INSIGHTS INTO STUDENT EXPERIENCES WITH MOBILE PLATFORMS AND APPLICATION DEVELOPMENT

Qusay H. Mahmoud¹, Shaun Zanin², Sacha Bagasan², Douglas Griffith², Justin Carvalho², Domenico Commissio²
¹Department of Electrical Computer and Software Engineering, University of Ontario Institute of Technology
²School of Computer Science, University of Guelph
Qusay.mahmoud@uoit.ca

Abstract – The proliferation of mobile devices such as smartphones and tablet computers are the newest paradigm shift occurring in the field of computing and engineering education. This paper presents an empirical and comparative evaluation of mobile application development platforms and tools as experienced by five undergraduate students who spent the summer of 2012 learning about and developing mobile apps at the Centre for Mobile Education and Research (CMER.CA), under the supervision of the first author. The students have worked with a wide variety of mobile platforms and tools, and hence the paper provides insights and recommendations into selecting mobile platform(s) to consider using in the Computing curricula.

Keywords: Mobile devices, mobile platforms, mobile app development, software engineering education.

1. INTRODUCTION

Mobile devices and the design and creation of software to run on them are areas of rapid growth. It employs many different principles from across the entire spectrum of the field of computer science & engineering. Some of these subcategories include high-level areas like Human-Computer Interaction, which is used in the design of fluid, touch-oriented interfaces. However, mobile devices also utilize principles and techniques from embedded systems design, such as the adaptation of existing applications and systems software for an environment which has limited hardware resources.

Due to their rapid growth and greater adoption in both enterprise and consumer smartphone markets, it is becoming imperative that students work with mobile device development over the course of their education. The responsibility for ensuring that students are ready to work with these devices falls to professors and program curriculum committees. Given their positions of responsibility, professors should try to empower their students by exposing them to mobile devices in the classroom, using practical, hands-on teaching approaches.

One difficulty which instructors may unfortunately face is choosing which mobile platform they should utilize in their coursework. Usually, the only way to see what would work is through a trial and error approach; the first iterations of a course in this case could end up being a taxing experience for both the students and the instructor as well. The goal of this paper is to advice instructors who wish to integrate mobile devices into their courses, and provide an empirical overview and a discussion of the strengths and weaknesses of each of the major mobile platforms available on the market today. In addition, this paper will propose the use of HTML5 as a preferable classroom alternative when compared with doing native development. This assessment is based on the experiences of five students who started working with mobile devices at the Center for Mobile Education and Research (CMER) as a summer job in 2012. They hope that their experiences and observations will bring faculty a student’s perspective of how mobile devices can be used to enhance the education of future computer scientists and software developers. They were all computer science students with differing levels of experience, specializations and interests. The Center for Mobile Education and Research is dedicated to pushing the research, design and development state-of-the-art educational material and applications, based around mobile devices, to enhance education in the field of computer science and beyond. The resources that CMER contributes include materials such as labs and course packs for instructors, to everyday applications which can be used in both educational and day-to-day settings [6,8]. Researchers at CMER also study and publish new, cutting-edge teaching approaches which leverage the constantly evolving nature of mobile devices [8].

The rest of this paper is organized as follows. Section 2 discusses some examples of the development projects which the students worked on in summer 2012. Section 3 presents a detailed evaluation and recommendations (from a student’s view) to instructors who are interested in integrating mobile application development into their courses. Section 4 discusses development challenges, and
Section 5 concludes the paper and presents ideas for future work.

2. MOBILE PLATFORMS AND APPLICATIONS

The students have worked with four major mobile platforms: iOS, Android, BlackBerry, and Windows Phone Mobile. They have done a comparative evaluation of the strengths and weaknesses of the major platforms which are commercially available today. In order to provide a case study that would allow for comparison between these different platforms, we believed it was necessary to create a basic application which used non device-specific features, and to implement it independently for as many of the major platforms as possible. Here we focus on BlackBerry, iOS, Android, and Windows Phone Mobile devices (smartphones and tablets).

