A Rapidsoft Systems' White Paper

© June, 2009

# Mobile Software Development and Mobile Porting – Technology and Cost Issues

**Summary:** There has been big growth in the mobile applications in the last few years. But, before you begin on the path to develop mobile applications, there are a few factors that you must consider to minimize the development costs. We have been doing mobile development for over 20 years, and this has taught us a number of things. The first and most important factor is that a mobile development is quite unlike development on a PC. Based on our internal experience and technology expertise of over 20 years at Rapidsoft Systems, we discuss some of the factors that can help you plan your strategy for mobile applications including game development for rapidly emerging mobile market. This is a rather technical paper, but anyone can gain by reading it.

## Introduction

For mobile developers, one of the most important choices to make at the beginning of the development process is on which mobile software platforms this game should run. Although developers have constantly complained about the limitations of the mobile phone hardware and software environment, it is somewhat surprising to see that few of them actually choose the more powerful mobile platforms, such as Symbian C++, Visual C++, and .Net Compact Framework, to develop their games. Most developers focus on the relatively constrained Java 2 Platform, Micro Edition (J2ME) and Binary Runtime Environment for Wireless (BREW) platforms. The reason is simple: Those are the two platforms with the highest volumes and, therefore, the best support from operators.

## Choosing the right platform

### BREW

BREW is a C++-based smart client platform for handsets based on Qualcomm's CDMA chips. Its main attraction is that the platform integrates very well with the operator's services and enjoys automatic support for content downloading and billing. BREW is very popular among CDMA network operators in the United States. For commercial mobile developers who lack the experience working with wireless

operators, BREW is a good platform to start on, although its reach is limited. The number of CDMA network subscribers is only one-fifth of GSM network subscribers worldwide. In addition, the C++ programming language is difficult for many developers to master. For hobbyist developers, the tight integration between BREW and the operator makes it difficult to distribute free or personal BREW games over third-party Web sites.

### J2ME

J2ME is by far the most popular application development platform for mobile devices. It is primarily available on GSM handsets and is supported by all major GSM manufacturers, including Nokia, Motorola, Samsung, and Sony-Ericsson. In fact, J2ME is even supported on Nokia's CDMA devices and Motorola's Windows Mobile devices. The J2ME handset manufacturers have a combined market share of more than 80 percent, and J2ME is available on more than 700 million handsets. By the end of the year 2008, more than 150 wireless operators worldwide had deployed Java programming support. With automatic memory management and a rich set of easy-to-use APIs, J2ME is designed to boost developer productivity. On the other hand, the large market share and diverse device support for J2ME also present the biggest technical challenge for J2ME developers -- the problem of porting an application to a variety of devices.

### Android

**Android** is a software platform and operating system for mobile devices, based on the Linux kernel, developed by Google and later the Open Handset Alliance. It allows developers to write managed code in the Java language, controlling the device via Google-developed Java libraries. Applications written in C and other languages can be compiled to ARM native code and run, but this development path isn't officially supported by Google.

The unveiling of the Android platform on 5 November 2007 was announced with the founding of the Open Handset Alliance, a consortium of 48 hardware, software, and telecom companies devoted to advancing open standards for mobile devices. Google released most of the Android code under the Apache free-software and open source license.

### iPhone

iPhone is a very popular handsets with developers and users alike. With millions of users, iPhone is a popular device for mobile development. At Rapidsoft, we have a separate development team for this. The iPhone is an internet-connected multimedia smartphone designed and marketed by Apple Inc. Its minimal hardware interface lacks a physical keyboard, so a virtual keyboard is rendered on the multi-touch screen instead. The iPhone functions as a camera phone (including text messaging and visual voicemail), a portable media player (equivalent to an iPod), and an Internet client (with email, web browsing, and local Wi-Fi connectivity). The first-generation phone hardware was quad-band GSM with EDGE; the second generation added UMTS with HSDPA.

iPhone OS is the operating system that runs on the iPhone (both models) and the iPod Touch. It is based on a variant of the same basic Mach kernel that is found in Mac OS X. iPhone OS includes the software component "Core Animation" from Mac OS X v10.5 which, together with the PowerVR MBX 3D

hardware, is responsible for the interface's smooth animations. The operating system takes up less than half a GB of the device's total storage (4, 8, or 16 GB). It is capable of supporting bundled and future applications from Apple, as well as from third-party developers. Software applications cannot be copied directly from Mac OS X but must be written and compiled specifically for iPhone OS.

