical questions regarding use of the system.

The tasks that the project sets itself envisage for future use of the complex system:

- Introduction and testing of new forms of teaching via group communication, using broadband audio/video communication services in a school setting;
- Laying the foundations for enabling teachers to transfer their lessons entire on to electronic systems;
- Making teaching effective and practising the new standards for the transmission of knowledge in all areas of school and further education;
- Facilitating access to archives of teaching arrangements;
- Evaluation of a pedagogical concept developed in response to these.

The successive steps in the development of the system are so placed that their results are immediately available for use by students and teachers.

The use of efficient multimedia and network components, transmission protocols and software has made possible the inclusion of digital multimedia material in teaching and broadband audio/

Table 1. **Differences in programming between Android, iOS and Windows RT**

	iOS	**Android**	**Windows 8 RT**
Development environment	Xcode Mac OS X (only)	Eclipse Mac OS X, Windows, Linux	Visual Studio Windows 8 (only)
Licence models	80€ p.a.	25€ as single payment	75€ p.a.
Standard programming language	Objective C	Java	C#
C++	Yes (compatible with Objective C)	No (only indirectly on NDK and Java as intermediate level)	Yes
Graphics interface	OpenGL ES	OpenGL ES	DirectX
Resources	unpacked	packed	sandboxed
Maximum app size	2 GB	50MB / 4GB (expansion files required)	250 MB
Operator guidance	Home button, swipe gestures	Home button, back button	Swipe gestures
Multi-touch	5 fingers	1-10 fingers (often proprietary)	5 fingers
Solutions	320x480 640x960 768x1024 1536x2048 640x1120	Depending on class of device (infinite variety)	At least 1024x768
Types of device	5 iPhones 3 iPads 5 iPods	3,502	approx. 20

video-based multimedia communication, and has opened up completely new prospects for "paper-free" teaching. Thus for instance, online/offline audio/video sequences, including possible 3D representations, can be harnessed to educational purposes, while courses and specialist discussions can be carried on via computer network between places far apart in real time – e.g. as a whole class, for students' questions to the teacher, for student discussions or for student involvement in the teaching, all of which are prevented during participation in on-the-spot teaching. Courses held by specially trained experts on behalf of teachers can be rendered accessible to a large, far-flung circle of participants. There are two technical options for implementing the project:

1. Use of a server-based platform provided with the latest Web technology (HTML5).
2. Development of native applications directly linked to the underlying tablet systems.

Both alternatives are being tried out in "Bildung serviert auf dem Table(t)t". This article devotes itself to detailing the second approach, whose focal point is the cross-platform development of native applications and which facilitates the thoroughgoing use of tablet functions.

Differences among platforms

Table 1 shows what, for software developers, are the most serious differences between the three mobile operating systems. These also represent the biggest hurdles for the development of native applications for all three systems.

Difficulties of developing software for multiple operating systems

1) Operating system and development environment

While development for Android can be performed on the basis of a Microsoft as well as an Apple operating system, iOS and Windows 8 RT require their own systems, making it necessary to acquire two different developer devices. Using VMware is possible only at the respective additional expense, since it involves the emulation of further mobile devices which require direct SLAT and DEP access. The exchange of data between the various developer devices can take place via git, svn or comparable cloud solutions.

2) Native programming languages

Objective C, developed by Apple, is a continuation of C++. It can also be made with no difficulty to incorporate pure cpp data. C++ can likewise be used with Windows 8 RT. Complications occur with Android, which is based on Java. NDK, on the other hand, offers the possibility of also using C++ in the form of a native library, but with corresponding limitations: no direct access to device-specific information, libraries or audio interfaces. Java must be bypassed.

Google does come nearer the mark with more recent SDK and NDK versions of C++. However, the equivalent Java classes are not bypassed altogether but merely hidden from the developer, which creates more problems than it solves.