One of the initial applications that all the Summer 2012 students at CMER created on all the major platforms mentioned previously was an app which is used to calculate a user’s body mass index (BMI), given their weight and height. This was developed using the SDKs available for each platform. Each application had the same design requirements, which the students hoped would give them a benchmark which could be used as a point of comparison between the different mobile platforms.

The experience of developing this mobile application gave the students a chance to make observations about the overall capabilities, and feel of each platform. It also provided some insightful windows into the development process. One example which varied greatly between platforms was the amount of time needed to develop the application for each platform. In this case, the Android application and the Windows Phone Mobile which can be seen in Figure 1. The Windows phone Mobile version required the least amount of time to develop.

The students also had an opportunity to work on various projects related to some of the different areas of research and development which CMER targets. For example, some students had the opportunity to focus on long-term development application that involved consulting with the client over a prolonged period in order to get the data needed to ensure that the application met their requirements.

Others had the opportunity to show off the versatility of the Adobe Air platform. Several of the applications are available on the major platforms, and provide an example of some of the capabilities which exist in the Android mobile platform. As a result of their development experiences, the students believe that they can offer instructors recommendations into how mobile devices can be integrated into their courses in order to improve the quality of education everywhere.

3. EVALUATION AND RECOMMENDATIONS

One of the overarching goals for this paper is to help instructors make informed choices about the benefits and type of mobile devices they should integrate into their curricula. Which mobile platform should they use? What aspects of which courses are mobile devices a good substitute over more traditional computing systems? This section of the paper will answer these questions, from a student-centered perspective based on empirical observations derived from working in an academic environment. Most of the evaluation metrics used will be based on the students’ mutual observations, rather than a quantitative measure of technological capability, or performance benchmarks of the different platforms.

3.1 Themes for Analysis

There were several overarching themes which the students all became aware of as their experience in mobile development grew. The first of these phases was simply the setup of the tool chains and development environments which was required before any software development could even comments, followed by the quality of the emulators, integrated development environment, and application signing and deployment (both for testing and production).

3.2 Setup

The first quality which an instructor should examine when looking out at mobile platforms is the ease of setup; this is especially important for courses in which practical labs play a large part of the learning and evaluation process. Delays and problems in configuration and deployment of the tools and packages needed to target a
platform can lead to many hours of wasted time, in which students will learn effectively nothing. This can lead to any material in a specific lab which was meant to occur at the same time as a particular course could be offset or removed entirely from the course. Students will feel frustrated not being able to actually use the skill which is being taught in these classes.

Setting up the environment for BlackBerry devices differed greatly depending upon the platform which you were using. The WebWorks package was easy to install, and no setup was needed for an IDE, as there was none provided (there are plenty of third party HTML/CSS/JavaScript development environments available). The signing process was usually easy, and was handled by an automatic Java executable, launched by the WebWorks Ripple emulator.

The next platform which was covered is the Java-based Android SDK. It integrates itself as a plugin into eclipse, and also includes a manager for creating and managing emulated devices. The process of downloading and installing the Android SDK is well documented and easy. The downside to this is the setup time. After installing eclipse and the ADT (Android Development Tools) plugin, you have to download files to target each particular variation of the Android kernel you wished to install. This includes separate emulator images for each version number. This process, especially the download of documentation in order to make it available offline, is a long one which can take a long time over a slow internet connection.

The setup for Windows Phone 7 is a relatively fast process. It requires a developer to download a version of Microsoft Visual Studio in order to develop for their mobile devices. A developer must also install plugins for .net development and other multimedia frameworks, however, they are easy to find and available. The install process for this version of Visual Studio takes a relatively short amount of time; however, it is necessary to register the express version after 30 days, regardless of whether a developer chooses to pay for a an upgraded version of the IDE. Overall, it is a relatively fast process, but it is very platform specific, being a windows-only option development environment.

Setup for Xcode, iOS is a straightforward process. The download is quite large, at over 700MB; once installed however, all one has to do is follow the setup wizard. The biggest issue with the iOS platform was the process of setting up signing keys for deploying applications to devices.

