



**Harwell Innovation Centre
Building 173 Curie Avenue
Harwell Oxford
Didcot
Oxfordshire,
OX11 0QG**

**+44 1235 838 531
www.redskiessoftware.com**

Development of mobile applications for multiple platforms

By Darren Oliveiro-Priestnall, Executive Technical Director

Date: 14th October 2011

Version 2.1

Contents

Introduction	2
Problem Statement	2
Separate platform applications approach	3
Shared code base approach through MonoTouch and MonoDroid	3
Shared code base through application WebView approach	4
Extending all three approaches to include non-Mobile applications	5
Conclusions	6

Introduction

In 2010, over 300,000 applications were downloaded over 10.9 billion times, a figure predicted to rise to 76.9 billion downloads in 2014, generating US\$35 billion in revenue (IDC), up on \$7.3 billion in 2011 (Canalys).

As the mobile application market continues to grow, so does the need to support the widest number of platforms to ensure applications reach the widest possible audience while giving the maximum return on investment to application owners. This is not an easy challenge to meet, especially since the mobile application market is still relatively new and is subject to rapid changes between the popularity of platforms.

Gartner predicting that by the end of 2012, Android mobile devices will account for 49.2% of the market share, largely replacing the current Apple iPhone user base. Other sources suggest a dramatic increase in Windows Mobile devices with the launch of Windows 8, thanks to the tight integration with Microsoft cloud technologies, office applications and enterprise level software such as SharePoint and Dynamics. At the same time Apple is continuing to innovate, Blackberry is rethinking its own offerings and as of yet new platforms will likely surface as the revenue potential from this new market continues to be demonstrated.

Problem Statement

With so many new platforms, where does this leave the application developer who wishes to support the largest number of available platforms?

Developing software to run across multiple platforms can be a complex and costly affair, both in terms of initial development costs as well as in the long term

maintenance of so many separate applications.
This problem isn't only constrained to mobile applications but also extends to all interfaces to the business whether that's the company website, its Intranet and Extranet or any specialized desktop software or distributed widgets

As mobile application usage continues to grow and desktop software sales continue to decline, the importance of managing the development, branding and distribution of company applications will increase.

This white paper covers three approaches to mobile application development which address these challenges in different ways. All three approaches are used daily on projects by Red Skies Limited, applied depending on the nature of the application being developed.

Separate platform applications approach

The most common approach to supporting multiple platforms is to develop independent applications for each platform.



This is a practical approach when an organization intends to target just one or two platforms or when an organization needs to optimize an application for a device, making the most use of specific devices API's. Gaming is a good example when every last clock cycle needs to be used on a phone to make a game run

smoothly. However, frameworks such as Unity3D (<http://unity3d.com/unity>) are even making cross platform development of games a more cost effective approach while still maximizing the benefits from a device's graphics API's.

Unity3D allows stunning 3D games to be developed quickly and deployed across iOS and Android devices as well as PC, Mac, Nintendo Wii, Sony Playstation, Microsoft Xbox and the web using a single shared .NET code base.

With this approach it is still sometimes possible to share a subset of code between a mobile application and other non-mobile applications.

For example, assemblies written for Windows Mobile can be used in other .NET applications such as a company website, WFC web service, a Silverlight application or even desktop WPF application.

With careful consideration up front it is still possible to make some gains through code reuse even when writing separate applications for each mobile platform.

Shared code base approach through MonoTouch and MonoDroid

Red Skies Limited started out as a .NET specialist, working as a Microsoft partner providing solutions across all Microsoft platforms including the web, desktop, mobile and embedded.

Building on the cross platform strengths of .NET, we have gradually extended out to include an increasing number of non-Microsoft platforms into our field of expertise while maintaining code reuse across all supported platforms.

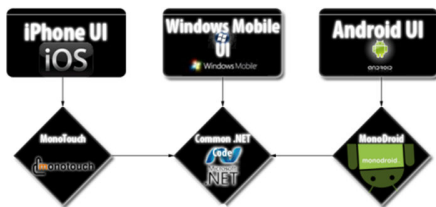
How does a .NET company share code with Android, iOS and Windows Mobile when all three platforms use different languages? When a high level of processing is required on the handset but a company wishes to use

a shared code base, we provide solutions developed around the MonoTouch (<http://ios.xamarin.com>) and MonoDroid (<http://android.xamarin.com>) frameworks.

These frameworks allow code to be developed in .NET but rather than compile them into assemblies, they are compiled into the code required for the individual handset.

This is architected by designing the software to move as much logic as possible into a shared class library which can be deployed across all three platforms as well as any non-mobile applications that could use the code.

Mobile specific content is then moved into separate thin classes, greatly reducing the amount of platform specific code.



The UI is still designed separately for each platform giving the overall result of a true mobile specific application while backing onto a common code base.

This achieves our goals of reusing code between platforms, reducing development and maintenance costs and giving access to three primary mobile platforms.

Where an application is needed to run on multiple handsets even when there's no connection to the Internet, this approach provides a very cost effective, attractive and engineered

approach to mobile application development.

New features can often be implemented once in the shared code base and picked up across all applications.

Shared code base through application WebView approach

Every mobile platform OS supports an embedded webview control for viewing web content from within an application, making it possible to develop applications that can be shared between all mobile devices including non-SmartPhone devices.

By developing skeleton applications for each platform containing an embedded WebView control set to run full screen, the UI and much of the logic can exist on a company server with additional logic running in Javascript on the handset.

To add new functionality or content, this approach makes it possible to make updates to a shared web server with all applications automatically picking up the changes as they run. Of course, moving processing to remote servers adds a dependency on web connectivity and a fast connection from the mobile device.



