WPF and Silverlight; Unifying the Development Platform for Desktop, Web and Mobile

A whitepaper by Colin Eberhardt, Scott Logic Ltd.

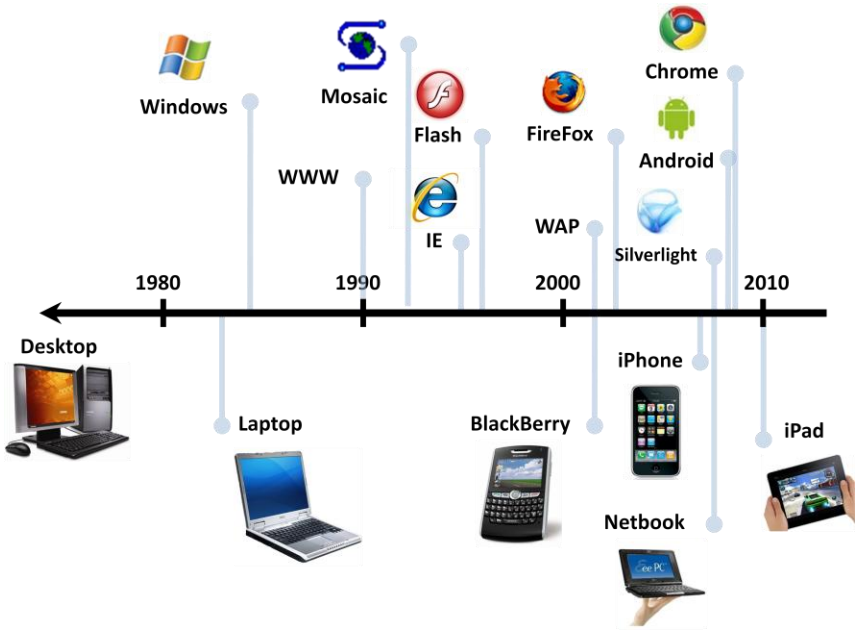
The recent advances in battery, screen and CPU technology, coupled with a growth in wireless connectivity have made it possible to deliver applications to end-users on a wide range of devices, from conventional computers like desktops, laptops and netbooks, to mobile phones, TVs and even household appliances. Applications are typically served to these devices via the desktop, web and mobile platforms. Unfortunately the languages, tools and frameworks used to develop for these platforms are quite different, making it a costly exercise to distribute applications via all three.

This white-paper describes how Windows Presentation Foundation (WPF) and Silverlight unify development across the desktop and web platforms, and with the upcoming release of Windows Phone 7, the mobile platform as well.

CONTENTS

INTRODUCTION

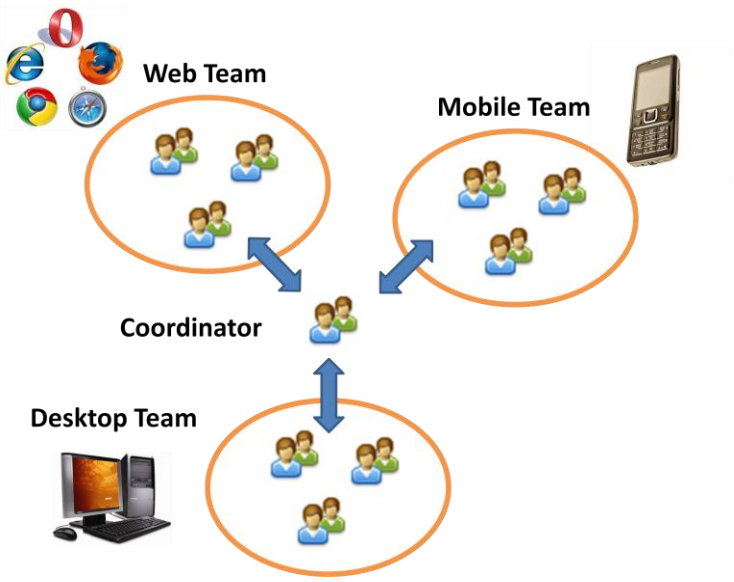
A few decades ago the internet revolutionised how data is shared between computers, allowing global access to information. More recently, the introduction of Rich Internet Applications removed the need to install applications on a particular physical machine, giving users the potential to access their applications from any computer connected to the internet. Wireless connectivity has provided roaming access so that users are no longer tied to physical locations, however, laptops and netbooks are still a little too bulky to take everywhere. With advances in CPU and battery technology mobile phones (smart phones) have now made it possible for users to take their applications with them everywhere, tucked into their shirt pocket.