3.3 Device Emulators

One factor which is often overlooked is the emulator (or lack thereof) which is provided with each platform. While it is almost always better to test on a live physical device, most courses will not have enough mobile devices for everyone to be using the exact same mobile device. Emulators provide students an excellent way to re-create any iterations of a device which may exist and allow them to experiment with different system configurations.

Blackberry simulators and emulators differed in quality depending on the platform. The BB10 and Playbook emulators came as VMware application images which were compiled for an x86 platform. These ran wonderfully on VMware player, which is a free product. A VMware image could potentially be converted to a virtual box image if needed, however this requires more configuration and testing that most faculty would not be willing to do for their courses. Another advantage of VMware images is the graphical acceleration which VMware player can provide.

Android provides a large selection of emulators for developers to test on; often manufacturers provide images of their own devices for developers to work with. Android’s emulators are based on the open-source QEMU emulator, which has long been used by the embedded development community to simulate various architectures in order to test their software. While a very realistic implementation, the Android emulator has a reputation for being slow and unresponsive. This has been alleviated somewhat by the fact that Intel released x86 based images, which give the emulator near-native speed, and are capable of utilizing hardware-accelerated graphics resources if necessary; this only holds true for applications which use no native libraries however.

Windows Phone 7 comes with an emulator with which developers can use to test their devices. The emulator can take advantage of the hardware accelerated graphics, if available, and is a faithful reproduction of the feel of a normal device. For applications which are developed using a Silverlight-based GUI, it is more than sufficient for testing. Unfortunately, it tends to suffer during high-multimedia applications when the GPU acceleration is not available. Applications that are based on the XNA framework are unlikely to run at any great speed. Overall, the emulator for Windows Phone 7 could be useful for a course which was geared toward software design or Human Computer Interaction, as the “Metro” interface used in Windows Phone 7 is very distinct and different from any of the competing mobile platforms.

The last emulator covered here is the iOS emulator. From the students’ experience, it is easy to use and setup. It is responsive when running applications and it is actually easier to deploy applications to it than to the device. Instructors who have access to Apple-based products would find the emulators very useful if they were planning to do application development for iOS. The emulators also provide the same “look and feel” as the iPhone. With this performance, the iOS emulator is one of the best options in terms of device emulation.
Unfortunately, this also locks the instructor into using the OS X operating system for the duration of the course.

3.4 IDE

By far one of the most important areas for instructors to consider for their students is the utility of the integrated development environment which they will be using for the duration of the course. Each of the platforms provided a different philosophy behind their development environment, giving the developer who is using it a range of options for creating applications.

Blackberry WebWorks come with no IDE. Most of the developers who targeted the platform ended up using an editor like Notepad++ or vim. Instructors may find WebWorks a good platform for newer, less experienced students. More serious or upper year computer science or software engineering majors will likely expect a better, more sophisticated range of tools to use, and would get frustrated without a good development environment.

The students who worked with the Android platform found that the Eclipse-based IDE was a great tool for development. It provides a visual drag-and-drop editor that allows for the rapid creation of application GUIs, which are stored in XML files and compiled into a Java class when the program is executed. The Eclipse editor is also available on all major platforms which are likely to be used on a student or instructor’s personal computer. Eclipse was a great tool to use, providing useful autocomplete features and giving the students clear error messages and warnings.

Windows Phone 7’s Visual Studio is by far the easiest IDE to use and develop an application with. The environment uses a drag-and-drop system for GUI design. Much like Android, the configuration files for these GUIs are saved in an XML based format, which in this case is read by the Silverlight framework and interpreted into a form which is displayable to the user. These files can be edited manually as well and feature the IntelliSense Autocomplete system, which helps developers speed up productivity and decrease errors while coding. The IDE which is provided is also intuitive to navigate. For example, in order to open up the code for the event handler for a button press, the only thing a developer must do is double click on the desired button, and a stub hander will automatically be generated and a coding window will opening showing the new code.

The development environment for iOS, Xcode, can pose problems for new developers. This is because iPhone applications are developed in Objective-C. For those who have not used Objective-C, they will find that it is very different from other popular languages such as C++, Java, and C#. Examples include object notation, and using ‘Yes’ and ‘No’, instead of ‘true’ or ‘false’ for Boolean values. Identifying the correct way to complete tasks using the Xcode interface is not immediately apparent; many icons are displayed which do not have any descriptive text accompanying them.