### Blackberry

RIM provides a proprietary multi-tasking operating system (OS) for the BlackBerry, which makes heavy use of the device's specialized input devices, particularly the scroll wheel (1999–2006) or more recently the trackball (September 12 2006–Present). The OS provides support for Java MIDP 1.0 and WAP 1.2. Previous versions allowed wireless synchronization with Microsoft Exchange Server's e-mail and calendar, as well as with Lotus Domino's e-mail. The current OS 4 provides a subset of MIDP 2.0, and allows complete wireless activation and synchronization with Exchange's e-mail, calendar, tasks, notes and contacts, and adds support for Novell GroupWise and Lotus Notes.

Third-party developers can write software using these APIs, proprietary BlackBerry APIs as well, but any application that makes use of certain restricted functionality must be digitally signed so that it can be associated to a developer account at RIM. This signing procedure guarantees the authorship of an application, but does not guarantee the quality or security of the code.

### Windows Mobile

Windows Mobile is a compact operating system combined with a suite of basic applications for mobile devices based on the Microsoft Win32 API. Devices that run Windows Mobile include Pocket PCs, Smartphones, Portable Media Centers, and on-board computers for certain automobiles. It is designed to be somewhat similar to desktop versions of Windows, feature-wise and aesthetically. Additionally, third-party software development is available for Windows Mobile. Originally appearing as the Pocket PC 2000 operating system, Windows Mobile has been updated several times, with the current version being Windows Mobile 6.1 and a future 6.5 release planned for release toward the end of 2009.

Microsoft projected in 2008 that shipments of devices with Windows Mobile will increase from 11 million to 20 million units, but it missed its initial goal in only selling 18 million licenses citing the delayed launch of certain smartphones. Windows Mobile's market share as an operating system for smartphones worldwide has fallen from 23% in 2004 down to 12% in 2008. Windows Mobile now has a worldwide smartphone market share of 14%. Microsoft licenses Windows Mobile to four out of the world's five largest mobile phone manufacturers, with Nokia being the other. Some current estimates suggest that 80% of the 50 million Windows Mobile devices made have been built by one contract manufacturing group, HTC, which makes handsets as for several major companies under their brands, as well as under its own brand. However, in February 2009 Microsoft signed a deal with the third largest mobile phone maker, LG Electronics, to license Windows Mobile on 50 upcoming LG smartphone models.

## Mobile Applications Porting Issues and Solutions

Although the J2ME API is relatively easy to learn, it is misleading to suggest that mobile Java language game development is somehow simpler than developing console or PC games. In fact, in terms of the

percentage share of the total revenue, the development and engineering cost for mobile games could be much larger than those for console or PC games. Most of the experts agree that the biggest (and most expensive) challenge in mobile applications development is the support of these many different devices in a fragmented market.

The J2ME platform makes the Java programming language and a standard set of APIs available across all the Java-compatible devices from multiple vendors. Ideally, a J2ME application developed for one Java handset should run without modification on all other handsets supporting the same APIs. However, Java's "Write Once, Run Anywhere" vision has yet to be realized, thanks to the fragmentation of the device market. Meanwhile, there are several reasons why there are so many phone models:

• Because mobile devices are highly personal, each one is designed for a very specific usage pattern. For example, enterprise users, consumers, messaging teenagers, price-conscious users, and hardcore gamers all get different phones with different combinations of functions and UI styles.

