GFB Consulting

Internet & Telecoms Consulting

A Strategy for Cross-Platform Mobile Development

Author Gaddo F Benedetti
Date 2 October, 2010
Email gaddo@gfb-consulting.com

TABLE OF CONTENTS

Legal Notice

Introduction

One of the principle challenges facing firms seeking to operate in the mobile sphere is a balkanized landscape where it comes to the operating systems used and, in turn, the development tools and resources required to cater for them all.

Even only eighteen months ago, many development strategies opted to concentrate on the iPhone, given its powerful position. However, with the advent and strong growth of Android based devices, this paradigm has been questioned by many as they see an increasingly heterogeneous market in the future.

The problem of course, is all of these operating systems require differing development resources. Native Symbian and MeeGo applications are written in C++, Android and Blackberry use Java and iPhone applications are not only written in Objective C, but require that development be carried out on an Apple operating system.

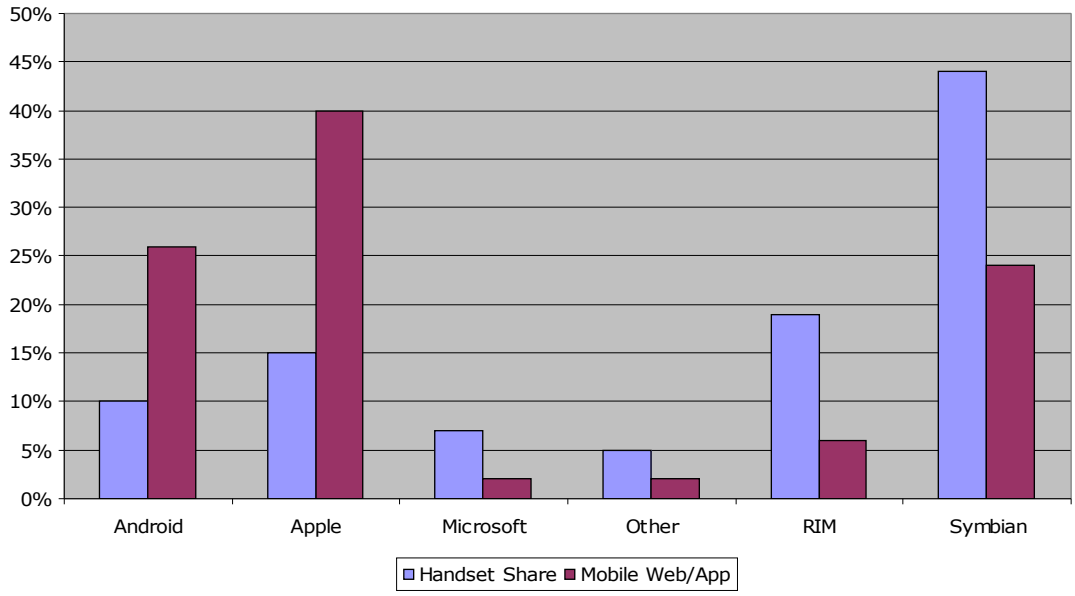
Naturally, this presents a dilemma for all but the most resource rich of development firms, as it would require different IDE's, hardware and, most importantly, development skill sets to develop for all of them.

What this whitepaper examines and proposes is a development strategy for this changing technology landscape. Principally it looks at what are the principle mobile operating systems out there and how one may adopt a cross-platform development solution to best cover these operating systems, while minimizing the resources necessary to do so.

The Present Mobile Landscape

Presently[1], the distribution of mobile operating systems in terms of mobile Web usage and market share has changed significantly worldwide. While the iPhone, with its iOS, remains the market leader, Android has grown rapidly – although largely at the expense of Symbian.

Symbian still remains a key player, as does RIM, but these are decidedly relegated to the role of tier 2 operating systems, after the iOS and Android. All other operating systems (tier 3), including Microsoft, account for the smallest segment in both terms of handset distribution and mobile Web or application footprint.



Sources: Gartner Q1 2010: Market Share[2] (handsets), AdMob Operating System Share, May 2010[3] (mobile Web / applications).