A timeline of technologies and platforms for application delivery

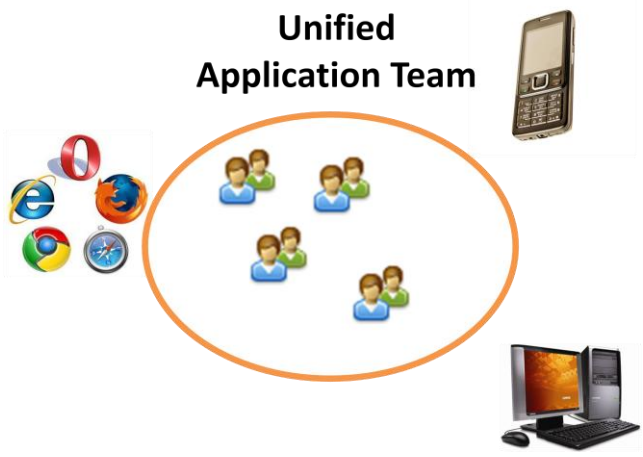
The use of desktop, web and mobile as platforms for delivery of applications gives great power and flexibility to the end user, however this trio of platforms presents a great challenge for the application developer. The differences in hardware, operating systems, programming language and development environment mean that in practice although the user experience across platforms may be similar, the code required to drive each is not. The recent emergence of web services and cloud computing allows the sharing of application logic and data across platforms, however, the physical applications running on each platform are still different.

Server-side logic aside, it is quite common for applications developed across two or more of these platforms to be completely separate entities, developed in different technologies using different tools. Furthermore, because of the skills and experiences required to work with each technology, they are often developed by different software engineers operating within distinct teams. The need for separate development teams, code duplication, and the coordination required to maintain a consistent user experience is clearly costly.



Technology differences result in separate development teams for each platform

If the same development frameworks and tools could be used to target all three platforms the process could be greatly simplified. Common logic and components could be implemented within a shared code-base, with relatively minor platform specific customisation, for example, touch-specific actions on mobile devices, being performed. Just as importantly, the software engineers could transfer their skills between each platform, taking with them important business knowledge and making cross-platform user experiences more consistent.

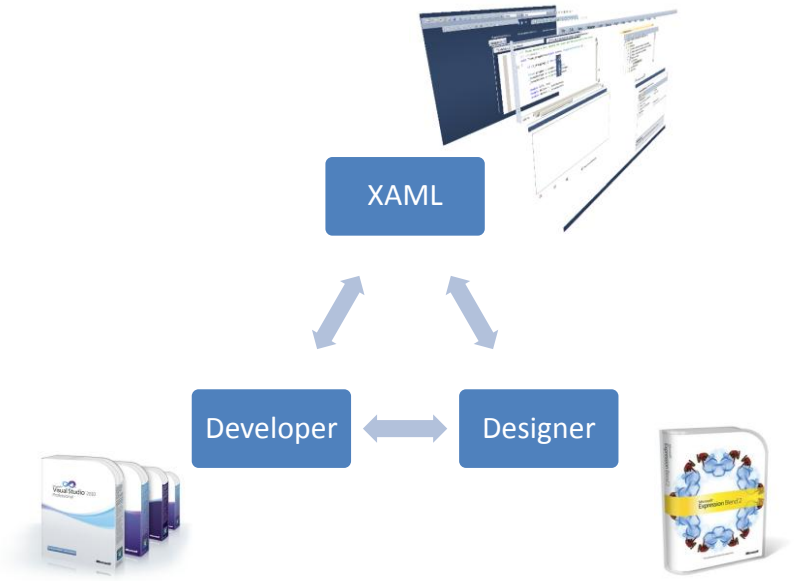


Unification of platform technology leads to a unified application development team

WPF, Silverlight and the recent launch of Windows Phone 7 make the unified application team a reality. We will look in detail at these application development frameworks and how they are related.

