CRITICAL EVALUATION OF APPLICATION PORTING IN MOBILE PLATFORMS

Kasra Madadipouya[1*]

[1]Department of Computing and Science
Asia Pacific University of Technology & Innovation
Kuala Lumpur, Malaysia

ABSTRACT

Smart phone has become increasingly popular and sales numbers from the third quarter of 2010 reported that an estimated 80.5 million smart phones were sold worldwide. The statistics expresses that mobile applications have grown significantly and mobile devices have become a part of the user's everyday life. As companies would like to see their mobile applications reaching a broader audience, the demand for porting applications is necessary before or after developing a software product. Software productions should make sure their products can be easily implemented in several environments, since it will allow the product to reach larger parts of the market in a cost-effective way. This becomes increasingly important when the target market is diverse, and is not eminently dominated by one software environment, such as the mobile market. However, the methods for making applications portable are often very unofficial and ad-hoc based with many constraints and difficulties. In this paper, we critically evaluated the current issues of mobile application porting from various angles. Then, we proposed a standard methodology with proper tools and techniques to overcome the difficulties of mobile application porting and increase the portability of mobile applications.

KEYWORDS: *Application development; cross compiler; cross platform; mobile applications; portability; porting applications*

1.0 INTRODUCTION

Nowadays, mobile devices are used widely and the usage of these devices growth continuously. Every day, new mobile device models with different and unique hardware/software specifications are introduced by different companies and released to the market. Furthermore, each released mobile device has its own platform. To be exact, it can be said that each device has its own specific platform even if they are running on the same Operating System (OS) because every device has its specific hardware configuration and the platform is matched based on the hardware (Acord, 2012). Therefore, developers and software producers need to consider and identify a way to support their applications for various types of platform by making them portable. Portability is generally considered as a desirable feature for a particular software product, especially in the ever-growing mobile market.

[*] Corresponding Email: kasra_mp@live.com

In the mobile application sector, one of the important facts which developers need to consider is, making software available for different types of platform in the market, which

is known as porting. They need to make sure that their products come with the portability feature and can be implemented and matched several environments easily, since it allows the mobile software producers gain broader market share in a more cost-effective way. This becomes increasingly important when the target market is diverse like mobile industry, and is not eminently dominated by one software environment. Although, portability brings many advantages to the mobile application development, it has surprisingly received little amount of attention. Mooney (1997) discussed issues in existing methods and measurements of portability, and stresses the fact that when companies develop software with portability in mind, it is mostly ad-hoc based. Moreover, he provided a general view of portability. He also tied portability to the software life cycle and discussed how it is measured, as well as how one can e.g. design, implement and test the software with portability in mind.

Portability issues can be categorized as performance and user interface limitations, the large differences between various platforms, and how input is handled which is covered in Section 2. Additionally, choosing specific programming language when aiming to develop a mobile application with portability as the main goal is another issue. Finally, there is a very few researches on portability for mobile applications, especially where results and data are obtained practically, which arisen the motivation of this project. In this paper firstly, the issues of portability of mobile applications are discussed. Then, in Section 3, a methodology for mobile application porting is proposed in details with the purpose of resolving the mobile application difficulties for software companies.

2.0 RELATED WORK

In mobile industry, producing compatible application has become big issue since used technology (hardware and software technology) in this field is growing fastly in line with high customer demand. Therefore, in order to bring satisfaction for the users and to stay in the market for longer time, the mobile application producers should develop the same application for different types of platforms. This requires more time and budget allocated which might be almost impossible to most of the mobile application producers because they are small companies and enterprises. As the solution, the best alternative for customer satisfaction as well as having better market is, to port applications for other platforms.

Generally, application porting refers to the act of developing software for different types of platforms without redoing the development phases or processes again. In other word, the developed application is adapted to the new platform(s) with little or no changes at all in the code (Mendoza et al., 2006). As mentioned earlier, the main aim of application porting is, to support software for different types of platforms and since in mobile industry many platforms (like Android, iOs, Windows phone) are available application porting rule is more significant and gives many advantages to the mobile application developers (Rapid Soft Systems Inc, 2012). These advantages are summarized as below (B.Orbital Technologies Inc, 2012):

i. Increasing market share especially for business applications.
ii. Customer satisfactions.
iii. Reducing the cost of application development like coding, documenting and so on.
iv. Reducing developing time.
v. Making applications development processes easier by reusing the available components and libraries.