3.5 Application Deployment

The devices which the students worked with had a range of capabilities. Each had its own specific strengths and weaknesses. The BlackBerry 5 & 7 lines of devices are the descendants of some of the first mass-produced mobile devices. They have excellent hardware capabilities and support for modern standards; the students developed applications for these devices using the WebWorks SDK. The development cycle for these HTML5 based applications functioning very quickly was very fast, allowing students to see their software functioning however, the devices themselves especially BlackBerry 5 devices were frustrating to work with. The BlackBerry Playbook and BlackBerry 10 development process was much better. Deploying applications for testing on a device required a few shell commands to authorize the device, but was relatively easy to use once it was setup.

Android was by far the easiest platform to deploy applications for. Having a device connected with the debugging functionality turned on (or an emulator created) was all that was needed to deploy applications on a device.

By far the worst platform for application deployment was iOS. The signing portion of mobile application development was the most extremely frustrating part of the process for the students who were involved with it. Signing keys for iOS must be created, then imported into Xcode. These can be difficult to get if a project is being developed by more than one person. The setup the students were given meant that the signatures for deploying an application to a device had to be created by a team lead, which severely restricted developer independence as every time a new project had to be started, the team lead had to go and obtain a new set of signing keys, to be handed out to the developers who needed them.

The Windows Phone was an easy device to deploy to, after the initial setup. Developers are required to download Microsoft’s Zune client and to have a Microsoft live account in order to register their device for publication. Normally the cost associated with becoming a developer for Windows Phone is $125 a year. However, through DreamSpark, Microsoft gives students the opportunity to develop applications for free.

4. DEVELOPMENT CHALLENGES

The above students’ experiences provide a large amount of information which an instructor can use as
suggestions or criteria when evaluating what development environment to use in course work. Unfortunately, the comparison of all the strengths and weaknesses of these platforms is large and time consuming to detail. In addition, the APIs, operating systems, are continually changing. Instructors should not have to update their lesson plans or labs simply because a patch was released for a particular environment.

This fragmentation in mobile operating systems, unfortunately, has no cure-all solution. Attempts such as cross-platform frameworks themselves lock the developer into using those tools, creating the same problem (code must be kept up to date if there are any major changes to the framework, locked in to a particular environment and way of doing things). Cross platform tools are often not kept up to date to the point where they able to utilize the most recent features which are present on a given system, and add another layer of complexity to an already complicated system [9].

4.1 Tools for Beginners

A unique approach which bears mentioning is the newer concept of having students not having to choose an IDE for programming a mobile device, but to perform all mobile devices programming on the device itself, as with TouchDevelop [9]. The designers of TouchDevelop argue that the way we use devices and the types of input devices users employ have changed with each paradigm shift in computing, and that the new input paradigm occurring right now is touchscreens and the cloud.

Basing the design around this idea, all programming of applications in TouchDevelop occurs on the device itself, using a touch-optimized interface. The framework has great success in encouraging middle and high school students to take interest in computer programming [9].

Another effort to introduce students to mobile devices and general programming skills was MITs AppInventor (formerly owned by Google). It features a drag-and-drop interface where users could build their application GUI instantly, and then use colored “blocks” to implement the logic for each component. This environment has been found to have benefit for students learning new programming concepts [10]. Innovative approaches like TouchDevelop and AppInventor are great for introducing programming to beginners in the field, as they are easy to use and allow users to see the results of their work quickly. They provide tools for instructors teaching beginners, but do not help with advanced courses.

4.2 Migrating and Updating

A student who attempts to migrate to from one mobile platform to another will find it difficult to reuse any previous code they have created without having to rewrite it. They will have to change languages, and for some platforms, perhaps even buy a new computer or license for an operating system before they can even start to create applications, in either educational or commercial environments.

Instructors can face similar difficulties when developing course material for students to use. Examples include deprecations of important APIs, changes in tools used, etc. This produces several issues as it forces a course instructor to keep apprised of every miniscule change that occurs with each revision of the device platform. Sometimes, heavy modifications would be required to even get the code provided in a lab to compile and run correctly.