• Mobile phone manufacturers need to differentiate their product offerings. Different manufacturers adopt different CPUs, memory sizes, base operation systems, and UI features. They also provide proprietary Java extension APIs.

• Operators need to differentiate their service offerings, such as by customizing the device hardware and software. They can also decide to enable or disable certain data service on the device. For example, NexTel would not permit a third-party J2ME application download to general consumer phones on their network.

• Mobile phones are evolving faster than Moore's law, with hundreds of new device models on the market every year. Those devices support different versions of the J2ME standard and different sets of the optional API packages. Right now, the low-end MIDP 1 devices have the largest installed base. But many new MIDP 2 smart phones, including advanced devices that support the J2ME 3D, Bluetooth, and Web services APIs, are starting to take over the high-end market.

Supporting all the popular devices on the market maximizes the J2ME game's chance of reaching the critical mass for commercial success. A Java application needs to be ported, optimized, and tested for each target device it is intended to run on, which is a complicated challenge. For example, even among Nokia products, Series 60 and Series 40 devices have very different screen sizes, memory sizes, and CPU speeds. Furthermore, the quality control of the Java Runtime Environment (JRE) on devices has been weak. Different devices could have different bugs, particularly in their thread or memory management implementations. You have to work around those problems one device at a time. This can be extremely expensive and requires extensive technical know-how to optimize for more than 70-80 devices in house.

According to J2ME game development expert, today's J2ME game industry has original application developers and specialized porting houses. The original application developer typically develops a generic version of the J2ME game for a representative mass market device (or one version each for the high-end and low-end devices). If the operator is interested in the game, the operator arranges for a

porting house to optimize and test the game for all Java handsets it carries.

For beginning developers, we suggest starting from a high-end device with the least amount of computational and API constraints. Those devices allow the developer to focus on the correct API usage and game design instead of advanced CPU and memory optimizations. A Nokia Series 60 MIDP 2 device would be a good device to start with. Then, as you gain experience as a developer, it's a good idea to move toward more restrictive mass market devices. For example, most Nokia Series 40 devices only support 64 KB Java Archive (JAR) file size and 200 KB heap memory size limits. To port a graphic-intensive application from Series 60 to Series 40 requires reworking the graphics and even changing the game play. This step is absolutely essential, though, for the commercial success of the game. As discussed here, the great strength of mobile games lies in the large volume it occupies in the mass market. The low-end devices have by far the largest volume in the mass market. For instance, the Nokia Series 40 devices -- the most widely adopted Java phone ever -- have sold 10 times more units than Series 60 devices. The skill to understand the low-end devices and optimize applications for them is what separates novice developers from seasoned ones.

The Nokia Series 40 was the most frequently mentioned line of devices at the conference. Developers love it for its huge market share, but hate it for the amount of optimization work needed to port applications to it. For mobile game developers, it is important to master the skills to scale applications up from or down to Series 40 devices.

We get a lot of questions about game porting process. So let us clarify Rapidsoft Systems's methodologies here.

## Porting PC Games or Applications to Mobile Platforms

Let say if you have a simple 2D C++ game engine, and have a PC game based on that engine, and you want to port it to different mobile platforms BREW, J2ME, iPhone, Android, Symbian, etc.

Needless to say, that you want to get that game quickly available to mobile users. What is the best way to do it?

Do you need to re-code the engine and the game for each platform? Or is there an easier and more efficient way? Everyone knows that the process is complicated since different phones have different graphic/ processor/ memory/etc.

We often deal with customers and companies that have a catalog of Flash or other games and would like to port them to mobile platforms. Unfortunately, it is not that straight forward since Flash is not something that is available on phones. Moreover, phones have limitations that are very different then developing games or applications on PCs or MACs. Just because a game was simple to develop on Flash does not mean that it can be made available to mobile users with the same ease since development technologies on phones are very different and require a very different set of skills. Besides, developing the application logic and re-doing all its graphics elements and user interfaces – one must deal with the issues of mobile device porting – in other words making sure that an application or game runs flawlessly on a target handset. And, this must be potentially done as many times as there are device and operator combination. So, how do you manage this task – well – we explain this to you in the next section.