Although, porting mobile applications brings many advantages for developers and software producers, it faces many constraints and difficulties as well, which are discussed in the following part. These difficulties can be categorized in two hardware and software issues (Comp.nus.edu.sg, 2008) which are described in the following section.

2.1 Hardware diversity

In mobile industry, many companies are manufacturing different types of the cell phones with various hardware characteristics and standards; unlike PC industry which most of the hardwares follow similar standards and have general characteristics and compatible to each other. The diversed hardwares in mobile industry make application porting process difficult and become the one of the hurdle of mobile porting. Different manufactured cell phones are, vary in display size, memory (RAM, ROM),processors types and speed (Stewart, 2012). As a consequence a ported application may work well on a particular model of cell phone but it might not work on other different models even from the same company. For instance, an application runs smoothly in a cell phone with 480 x 800 capacitive display size but it might not fit on another cell phone which has 240 x 320 display or causes unexpected crashes in the applications due to slow processor speed and memory shortage.

2.2 Software issues

Additionally, software issue could be another barrier for mobile application porting which can be categorized into two development tools and development techniques. Each available mobile platform has its own Software Development Kit (SDK) which contains especial programming language(s) and libraries (Stewart, 2012). Different SDKs may support various or single PC platform(s) as well. For instance, Android SDK utilizes JAVA programming language by default and official development language for Android platform and developed applications run upon Google's Dalvik virtual machine (VM) (Ehringer, 2010). Furthermore, Android SDK is supported and installable on Microsoft Windows, Mac OS X and Linux (Developer.android.com, 2012). In contrast, Apple's SDK (X code) uses Objective C as development language and just available for Mac OS X platform (Developer.apple.com, 2013). Moreover, each SDK has its own specific libraries which probably are not available in other SDKs with the same functionality.

In addition, mobile platforms are updated frequently (Mike, 2011) which double compatibility and portability problems because many libraries are deprecated and also new components are not available for the legacy systems (Shinder, 2010). Sometimes,an application is developed to be compatible with different versions of the same platform but in the older versions, it may continuously crashes because especial programming

mistakes and bugs are existed on OS which removed in the new versions not the old ones. As a consequence, it makes ported application futile.

Mentioned software issues make an application impossible to port to another platform or make porting process time and cost consuming. Often, some parts of the original source code need to be written again and those parts also must be tested in order to ensure their functionality and compatibility with other components. Therefore, making changes in the application code and testing them sometimes takes more time and budget than developing application from scratch (Burnaby, 2005).

Furthermore, the ported application may not have the functionality as the original one because all used libraries are not available or compatible for the target platform(s) and occasionally some application features must be removed (Burnaby, 2005). A clear example of unsuccessful application porting is angry birds game which ported from iOS to JAVA (J2ME) platform and resulted in having a poor ported game with different characteristics (different game levels) with slower game speed and delay in responding to the users actions which made the game barely similar to the original one.

On the other hand development technique could make mobile application portability difficult. In mobile application industry, there is no specific application development method and methodology available like general software industry (Agile, Waterfall, etc.) and therefore, each company develops software differently. Lack of appropriate methodology could arise problem in documentation as well. It means that mostly, complete and proper document is not written for the developed mobile applications and when it comes to porting an application for another platform, this lack will be more tangible since when an issue comes up, developer will need to refer to the document to overcome the problem and to solve it. For instance, many crucial libraries in applications are OS dependent and cannot be neglected. As a result, the development team must implement those libraries and since there is no proper and enough documentation available for the implemented libraries, porting becomes tough and almost impossible.

3.0 METHODOLOGY

Software is portable only when the cost of porting to a new platform is lesser than the cost of writing it from scratch. Otherwise, developing a software product will be safer activity and more cost effective. Hence, in designing a method for application porting, various aspects such as cost, feasibility as well as reliability should be considered carefully. The suggested methodology in this work is based on agile methodology which is basically a new approach to software development and has become widespread specifically for quick software design [need citation]. Agile methodology is very effective in dynamic environment which entails time to time modifications in customer needs and their expectations.