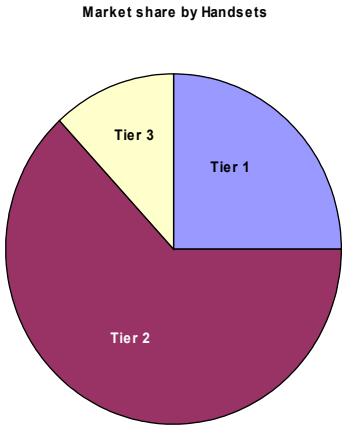
Now while handset distribution may still heavily favour Symbian and RIM (with a combined 63% of the market), *mobile Web and application market share is a more important metric to consider from the perspective of ROI*. In this regard, the tier 1 (Apple and Android) dominate two-thirds of the market, and if one goes

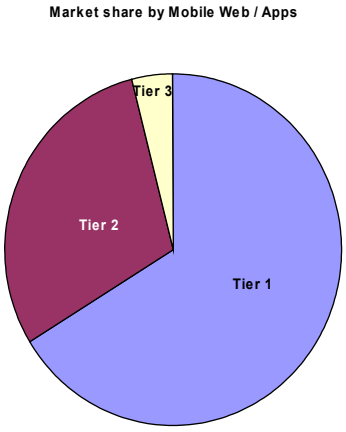
[1] September 2010

[2] http://www.gartner.com/it/page.jsp?id=1372013

[3] http://metrics.admob.com/wp-content/uploads/2010/06/May-2010-AdMob-Mobile-Metrics-Highlights.pdf

so far as to include the tier 2 operating systems (RIM and Symbian), this brings us up to an overwhelming 96% of market share.





As a result, the market has developed to the point that developers can no longer ignore the tier 1 operating systems, and must take them as a minimum target for development. For more comprehensive coverage, tier 2 should also be strongly considered, while the final tier could be discounted unless your market specifically bucks this curve.

It should be noted that this market landscape is unlikely to retain these ratios in the long term. While the tier 1 operating systems are presently dominant, the others are presently in a process of renewing their strategy roadmaps. Depending upon their success implementing these, we are likely to see some changes in the next 12 to 18 months.

Finally, as an aside, it is also interesting to note that both Apple and Android are also the only two players whose mobile Web / application share outstrips their actual handset market share. I would cite this as an early and key indicator for market growth in the smartphone market and so if it can be observed in any of the non-tier 1 operating systems, this would likely point to a future growth in market share.

A Cross-Platform Development Alternative

Up until relatively recently two platforms had emerged for cross platform development on the different operating systems; Adobe Flash for mobile[4] and Java Platform, Micro Edition (J2ME)[5]. Unfortunately, Flash was specifically targeted for exclusion from the iPhone[6] in the last year and J2ME, while still the most universal of mobile platforms for applications, has fallen out of favour, largely due to poor support on newer smartphones and lack of innovation in the technology since Sun's acquisition by Oracle[7].

In this vacuum, a number of alternatives have arisen, such as Unity[8], Airplay[9] and Titanium[10]. Of these, the last appears to be the most promising for various reasons; as an open source project, entry is low cost and has a vibrant developer community already in place. Secondly the language used for development is JavaScript, a language that is already popular in Web development which would allow for a shallow learning curve for non-mobile developers.

Another advantage of JavaScript is that this is the language of choice for another emerging development platform, mobile widgets[11]. These are essentially Web applications (written in a mixture of JavaScript, HTML and including images) that are packaged up into a zip archive. Presently Symbian already supports widgets, which it calls Web Runtime (WRT)[12]

[4] http://www.adobe.com/devnet/devices.html

[5] http://www.oracle.com/technetwork/java/javame/tech/index.html

[6] http://www.dmwmedia.com/news/2010/04/29/apple-ceo-jobs-explains-iphone-flash-ban-open-letter

[7] http://www.theregister.co.uk/2010/09/20/java_me_loses_to_android/

[8] http://unity3d.com/

[9] http://www.airplaysdk.com/

[10] http://www.appcelerator.com/

[11] http://www.quirksmode.org/blog/archives/2009/04/introduction_to.html

[12] http://www.forum.nokia.com/Develop/Web/

applications and has left the door open to their use with MeeGo, as long as a native wrapper is placed around the widget[13].

On the positive side widgets are quick to build and appear to the user as any other application, although as negatives they are not compiled and so the source is potentially open to view and modify and as they are interpreted, performance will naturally suffer for CPU intensive applications.

Visible source code can certainly be an issue, and for this reason widgets are probably not recommended for applications that are commercially sensitive. Neither are they or Titanium terribly suited towards graphic intensive applications either – for that, the Unity platform is specifically geared towards this end and may prove a better alternative.

However, where high-end graphics and compiled code are a secondary consideration, a combination of Widgets and Titanium present a viable development platform solution, especially given their common use of JavaScript.