A real-world example would be of problematic changes to APIs for instructors, is the recent modification to the IO rules for the most recent iterations of the Android operating system. As of version 4.0.3, disk and network IO cannot be performed on the same thread which the main loop for handing user interface events is running on. It is the addition of a new set of mandatory methods, and code restructuring, to properly permit the program to execute. In this particular example, if a lab designed using an earlier version of Android were used with the most recent tools, and included any sort of socket or disk bound operations, the program would compile, install and then appear to do nothing at run time, as the error is not detected at compile time, and is only logged in a file which is not viewable on the device itself.

It may appear that mobile platforms are far too difficult and transitory in nature in order to be used as a practical tool for instructors to use in their courses. The solution to this, however, is present: some form of standards should be codified to allow certain application functions to work regardless of the platform; essentially similar to the way in which protocols such as TCP/IP or SMTP are defined. For most individuals who follow trends in information technology, the name these of these new standards will be familiar, if somewhat surprising: HTML5.

4.3 Overcoming Bias

JavaScript is the primary client-side language used in HTML5-based applications today. Unfortunately, many professionals in computer science have a negative perception of it [11]. Developers’ aspersions were commonly focused around its poor performance and the poor coding practices associated with it (often being described as being an inherently “sloppy language”).

Recent advancements have changed the validity of these perceptions however. While JavaScript is often abused by amateur web developers, and otherwise used poorly (just like any other programming languages can be), JavaScript has largely matured, and proper object-
oriented programs can be written today. Additionally, browsers have made great strides in their efficiency, as well as taking advantage of hardware optimizations which are available. An example is WebGL [12], an HTML5-addition, now supported by two of the three major browsers on the market. If available, it takes full advantage of any hardware accelerated graphics cards present on the client’s device to display 3D graphics. Other available APIs which give HTML5 applications incredible power include: geolocation, WebSockets (network sockets) [13], video, canvas (2D graphics), File Access, and even message-passing parallelism.

HTML5 presents a great potential for becoming the standard application platform of choice for development on mobile devices. It operates in a sandboxed environment, similar to how most applications on smartphones are currently implemented. Indeed, many of these platforms have applications which appear to be compiled packages, but are really a set of HTML5 pages which occasionally use special JavaScript interfaces provided to them in order to obtain access to certain protected functions. Examples of platforms currently doing this are BlackBerry 10 WebWorks, Windows Phone 7 and 8, and certain Android applications.

While HTML5 offers a great option for course instructors, there are some disadvantages which instructors should be aware of. Occasionally some features can perform quite differently on one device to another. An example of this is the jQuery Mobile framework [14], which is used to rapidly build mobile websites and web-based applications. It features several fade effects in its user interface, which acts as a decorative animation when a user changes from one menu to the next. On Apple’s iOS devices it produced a smooth, professional looking result. On Android devices, it produced a flickering effect which annoyed both the users and developers. This was due to an underlying issue with how each browser handled video processing for fade effects.

5. CONCLUSION AND FUTURE WORK

The intention of this paper has been to help instructors see how their students think about new and emerging technologies and how it can enhance and further the quality of their courses. Since instructors are entrusted with the responsibility of educating the next generation of computer scientists and software engineers, it is imperative that they keep their courses open to new ideas and constantly look for ways to enhance their courses.

Mobile devices present instructors with a rare opportunity to re-examine old concepts, such as how to design software with limited resources, to large-scale distributed systems (e.g. “cloud” systems). The rapid turnaround time for mobile development gives students the opportunity to create applications which are useable in the real world and which students can publish on real markets. The sense of accomplishment this brings will hopefully motivate students to work harder and be less apathetic about courses, which often happens with many subjects in computer science and software engineering. Publishing applications to an “app store” may also motivate students to design and construct high quality software, as they want their name to be associated only with excellent products, which can then become part of their “software” portfolio, and hence greatly improving their chances of employment upon graduation. We recommend HTML5 for cross-platform applications.

References