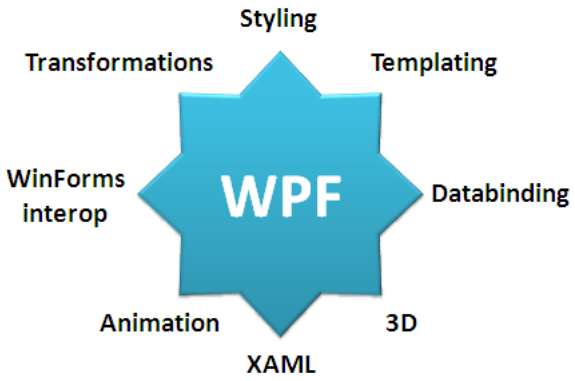
Desktop Development With WPF

Released in 2006 under the original name of Avalon, Windows Presentation Foundation (or WPF) is the latest desktop application development framework from Microsoft. It can be considered a replacement for Windows Forms, which has been a very popular framework for a number of years. WPF introduced a number of new concepts, including eXtensible Application Markup Language (or XAML), an XML based markup language for specifying the applications user interface. The separation between UI and application logic which XAML enforces enables a more streamlined developer-designer workflow. Graphic designers using Expression Blend, a tool tailored for designing XAML UIs, are able to work on the graphics and layout for the application UI, creating the required markup, whilst developers working with Visual Studio can create the interaction / business logic.



Developer-designer workflow

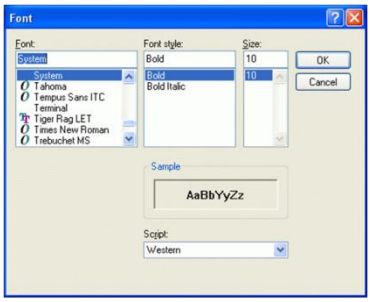
However, WPF does much more than bring graphic designers directly into the development process. It has a number of features not found in Windows Forms that allow more rapid application development and a more flexible end product.



WPF key features

Some of the headline WPF features like animation, 3D, skins / themes may not be immediately attractive to the developers of business applications. However, we will have a brief look at how the underlying architecture that makes these possible also allows the more rapid development of business applications through the WPF concept of 'templates'.

Application user interfaces are becoming much more graphical, with users expecting a more engaging and 'lively' experience. However, graphics do not have to be used to simply add visual appeal, they can be used to help users visualise the options presented to them. Using Microsoft Word as an example, the font selector in Word 2002 uses common UI controls, and a simple 'preview' area. The same selector in Word 2007 renders each selection in the font directly and also includes other graphically depicted information such as the recently used font group. This more visual approach makes it much easier for the user to locate a font suitable for their needs. This is one small example, there are many more if you compare the older and more recent versions of Word.



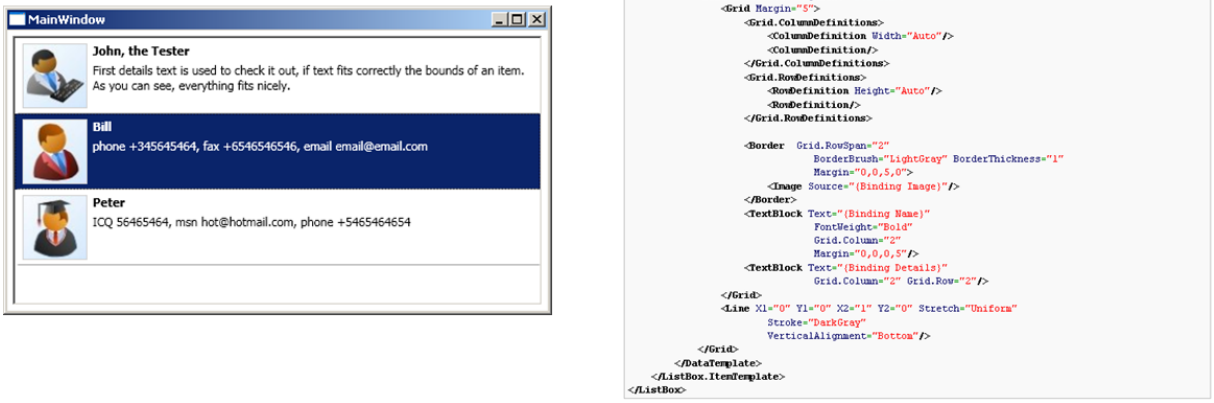
Word 2002

Word 2007

The development of graphically rich UIs presents a challenge for the developer. Windows Forms supplies a number of standard UI controls that permit a small amount of customisation. If a developer wishes to customise the UI in a way which the control designer did not originally envisage, for example adding icons into a ComboBox, creating round buttons, or nesting controls within a DataGrid, they must typically go down the 'owner draw' route, where the developer is responsible for rendering of some

parts, or all of the control. There is no scope for making small changes to the control's standard render process.