Between them they can be utilized to cover Symbian, iOS and Android – 69% of the handset market and 90% of the mobile Internet / applications market. Additionally, Titanium has recently introduced support for RIM, allowing also Blackberry development – bringing us up to 88% of the handset market and 96% of the mobile Internet / applications market – as well as including support for desktop operating systems and, most interestingly, the iPad.

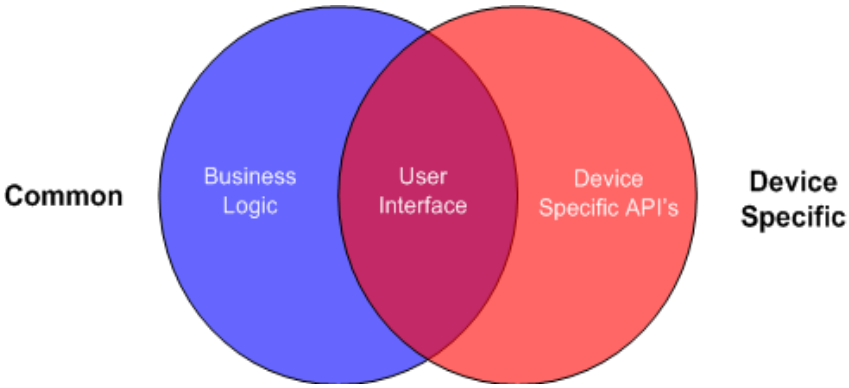
If considering a combination of Titanium and widget (Nokia WRT) development so as to bring your application to as broad a market as

[13] http://wiki.forum.nokia.com/index.php/Porting_WRT_widgets_to_Qt_applications

possible, one has to first though consider the limitations of cross-platform development though.

Cross-platform development is never actually going to be fully cross platform. Different operating systems and devices will inevitably have their own specific API's or will simply differ in terms of capabilities or even screen size. As such it is important to remember that you will *never* get a one size fits all solution and will need to find a best fit solution to minimize the customization that needs to be applied after your core application is built.

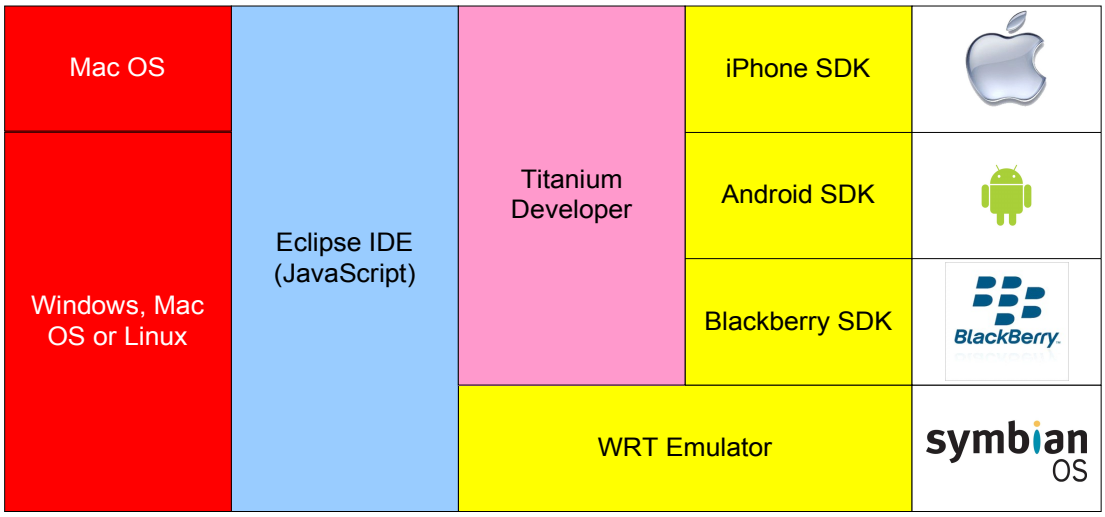
For this, we can broadly divide our application code and other assets into three basic areas; business logic, user interface (UI) and device specific API's.



Business logic will tend to be purely computational; algorithms that can be packaged into libraries and reused on all platforms without any need to amend them. At the other end of the scale are device specific API's, that would access functionality or protocols specific to the device or operating system being accessed, and these will always have to be written specifically for each device and/or operating system.

In-between is a grey area related to the UI. There some elements may be common for all devices while others may be very much device specific. In some cases, a *lowest-common denominator* approach may be adopted so as to reduce device/OS specific development; however care should be taken so that such an approach does not negatively impact upon user experience – sometimes to the point that the application becomes unusable, as we learned with WAP a decade ago.