Abrahamsson (2007) explained the technological constraints which apply to various mobile platforms in the aspect of limited physical resources and the ever changing specifications. This is further explained by the existence of various devices with particular hardware characteristics and its unique operating systems. Other constraints associated

with mobile applications according to (Hayes, 2003) are evolving and inherent. The evolving constraints includes coverage, security and bandwidth which currently applies to mobile technology but are most likely to be resolved in the nearest future while the Inherent constraints includes reduced data entry capability (limited keypad could be an example), limited screen real estate, processing power, memory capacity and limited power reserve which are relative to desktop environments. Hence, various approaches should be considered to minimize the impact of the aforementioned inherent constraints. As a resul, the six phase method had been proposed based on agile methodology to overcome mobile application constraints and make porting feasible and more cost effective with less software defects.

3.1 Phase 1: Analysis of the suggested device

Prior to undertaking the porting process, experts need to analyze the operational system or platform which the device features. This includes both the application which is currently running and the application to be ported to. Moreover, comparison are made in terms of all the aspects which will affect the application performance including screen size and the memory or storage issues.

Moreover, on this stage, bottlenecks of the target system need to be analyzed along with the specification of the system which may make porting difficult. As a result of this phase, porting experts will produce a document which will specify the concerned differences between the platforms same as the eventual bottlenecks of the target platform or system.

For example, if an application is to be ported from palm device and operated by Palm OS 4 to PPC 2003, it should be considered that the screen size on Palm is 320x320 pixels, while on the PPC, its 320x240 pixels. Moreover, Palm OS is a single taking OS the CE on PPC is a multitasking OS.

3.2 Phase 2: Application redesign

Depending on the function of the application, its architecture can undergo significant modifications on different platforms. This modification may be required so as to enhance the development efficiency and optimize the eventual work hours when customizing the architecture to another platform.

Moreover, it is much faster and easier to redesign the architecture (because of the specific features of the platform) instead of reworking the existent architecture and changing it for another platform. Generally, such a rework will definitely cause nothing but makes the application operation to be unstable and eventually increases in the time spent on further rework. However, redesign is a way to eliminate application instability on a new platform. During this process, specific features of the platform are always considered.

3.3 Phase 3: Decision Making

This phase plays a crucial role of the successful completion of the above two phases. It entails making the right decision regarding:

i. Possible risk when porting.
ii. Estimation of effort and time required for porting.
iii. Calculation of the cost.

Moreover, at this phase, it is being confirmed that every team member has a clear vision of all the features, risks and restrictions which can be accrued from application porting and the specifics of the software and of the platform which the software will be ported to. Furthermore, it is necessary to take proper note of porting requirements of each and every client by creating a reference port and then decide the mobile devices and the OS platform to be used. Gap analysis may be performed to satisfy the client.

3.4 Phase 4: Graphics and Interface Design

This phase entails the comparative analysis of the devices and exploiting the results obtained from the first phase and pick up devices which are needed for graphics and user interface design. To be considered first are the screen size and color depth which are provided by the devices.

Since interfaces are normally different from one platform to another, the underlying functions in the program can be ported easily. But the main problem arises in the communication between these functionality and the interfaces when porting. A specific area of porting is the user interface which is generally known to vary in different environments.

Depending on the screen size, the user interface may have to undergo significant changes. So devices have touch screen while others don't have, therefore, when porting, these issues should be taken into consideration and rendered back in interface modification to make it convenient for the user. Attention should be paid to selection of colors because different devices are featured by different color depth. Furthermore, the RGB color pattern can vary dependent on the device.

3.5 Phase 5: Source Code Porting

This phase is the most labor intensive in the entire process of porting. The source code may vary dependently on the platform especially in the case in which the application is supposed to utilize API-functions specific for particular device or in the case in which the functionality provides for the device's hardware exploitation. Omitting some issues in this phase may cause impairment of the ported application. Cross compilers could be a proper solution to overcome the constraint. In this approach, application code will be ported to an intermediate code which can be translated into equivalent machine code sequences using a code generator to create executable code in target device.

The use of an intermediate code enhances the portability of the compiler. This is because only the code generator or the interpreter of the compiler itself needs to be ported to the target machine. Moreover, the remaining compiler will be imported as an intermediate code which will be further processed by the interpreter or the ported code generator thereby executing directly the intermediate code on the interpreter or the compiler software will be produced.

An interpreter is less complex; some are extremely easy to port than a code generator. This is mainly because it is not able to do code optimization due to its limited view of the program code. The following approach known as the compiler bootstrapping are feasible on the target machine in the porting process:

i. Assembly code by porting the interpreter.
ii. The source code of the code generator should be adapted to the new machine.
iii. Execute the source which is adapted using the code generator source as input.