WPF introduces the concepts of templates which are re-usable XAML definitions of how a specific control or data type is rendered. This provides almost endless flexibility when customising controls or adapting them to display your data. Rather than having to completely re-draw a control from scratch the developer is able access its template, tweak it, add new elements, or change it completely. The example shown below demonstrates how easy it is to render non-standard graphics within a listbox via a small XAML template:



A simple XAML DataTemplate

In contrast, the same UI created with Windows Forms requires complex and primitive drawing operations [1]. The clarity and flexibility of the WPF templating approach mean that complex UIs such as the Word 2007 font selector above can be developed with WPF in a matter of hours, compared to the many days required to do the same with Windows Forms.

It is easy to see the appeal of WPF for authors of consumer software, where eye-catching graphics are very important in order to stand out in the marketplace, and Microsoft expected that the early adopters of WPF would be from this mould. Interestingly Microsoft's Tim Sneath noted that the strong architectural patterns present in WPF, including templating and binding, have resulted in authors of enterprise application adopting the WPF framework [2]. However, at Scott Logic we have not experienced much interest in WPF from our clients within the financial domain. There are probably a number of reasons behind this apparent slow adoption; one is the cost of training existing development teams in this new technology. However, the biggest factor is most likely that an application developed in WPF does not open up any new business opportunities, it may cost a little less to develop than a Windows Forms equivalent (training aside) and look a bit nicer, but it still delivers the same functionality via the same medium of delivery ... the desktop.

When WPF moved onto the web, under the new name of Silverlight, this all changed.

Web Development With Silverlight

Application delivery on the internet via the World Wide Web, has always been a tricky business. Web browsers still rely heavily on HTML / HTTP, technologies developed 20 years ago, that are geared towards static book-like content with interconnecting hyperlinks. The HTML fabric of the web lacks the concepts that an application developer needs such as controls, validation and rich user-interaction.

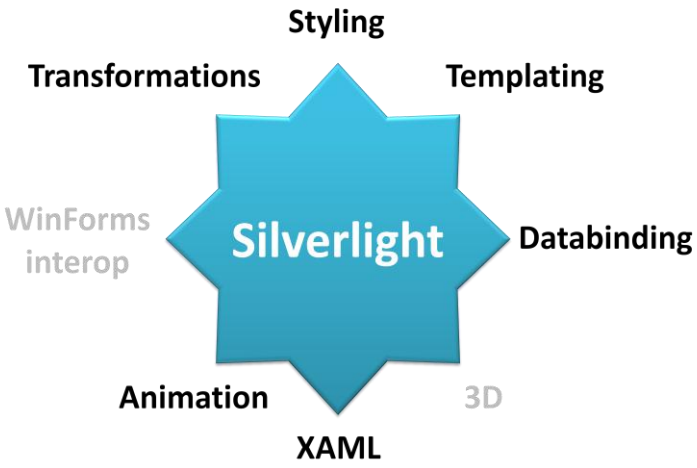
In the mid 90s a number of technologies started to emerge which filled this gap. With JavaScript, developers were able to make HTML dynamic, this technology, combined with CSS for styling, are still very popular today. However, despite the existence of numerous standards, cross-browser compatibility is still a big problem, with web projects typically including long phases of cross-browser testing and bugfixing.

An alternative route to the delivery of applications over the web is the use of browser plug-in technologies, with early examples being Sun Java and Adobe (formerly MacroMedia) Flash. These technologies have the advantage that they provide exactly the same runtime regardless of the browser used. Also, browser plugins can bring with them features that are missing in the static HTML web, such as suites of UI controls. The main issue with plugins is that they are not a 'standard' part of the web, and it is typically up to the user, or the administrator of their PC in a corporate environment, whether they install the plugin.

Originally conceived as WPF/E, or WPF Everywhere, Microsoft Silverlight was launched in 2007. Silverlight is a browser plugin that provides a slimmed-down version of WPF; the WPF runtime is 54Mbytes, whereas Silverlight is less than 5Mbytes. Both frameworks use XAML markup at their core and share a very similar feature set. Silverlight applications are developed with the same toolset as WPF, with developers writing C# or VB.NET applications within Visual Studio, and designers working on the shared XAML markup that describes the application UI via Expression Blend.