## Overview of Mobile Game or Applications Porting Process

There are several ways of attacking mobile game porting. First of all, until very recently it was mostly BREW and J2ME. The iPhone, Android and BlackBerry are changing this landscape and making the impossible task of mobile game porting even more impossible. The author worked in 3rd party mobile game development for many years until recently. In the last few years, we have seen less focus on BREW and saw publishers completely focus on J2ME as the cost of porting is strangling the industry. There are estimates to its cost, both time and money, and it seems to bell curve around 50-60% of the total development cost for each game is just porting.

At our company, we handle porting by having two engines that paralleled each other, one in BREW, one in J2ME. Most developers never requested Symbian port as Symbian development does not make any money. It is mainly for high-end tech demos that might be on one or two devices, nothing that could reach the mass market. Plus, most Symbian phones support J2ME.

We are required by publishers to provide anywhere from 7-23 reference builds of the game, targeting many different devices, in both BREW and J2ME. Just before moving on, publishers were also starting to require a J2ME touch screen reference version, and an iPhone SKU was being left as "to be determined" based on the final product and how cost effective an iPhone version would be at that time. The reference versions would then be passed on to a porting house to translate the different references to the thousands of other required SKUs.

Many big companies still use brute force their way through porting. That's why their games are constantly at a higher quality than the rest of the industry. However, it is just not possible for smaller companies to attack the problem this way due to costs. Not everyone can afford an office in Beijing with 5000 developers. That’s where Rapidsoft Systems can help in finding a suitable solution.

There are many companies, such as Mobile Distillery, out there developing engines to cut porting costs. It is hard to vouch for their efficiency without using it. So, we can't vouch for such engines. The problem here is that you will be at the mercy of other companies’ engines. Performance could be problematic due to the fact that it is being built to target thousands of SKUs. Plus, you really have little control over the low level implementation of your game in this instance. The end result seems to be a game that targets the lowest common denominator of phones.

Finally, a lot of developers are just abandoning the idea of supporting all mobile platforms. There is a huge flood on games on the iPhone because 1) it requires only targeting one platform and 2) there is a 70 percent profit share through the AppStore for developers. Through carrier releases, the percentage is not even comparable.

### Solving Multi-Platform Puzzle

The problem of developing an application that runs on multi-platform is not unique to mobile software since the problem existed in the server domain too but to a lesser degree. Mobile applications simply make it more complex. One well tested and time proven methodologies is developing hardware abstraction layer. At Rapidsoft Systems, what we have used for multiplatform development is to implement a hardware abstraction layer. The engine is coded in C++ and provides an application level interface to implement the system calls. Interface is written both in C and C++. This allows you to use

full-fledged C++ for your game and engine and link with the system abstraction written in whatever language your platform needs. Symbian doesn't support 100% of the features of C++, and still has a few bugs, and iPhone API uses Objective C. C is compatible which most of the platforms that we deal in mobile space (well, not Java). Linking C is easier than C++, as there are less problems.

Implementing an additional interface in C is a little slower, but helps you a lot when porting it to other platforms. Also, it allows you to have a Win32/ Linux/ Mac build besides the Symbian, BREW, etc. Hands on experiences shows that the debugging capabilities of Symbian and N-Gage platforms are signifficantly behind Visual Studio or GDB. iPhone, on the other side, has a lot of cool tools to debug and profile your app. So, development times and porting times often relate to the debugging capabilities of underlying phone.

The platforms you are considering are not compatible, but most of them allow you to run C/C++ code. Hence, in theory you could port the engine to some Standard, such as ANSI, or C-99 and it would compile in most of the platforms PC, BREW however this does not take in account the libraries your engine might need. Game development has its own complexities since games exploit audio/ video capabilities of underlying system. For example, if your game engine uses OpenGL then it would work in the PC, and some consoles, but on Symbian devices you need OpenGL-ES which is not exactly the same, so you need an abstraction for all libraries you use. This is not as easy as it might first seem. This adds to the cost of development for multi-platform mobile games.