Finally, the difficult part of coding the routine is usually done using the high level language against the assembly language of the target. Components of the program which needs a lot of conversion so as to work across different platforms needs to be isolated. Moreover, they should be restricted to only one area or component of the system. Some of these functionalities that are very different between the various platforms are file handling, media management etc.

3.6 Phase 6: Testing ported application

In order to speed up the process of testing and make it more efficient, various software automation methods could be applied in testing process. In addition to that, device emulators can be used intensively which are not only for sound time saving, but also enable the multi device support. Eventually, during the testing of the ported application, the following activities should be considered highly to ensure successfulness of software testing and the quality of the ported application:

i. Comparative analysis of the application performance on all platforms in order to ensure that porting is complete and no feature is omitted.
ii. Documentation and study out of differences which are derived from the target platform specifics.
iii. Bug tracking especially of those which are detected on the source device.
iv. Complete testing of application in order to check for new bugs. In this, particular attention is paid to tests which may reveal bugs not showing up earlier.

4.0 CONCLUSION

The high demand of mobile applications has led to the challenges of mobile application companies to come up with mobile applications which can run on multiple environments. The operating systems and devices vary same as the market requirements, the features of these devices, screen sizes etc. It is imperative for the mobile applications to be compatible across the multiple mobile devices. Hence at this juncture, mobile application porting becomes useful. An application developer has to make sure that the mobile application being released must be portable from the android to the iOS, windows phone or blackberry devices and vice versa. However, compatibility and porting issue could be a hurdle for the development team if the application porting is done ad-hoc. In this paper, we proposed a methodology based on agile model which can be adopted for mobile application porting by companies to improve the quality of the mobile application working as well as speed of the application porting.

REFERENCES

Abrahamsson, P. (2007). Agile software development of mobile information systems. Proceeding of CAiSE'07 Proceedings of the 19th International Conference On Advanced Information Systems Engineering, 1-4.

Acord, C. G. (2012). Cross-Platform Mobile Application Development: A Pattern-Based Approach, Naval Postgraduate School, USA.

B. Orbital Technologies Inc. Vancouver. (2012). Application Porting Services, Software Porting from Orbital Technologies - Windows, Mac, UNIX/Linux Porting, Retrieved from http://www.orbitaltech.com/application-porting.htm.

Burnaby, B.C. (2005). *How to Achieve Application Software Portability*. COTS Journal.

Comp.nus.edu.sg. (2008). Device Fragmentation of Mobile Applications, Retrieved from http://www.comp.nus.edu.sg/~damithch/df/device-fragmentation.html.

Developer.android.com, (2012). Android SDK. Android Developers, Retrieved from http://developer.android.com/sdk/index.html.

Developer.apple.com, (2013). Xcode – What's New - Apple Developer, Retrieved from https://developer.apple.com/xcode.

Ehringer, D. (2010). The Dalvik Virtual Machine Architecture, (1st ed.). Retrieved from http://davidehringer.com/software/android/The_Dalvik_Virtual_Machine.pdf

Hayes, I. (2003). *Just enough wireless computing*. Upper Saddle River, NJ: Prentice Hall.

Mendoza, A., Skawratananond, C. & Walker, A. (2006). *Unix to Linux porting*. Upper Saddle River, NJ: Prentice Hall.

Mike, I. (2011). Ice Cream Sandwich: A Deep-Dive Tour With Android's Chief Engineer, Retrieved from http://arstechnica.com/gadgets/2011/10/a-deep-dive-tour-of-ice-cream-sandwich-with-androids-chief-engineer/.

Mobile Pundits Inc, (2011). Mobile application porting. Porting mobile applications, Retrieved from http://www.mobilepundits.com/Mobile_Application_Porting.html,

Mooney, J. (1997). Bringing portability to the software process, Dept. of Statistics and Comp. Sci., West Virginia Univ., Morgantown WV.

Rapid Soft Systems Inc. (2012). Mobile Application Porting Services, Mobile Development Services, Mobile App Outsourcing, iPhone Software Development, Android App Development, iPhoneDevelopment, Android development. Rapidsoft Systems, Retrieved from http://www.rapidsoftsystems.com/mobile-porting-services.html.

Shinder, D. (2010). 10 things that can go wrong when you upgrade your operating system, Retrieved from http://www.techrepublic.com/blog/10-things/10-things-that-can-go-wrong-when-you-upgrade-your-operating-system/.

Stewart, B. (2012). Portability Study of Android and iOS, West Virginia University, USA.