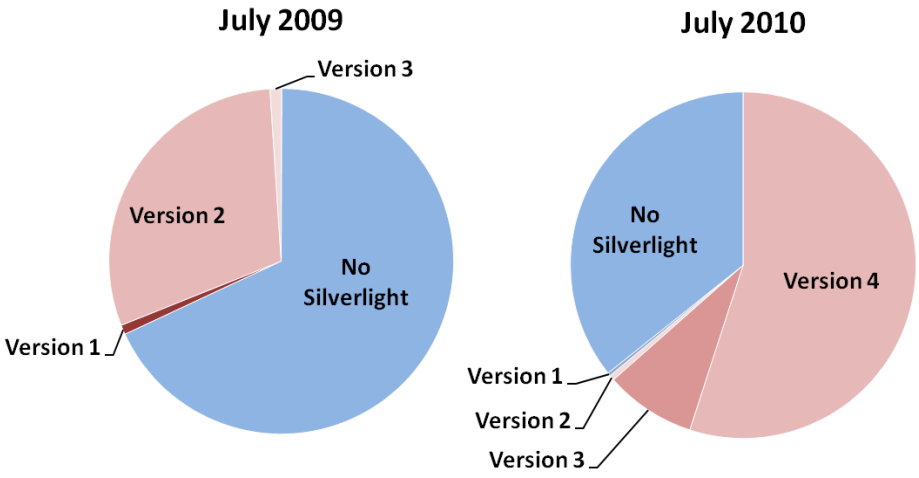
In practical terms, Silverlight does lose a few features to WPF, for example, Silverlight lacks a fully-featured 3D framework. There are also smaller API differences at the detail level. However, with knowledge of the differences [3] it is a reasonably straightforward process to author an application that targets both platforms, with the main challenges being usability issues, i.e. how to make best use of the desktop / web / mobile platforms.

From an end-user experience, Silverlight is in many ways very similar to Adobe Flash. Both are browser plugins that share a great many features, including 3D frameworks, support for HD video, hardware graphics acceleration, and more beside. They are also both able to operate out-of-browser, allowing the user to install an application from the web onto their desktop. Where they differ most is the languages and development tools used to develop for each. Flash applications are written in ActionScript (an object oriented language influenced by JavaScript) using the Flash Builder IDE, whereas Silverlight application are written in C# (or other .NET languages) using Visual Studio, hence the developer skills required to work with each are quite different.



Silverlight key features

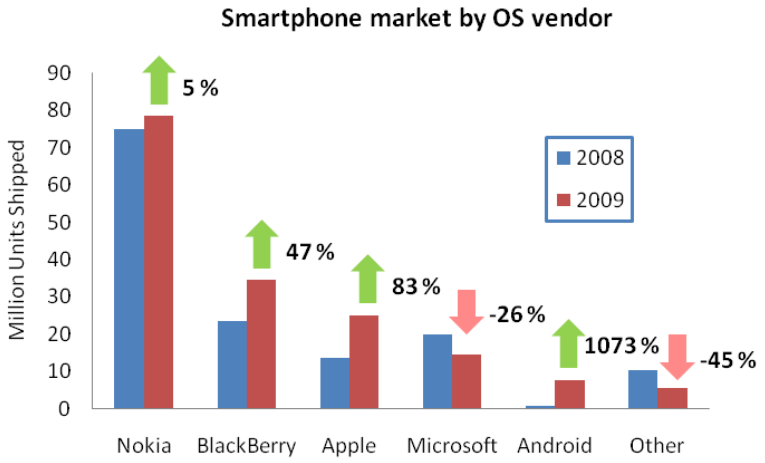
Probably the most important decision that needs to be made when developing a web application using plugin technologies is whether your target customers have the plugin installed, and if not, whether they would be willing to install it to use the application which you are offering. Adobe Flash has a distinct advantage in this area, with recent statistics indicating a 97% adoption for version 10 of their plugin, which was release in 2008 [4]. Silverlight is a relative newcomer to the plugin market; despite this, the adoption has been quite rapid. About one year ago when Silverlight 3 was launched adoption was around 30% [5], now it is around 62% [6] and growing steadily.