3) Graphics interface (OpenGL ES and DirectX)

OpenGL ES differs from OpenGL in being a performance-oriented slimmed-down version. Hence most graphics or interface libraries for desktop operating systems (Windows, Linux, Max OS X) are incompatible with mobile systems.

The mobile operating systems Android and iOS use OpenGL ES for graphics output. Therefore, applications can be ported between both systems without further expense on the side of the graphics interface. The one reservation is that certain adjustments may be necessary in the case of particular developments of OpenGL ES, since the graphics chips used in some Android devices are not the same as those used in an iPhone or iPad. On the other hand, an Android app that supports all 2,400 types of device is automatically compatible with an iPhone graphics chip, so that the porting of Android

4) Resources and maximum app size

In case where iOS is concerned, standard functions such as fopen/fwrite/fread/fclose can be used without difficulty. Android permits their use too, but regularly delivers resources files together with the app, which have to be read with the aid of zip_fopen or similar. With Windows 8 RT, by contrast, such functions are completely forbidden on account of the sandbox environment. To add to the difficulty, only the iOS operating system supports applications with up to 2 GB. With Android (Google Play), the maximum app size was for a long time 50 MB, a limit which can now be overstepped by means of expansion files, which allow up to 4 GB and are downloaded from the app. However, the development cost of this is high. In particular, the user is asked to acquire additional licences and acknowledge them himself in others (for instance, internet permission for an application which does not actually need it).

With Windows RT, at least for the time being, there is no chance of passing the limit of 250 MB.

5) Solutions and various classes of device

The number of different solutions for mobile operating systems meanwhile exceeds the number of active fixed-line computers. In particular, the iPad Retina has a solution of 1536x2048 – a million more pixels than in the case of an HD monitor with 1920x1080, and over a significantly smaller area.

Along with solutions, the underlying display size has to be taken into consideration also, since otherwise there is a danger that the control elements will appear too small and can no longer be picked out by fingers of the user.

6) Operator guidance and multi-touch

To some extent, individual mobile platforms are operated in very different ways. Thus, iOS has only one home button to end with, but real operator guidance takes place within the app by means of graphics buttons and digital menus (dashboard).

Most Android devices are also equipped with a back, context and search button, which render graphics buttons largely superfluous.

Windows 8 RT, on the other hand, relies strongly on swipe gestures, which feature in iOS too, though only as an option.

Users who change from one platform to another need a suitable period of learning to cope with these differing philosophies of operator guidance.

The very design of an application is worth noticing in order to get used to the habits of the various systems and so acquire an overall "user feeling".

Summary

In order to cover all three mobile operating systems, it is essential to implement an independent native solution which contains all interfaces in readiness on the operating-system level and brings them together so as to utilise these data on the application level without precise knowledge of the corresponding system.

By contrast with classical operating systems, it is advisable to create additional interfaces for access to files and activating graphic functions, to keep within maximum app sizes, and to make every control element movable and scalable for the sake of varying solutions and display sizes.

Programming languages other than C/C++ cannot be used. Libraries have their strict limitations too, since they cannot be compiled in advance but must remain a source code, unable to carry out their own file access and forbidden to make use of other, unsupported interfaces. Among other things, this applies to the use of fonts and importing of images.

Cross-platform development is rendered the more difficult by diverse concepts of user interface, which make it very dear in the matter of hardware because of the number of developer systems and the variety of test appliances involved.

However, a counterweighing application conception and an extensive creation of interfaces make it possible to embrace all three systems and, at the same time, to establish a development environment which will simplify the porting of future applications and automatically supply updates of already existing applications within all three systems.

In such a development environment, further mobile operating systems may later come to be integrated, since all applications developed on this basis still comprise only the existing program logic and outward transmissions of access are defined by correspondingly freely verifiable interfaces.

Therefore, all the programs developed within the framework of the project "Bildung serviert auf dem Table(t)t" will be available to all currently active mobile operating systems, so that students will be able to use their own personal devices for learning irrespective of the underlying system.