If a reliable connection exists then then approach can be a very fast way to develop and maintain mobile applications across different mobile platforms.

Software such as iWebKit and BlackBerry HTML UI allows web interfaces to be styled to match the phone's native applications while software such as jQTouch and Sencha



Touch provide dynamic JQuery effects for such mobile applications.

A number of frameworks now exist to extend this concept by providing additional processing, storage, synchronization and presentation features on the device using web technologies including HTML, CSS and Javascript while simplifying access to local mobile data and hardware features.

The result can often be greatly reduced development and maintenance costs since future updates often don't require any modification to the applications themselves.

Rhodes is a cross-platform smartphone application framework from Rhomobile (www.rhomobile.com) which supports iPhone, RIM, Android, Windows Mobile and Symbian. Mobile software is written in HTML, CSS, Javascript and Ruby and allows a single application and interface to be developed and deployed across multiple platforms. Rhodes utilizes an MVC style pattern similar to Ruby on Rails and includes a local Object Relational Manager (ORM) called Rhom for local data storage as well as RhoSync for synchronizing data with remote servers including remote web services.

PhoneGap (<http://phonegap.com>) is an open source framework which supports iPhone, Android, BlackBerry, Palm webOS and Symbian WRT (Nokia).

Software is written in HTML, CSS and Javascript and is ideally suited to expert Javascript developers. There is no direct support for synchronizing with remote servers or for local data storage so data storage is restricted to support provided by Webkit's Web Storage features. PhoneGap also takes advantage of HTML5 to provide rich interfaces but requires a high level of Javascript knowledge to work with effectively.

Titanium Mobile (www.appcelerator.com) allows applications to be developed in Javascript and subsequently compiled down to native code and components using the toolkits compiler.

However, this approach does rely on a reasonable Internet connection to pull pages from the server and does not make optimized use of mobile hardware. For applications that depend on live data such as blogging, forum, newsfeed, live news and other such apps then this becomes a very attractive option reaching the largest audience with the smallest overhead.

Care also needs to be taken to ensure the platforms usage policies are not breached by this approach. What content can be displayed in a web view can often be regulated in a platforms terms and conditions and so it is vital to check on any content regulations for a specific platform.

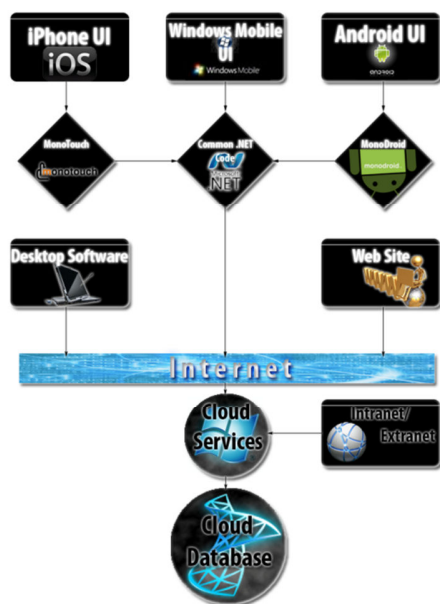
Extending all three approaches to include non-Mobile applications

Regardless of which mobile development approach is adopted, it is often useful to integrate an application with other remote services across the Internet.

One example might be to share web content between a company website and company mobile applications.

Another example might be feeding live Microsoft Dynamics or SharePoint content out to and back into company mobile devices. Yet another example might be authenticating users through secure web services or using tracking data to locate salesmen in the field.

The diagram below demonstrates one configuration which uses a MonoTouch and MonoDroid solution backed onto a shared code base which manages a secure connection to an Azure cloud service which in turn manages communication with a company Intranet and/or Extranet.



Of course cloud services could be replaced by WCF services running on a Windows Server box or virtual instance or any other web service, the cloud database could be a SQL Server or other database and the Intranet/Extranet link could be any third party system of use to the mobile application.

Conclusions

While developing individual applications for each platform is still a viable approach to application development, the increased range of platforms is making cross platform frameworks increasingly important to reduce development and maintenance costs while maximizing an applications target audience.

Quicker Development Time

Sharing code across multiple platforms allows core functionality to be written once rather than re-written for each platform. This not only leads to a more rapid development process but also greatly reduces the development time for future applications which can reuse the shared libraries.

Improved maintainability

The software development lifecycle rarely ends when software is deployed, with new features and extensions often being required to be developed over time. By applying one of our cross platform development approaches, new core features can be written and picked up across all platforms.

If using the MonoTouch or MonoDroid approach to development, code developed can be used across non-mobile MS platforms whether for the web, desktop or even embedded platforms.

It is also anticipated that the Mono family will be extended beyond iOS and Android to include other platforms, further increasing the shared code base opportunities.

Where code is shared, testing schedules become shorter, both for any initial development and for future applications. Since MonoTouch uses .NET then it becomes possible to integrate unit testing and run Test Driven Development over the whole system.



Benefitting from the enterprise level .NET platform through Mono

Core to this approach is the .NET platform, an enterprise level framework installed into Windows and used across industry from embedded systems to banking transaction software.

.NET allows software to be written in tens of languages and used across tens of platforms and is trusted in every industry.

Grow with your business needs

Our shared code base approaches provide a highly scalable and flexible approach to development, allowing a software family to be developed gradually as the needs of a business grows. A single mobile device application or a single web application could be developed to begin with and launched as a highly effective product. At a later date, code developed for the initial application and held within the core library can be re-used in other products, reducing development time and providing a level of consistency and trust in the code which would already be in daily use.

While some development companies may specialize in a single platform, only we are able to provide a true cross platform experience from enterprise server side services and secure interaction with a corporate network through to hand held devices such as Android, iPhone/iPad, Windows Mobile and BlackBerry.