Silverlight Adoption Statistics

Mobile Development With WP7

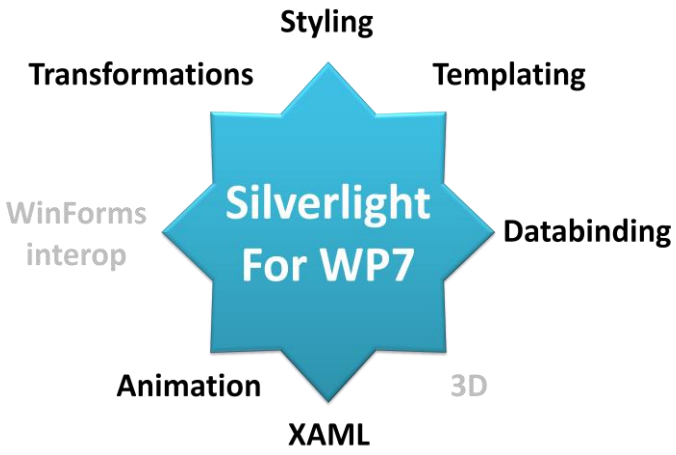
Smartphones, mobile devices capable of running complex graphical applications, are the fastest growing segment of the mobile market with BlackBerry, Apple and Android closing the gap on Nokia which based on 2009 sales are the current market leader [7].



Smartphone market trends from 2008 to 2009[7]

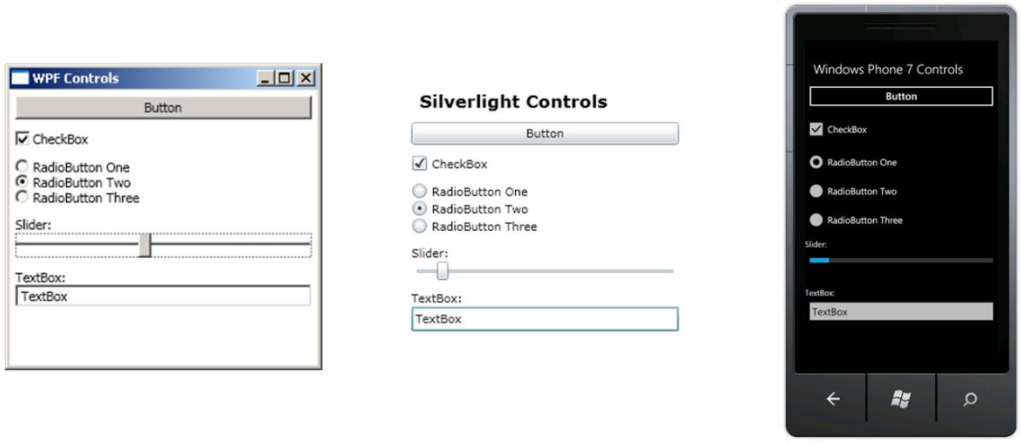
Microsoft's Windows Phone smartphone OS has lost market share and has seen its overall sales drop from 2008 to 2009. To tackle this problem Microsoft are about to launch Windows Phone 7 (WP7), which is a 'reboot' of the Windows Phone range. WP7 has a very different architecture and software stack than its predecessor, Windows Phone 6.5, and is not backwards compatible. These radical changes are Microsoft's gambit to regain a footing in the smartphone marketplace. The beta toolset for WP7 development became available in April 2010, and shipment of consumer devices is expected to occur towards the end of 2010, with devices expected from HTC, LG, Samsung, Dell and ASUS [8].

Windows Phone 7 application developers have a choice between using the XNA framework, which is geared more towards game development, and Silverlight, which is of more interest to business application developers. This means that WP7 applications are developed using the same .NET programming languages, XAML and application framework as their WPF and Silverlight for Web counterparts. It also enables the same developer-designer workflow, where designers are able to use Expression Blend to design the UI for phone applications



Silverlight for Windows Phone 7 Key Features

Whilst the various controls present in the WPF and Silverlight (on the web) differ in their cosmetics, the same controls differ more markedly in the Windows Phone environment. This is due to the very different form factor of a mobile device, where buttons and other controls must have a suitably large surface for touch interaction with the fingertips, and scrolling is performed via swipe gestures. The screenshot below shows exactly the same application code running on WPF, Silverlight and WP7 to illustrate how the UI is rendered differently on each platform:



The same application UI rendered by WPF, Silverlight and WP7

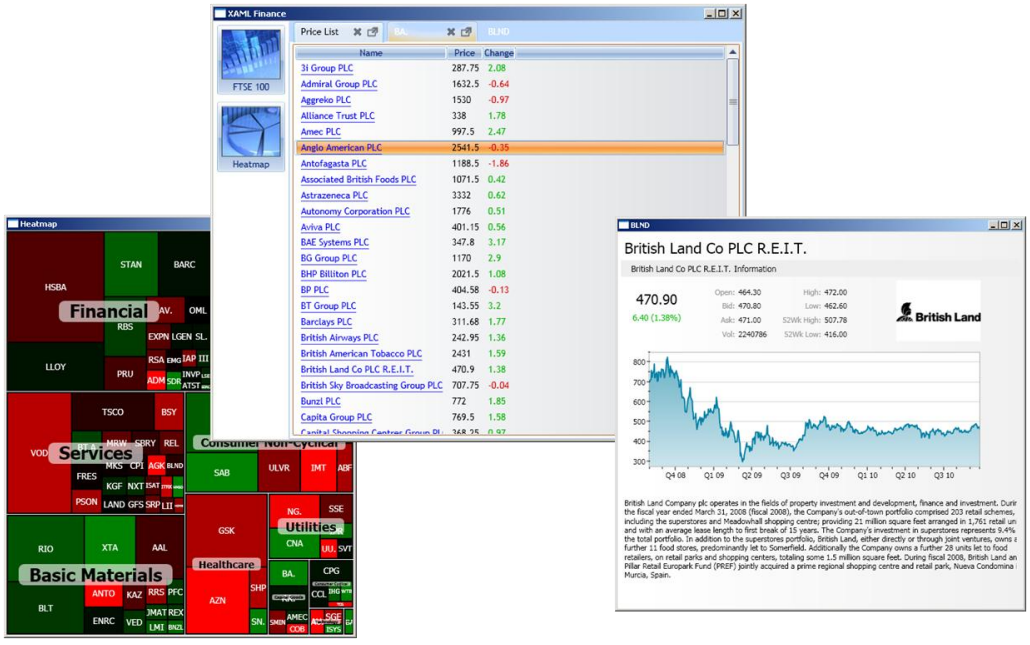
Microsoft have been keen to emulate the success of Apple's iPhone strong branding, with the Metro Design Language (a set of fonts, controls, navigation concepts etc...) bringing a standard look and feel to the phone's applications, and a comprehensive set of UI guidelines [9].

MULTI-PLATFORM XAML APPLICATIONS

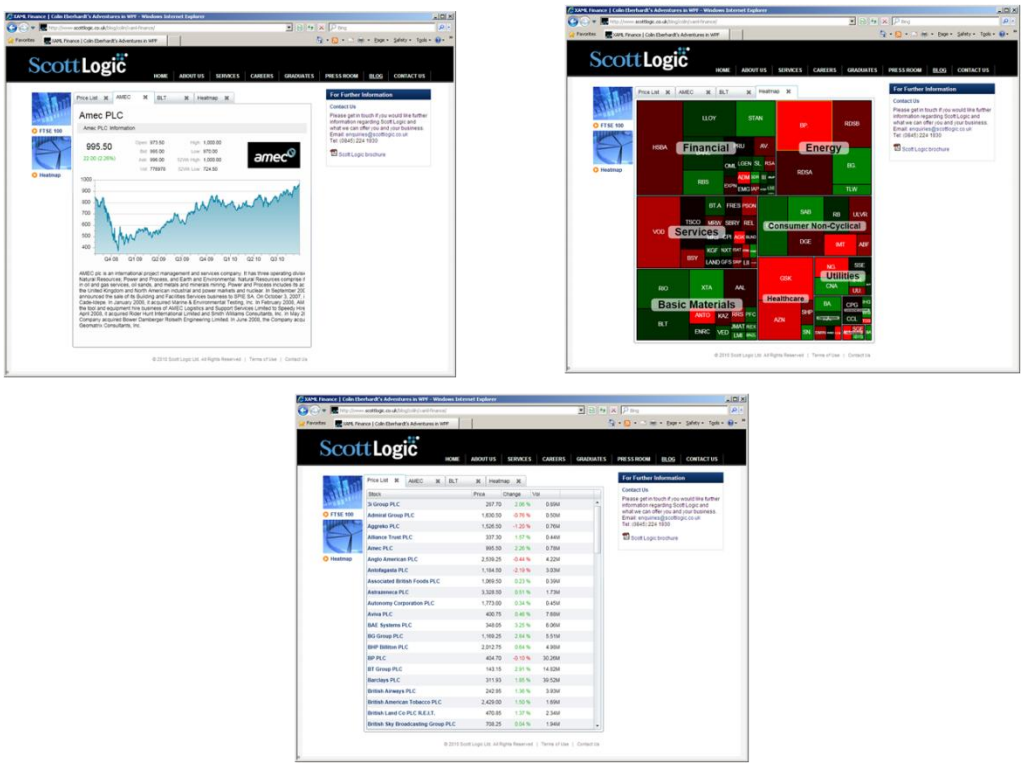
The three XAML based frameworks, WPF, Silverlight and Silverlight for WP7 make it possible to write an application that can be built and run on the desktop web and WP7 mobile devices from the same shared codebase. This enables companies to target multiple platforms without having to re-implement the same logic with different languages / frameworks within distinct development teams. The idea of having a single unified application development team is now a practical reality, with the all the team members working with the same framework and tools.