References

1. Source: Katy Huberty, Ehub Gelblum, Morgan Stanley Research, Data and Estimates as of 2/11. – Mode of access: http://www.slideshare.net/kleinerperkins/kpcb-top-10-mobile-trends-feb-2011. – Title from the screen.
2. Canalys Insight, Innovation, Impact, 2013. – Mode of access: http://www.canalys.com/. – Title from the screen.
3. iPad Mini 7.9. Announced Apple Event October 23, 2012. – Mode of access: http://www.youtube.com/watch?v=x3QYMCFhK3M. – Title from the screen.
4. Live from Google I/O 2012's opening keynote! – Mode of access: http://www.engadget.com/2012/06/27/google-io-2012-keynote-liveblog/. – Title from the screen.
5. Strategy Analytics vs Canalys: Q2 2013 tablet sales are off by 15 million Android devices. – Mode of access: http://www.androidauthority.com/strategy-analytics-canalys-q2-2013-tablet-sales-15-million-253124/. – Title from the screen.
6. Google Play Developer Console, August 2013. – Mode of access: https://play.google.com/store/apps/details?id=com.saurabh.developerconsole. – Title from the screen.
7. Distimo App Analytics, 2013. – Mode of access: http://www.distimo.com/. – Title from the screen.

Шнайдер К., Херманні Х. фон, Хенсген К.

КРОС-ПЛАТФОРМЕННА РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ МОБІЛЬНИХ ОПЕРАЦІЙНИХ СИСТЕМ (ANDROID, IOS, WINDOWS 8 RT)

У роботі зроблено огляд технологій, що використовуються в розробці програмних продуктів під мобільні операційні системи (Android, iOS, Windows 8RT). Проведено порівняльний аналіз проблем, які постають перед розробниками. Також пропонується один з підходів до вирішення знайдених проблем.

Ключові слова: Android, iOS, Windows 8RT, розробка під мобільні операційні системи.

Матеріал надійшов 25.09.2013

УДК 004.7

M. Finsterbusch, K. Hänßgen, P. Schmidt

OPTIMIZED CHANNEL SELECTION FOR MULTI-RADIO IEEE 802.11 BACKBONES

This paper presents the Simple Backbone Channel Allocation (SBCA) algorithm. It is used to optimize the channel assignment in IEEE 802.11 wireless backbone networks. We point to some specifics in channel assignment of backbones. It differs from channel assignment of single APs which is a node colouring problem of graph theory. Backbone channel assignment needs new strategies to cover all general conditions. We declare the SBCA algorithm and show its operation on an example. Our measurements show that it is very fast, correct in result and finite.

Keywords: IEEE 802.11 channel assignment, wireless backbone, mesh networks, network optimization.

I. Introduction

The use of IEEE 802.11 wireless LANs (WLAN) has been grown rapidly for the last ten years, promoted by inexpensive IEEE 802.11 capable PDAs, smart phones, laptops and WLAN routers. WLAN is used for business and private networks. Now the WLAN technology is increasingly used for wireless backbones to connect networks in countrified areas without broadband service or to cheaply connect different departments of a company which are located in different parts of a town [1].

Backbone networks

The backbone is an essential part of complex network topologies. Larger networks are often separated into different network segments (LANs), connected to each other and the wide area network (WAN) by the backbone. Due to its important aim to achieve the connection between many network segments, it has to realize special quality requirements. Backbones in general must guarantee high bandwidth, high availability and common *Quality of Service* (QoS) parameters.

The connection of different network segments to the backbone is done by *ingress-* and *egress-*nodes. An end-user station is never directly connected to a backbone network. It has to use the ingress gateway. If necessary, ingress- and egress-nodes also perform a media conversion (e.g. from IEEE 802.3 to IEEE 802.11 and vice versa). Backbone nodes themselves have only the task to route all network traffic. We denote them \emph{forwarding}-nodes or just routers.