Nonetheless, if care is taken, an application can designed to a point where device/OS specific development is minimized, and largely limited to UI issues.



The proposed development environment could be set up as shown above. Development and testing of iPhone applications still requires use of the Mac OS, while the other platforms have no such limitations, thus at least one Mac OS development machine would be required (Dual-boot systems and OS emulation are also options).

Eclipse is a widely used and respected development IDE and has been chosen in this model as it can be customized to make cross platform development smoother. Beyond this, the Titanium Developer platform

would then be used as the compiler for all but the Nokia WRT application version. Finally, the relevant SDK's and emulators would sit on top of all of these, either to allow Titanium to compile to native code and/or for emulation and testing purposes.

Please note that while the Blackberry is included in this model, Titanium support for this platform is presently only at beta stage.

Ultimately, using a combination of these two platforms a developer can potentially minimize the resources and development time required to produce applications for the vast majority of the current smartphones on the market. It's not a magic pill solution to the problem, however for many development houses, with limited resources, it can make cross-platform support a realistic proposition.

Conclusions and Caveats

In summation, what this document proposes is an approach to development for multiple-mobile platforms that uses two existing technologies; Titanium and mobile widgets (in the shape of Nokia WRT applications).

Not everything would be cross-platform; device specific functionality and UI elements would still have to be specifically developed. Additionally, iPhone application testing would still require the use of an Apple operating system. However, the bulk of the work would be eliminated.

It is also important to reiterate that this solution is not suitable to all forms of mobile application development. Use of mobile widgets is unsuitable for applications where source code is commercially sensitive. Also, neither widgets nor Titanium are presently suitable for graphically intensive applications. However, where it is suitable is with applications that hold their 'value' remotely and act largely as clients – the *Facebook* mobile application for Nokia, written as a WRT application, is a good example of this.

Another important caveat falls into the realm of commercial politics. While Apple has recently relaxed their restrictions on non-Apple development tools[14] to allow for platforms such as Titanium, there is in theory no reason that they may not change their minds in the future. Given this, commercial and regulatory pressures are going in the direction of more, and not less, openness with development platforms and so while this danger remains, I would judge it to be acceptable.

A future resurgence in the functionality and support for J2ME should also be taken as a future consideration. However, while Oracle has finally laid

[14] http://www.apple.com/pr/library/2010/09/09statement.html

out its roadmap for this[15], even in a best case scenario it will take at least 12 to 18 months for the effect of this to be felt on the market, let alone warrant adoption.

A final caveat rests upon the actual range of mobile operating systems that this solution would support. While my defined tier 1 and 2 operating systems presently represent the vast bulk of the market, there is certainly no guarantee that this will remain the case and others, in particular Windows Mobile, may regain sufficient market share in the future to make its exclusion unattractive. Additionally, this is the worldwide state of mobile operating systems; national, regional and specific demographic differences vary and many projects would not necessarily benefit on the same level.

Overall, while not a complete cross-platform development solution, the use of a single and simple language, where a significant proportion of source code would be platform neutral, a single IDE and a minimum of platform specific resources are at present the optimum solution to the resource conundrum that faces many mobile development projects.

[15] http://www.theregister.co.uk/2010/09/26/mobile_java_oracle/

About the Author

Gaddo F. Benedetti has been working in the area of IT consultancy since 1995, originally developing in-house database solutions and software tools with a number of clients. Later Gaddo was Wireless Technologies Manager to Labyrinth, a subsidiary of British Telecom. The role was a combination of business analysis and consultancy, and as a result he was technical lead for the Digifone On Line WAP portal in 2000, as well as writing numerous articles and a book on the technology.

From 2002 to 2005, Gaddo was a director of mPerium, a mobile consultancy firm, whose clients included the IFA Telecom, Thomas Crosbie Media, Champion Sports and Irish Music Magazine. Thereafter he held a number of roles management or consultancy roles throughout Europe, for the Vodafone group in the UK and Red Circle Technologies (now Zamano plc) in Ireland.

Gaddo works on a project, contract and (occasionally) permanent basis, for public and private sector bodies throughout Europe. His focus remains in both mobile and fixed line Internet related technologies, but importantly only in how they can complement a business rather than the reverse - from start ups looking to apply the right technology to their business model, through to multinationals and government bodies seeking to leverage their assets in the most efficient and cost effective manner. He also regularly writes and speaks publicly in these areas.

If you are interested in speaking with Gaddo on what he can do for you or your organization, please feel free to contact him by email at gaddo@gfb-consulting.com.