About J2ME and Android - they are Java platforms, so no C/C++ can be run there without any special VM library at least. In this case you need to port the C/C++ code to Java which can be overkill and is a complete re-write of your game.

So an expert's answer to this is while you can make an abstraction to your libraries (libs) and code using standards you might be able to use the same engine in several platforms as long you can use the same compiler for them. And maintaining this is not always easy.

Take a look for example to this engine CubicVR it allows you to compile the same engine for PC/ Linux/ MAC/ iPhone (maybe)/ Sony PlayStation Portable device.

If your engine is built on top of some 'platform independent' framework such as Java, then it's supposedly easier than if it's written in a lower level language. However, neither Java nor Flash is, for example, supported on the iPhone/ iPod touch, and it'll probably take a while until they are. On the other hand, the only SDK available for those platforms is Objective-C, which, guessing wildly, you didn't use to implement your engine.

In general, it depends on the application/game engine you've written. Most likely you'll have to change something, as very rarely are all features available on all platforms. J2ME graphics is a classic scenario of 'platform independent'-dependence, or so I've heard. How much you'll have to change solely depends on how portable your code is, i.e. how well did you separate out the parts that potentially need changing.

## Cost of Mobile Applications Development and Porting

There are 3 types of Ports (that we need to consider):

**A) Standard platform -> Mobile**

Here the task is to port a PC game and make it available to mobile phones. Of course, there are many mobile platforms; we can choose PC to J2ME Phones porting. This is essentially a re-write of applications in most cases. This can be an expensive exercise depending upon the level of complexity of the game or mobile application.

**B) Mobile Standard -> Mobile Standard (Mobile Platform Change)**

Here the task is to port games written for one mobile platform to another mobile platform. For example games written for J2ME platform to be ported to Brew handsets. Again, because of incompatibilities between the system, one has to write the application from scratch. This can be an expensive exercise depending upon the level of complexity of the game or mobile application.

**C) Mobile Phone -> Different Mobile Phone (Simple Application Porting)**

This case involves porting mobile games or application between different phones. For instance, porting a J2ME game over a range of 100 J2ME handsets falls in this case.

In above scenarios, Case C is usually comparatively easy and cheap; B is a bit more time consuming and expensive, since an application must be often re-written, but A is the most expensive. A typical cost cited is about $2.5K for a case between B and C. A involves the creation of a new product from a brand/ design which is more then just 2.5k, but in comparison of a PC game development, it can be thought of a "minor" fee.

The simple calculation shows the cost of doing 50 reference builds with 20 languages. making a feature of the "ports" to support multiple languages in the application, necessitating only the price of translations per port. The calculation below shows the cost:

20 (languages) x 50 (handset) x 2500 (Ref build cost) = $ 2,500,000

This number can be reduced a bit even more by developing apps that automatically adjust to the phone they are put on, (A multi-port will usually involve phones w/ similar speeds, and screen size)

So if the 50 popular phones are compressed down to 10 ports or less (5 phones a piece on average is very possible, although there will be ports for "special" phones (1-3 phones), and some with 15+ phones). The calculation below shows the cost:

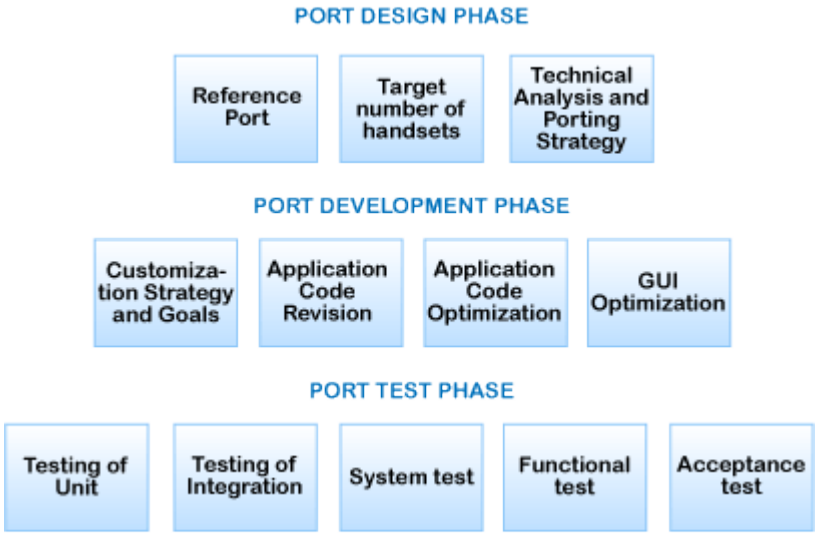
20 (languages) x10 (handset Groups) x 2500 (ref build cost)= $500,000

Finally, now for the downside, the above numbers don't include the initial port cost of moving the "game" to that specific platform. Also, developing for these spectrum of devices is not an easy task, and a serious task force is required just to learn the details of each device (through extensive internal testing, and documentation).

No one can't really give a recommendation about how to handle this. If your group doesn't do Mobile game development already, we suggest finding an experienced contractor(s) / group like Rapidsoft Systems to do the initial port with specifications that it should support multiple languages, and automatically adjust to a certain initial multi-port. Then use the same company to do the porting to other phones, or find a local company to do the rest of the porting, if cheaper.

If you already do Mobile gaming in the field that you are doing porting, then it comes down to how much internal investment you want to commit to the project. There's a speed/money issue with doing the research in house, but conversely, relying on a outside source may also be problematic if not selected properly.

To work with a vendor, you need to know the porting process deployed by them. The diagram below shows Rapidsoft Systems' porting process. Understanding of this process will help in working with the vendor.




## Managing Process and Cost of Mobile Porting

The key element in managing the cost of porting is analyzing the commercial potential of your mobile applications. Whether you are planning to develop a latest mobile application or a trend setting mobile game, you should try to formulate answers to the following questions:

1. Do we need a port on every handsets that carrier offers or only on those that are really popular with the users? The data may be difficult to get buy, but your service provider or porting vendor should be able to help you in this.

2. Within a device set, do you need a port to be tested on every device or only some of the devices? For instance, do you need to test your blackberry port on every variation of Blackberry 8800, Blackberry 8900 etc., or you can test only on the new models.
3. Do you need to support multi-platform like Blackberry, Android, iPhone, J2ME, Brew or Windows Mobile or you can wait to see the result of first making application ready on one or two platforms? This is an important element in managing the cost. If your application does well on one platform, the revenue may be funneled to do other platform ports.

## Conclusions

Mobile application distribution has become a lucrative industry with a manifold increase in mobile subscriptions and the number of avid mobile gamers around the world. However, success depends on reducing the cost of development and porting and timely execution of the development and application porting.

Use of external porting can cut down the cost and time to market applications. While it is possible to create an internal porting capabilities for most companies that option will be quite expensive since it requires significant man power efforts that are cyclical in nature. This means using a trusted partner like Rapidsoft Systems to develop and port your applications can be the best solution for you.

For more information and specific questions, please contact us at:

**Rapidsoft Systems, Inc,**
Mailing Address: 7 Diamond Court, Princeton Junction,
New Jersey 08550, USA

(Princeton) New Jersey, (San Jose) California, Delhi/ Gurgaon (India), Mumbai (India), Chennai (India)

**Web:** www.rapidsoftsystems.com

**Phones**: **1-609-439 4775 / 1-609-439-9060 (US East Coast, NJ Office)**
**1-408-829-6284/ 1-408-890-2509 (US West Coast, San Jose Office)**
**Fax: 1-831-855-9743**

**Email:** info@rapidsoftsystems.com