However, in order to make best use of the different platform's form factors it is anticipated that there would be some differences in the application UI across these platforms. For example, a docking layout manager provides a flexible workspace for desktop applications, but would probably be quite impractical on a mobile device where screen dimensions are much smaller.

With the use of suitable UI design patterns, such as the Model-View-ViewModel pattern [10], it is possible to selectively share UI code, whilst permitting platform specific customization. To demonstrate the feasibility of this approach, Scott Logic has developed XAML Finance, a cross-platform application for exploring FTSE stocks.



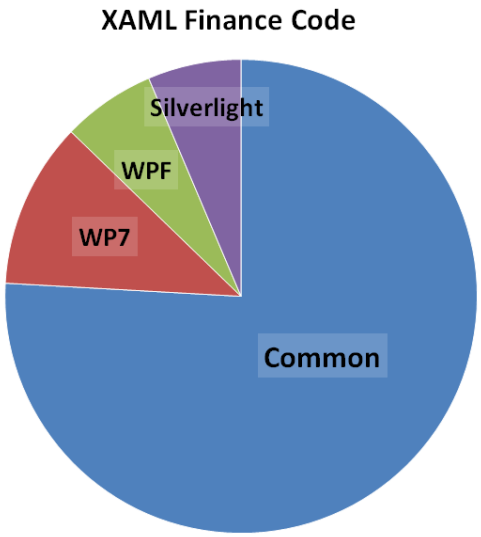
XAML Finance WPF Desktop Edition



XAML Finance Silverlight Web Edition

XAML Finance Silverlight Windows Phone 7 Edition

The XAML Finance application shares just over 75% of its C# codebase across all three versions. The code that is not shared is tailored to platform-specific tasks, such as the management of windows in the WPF version, tabbed navigation in the Silverlight version, and the WP7 navigation bar. However, it should be noted that this platform specific code is still of course implemented using the same language / framework. There is also some platform-specific XAML which is tailored to the screen size of each platform. With XAML Finance being an application which is mostly concerned with the presentation of static data, there is very little business logic in the application code. For more complex applications the amount of shared code would be increased.



The extent of code shared between the three XAML Finance applications

Care must be taken when creating a cross-platform application in order to accommodate some of the slight differences between the three platforms. Silverlight is not a strict subset of WPF, and likewise Silverlight for WP7 is not a strict subset of Silverlight for the web. These differences are most likely due to the independence that Microsoft afforded the team responsible for each, with innovation being encouraged as opposed to a more restrictive specification / standards driven approach. The flow of innovation is not always from WPF into the Silverlight subset, with new concepts such as the visual state manager being trialled in Silverlight first before making it into WPF in the .NET 4.0 release. This innovation-led approach has certainly brought Silverlight and WP7 development to the market more quickly, but is an issue for developers of cross-platform applications. Fortunately Microsoft sees Silverlight and WPF as being on a path of convergence [11], however, when this will happen and what this means in a practical sense is not yet clear.

CONCLUSIONS

In the introduction the typical team structure required for developing an application for the desktop, web and mobile was described, where three independent teams are coordinated in their development of the three distinct applications for each of the three platforms. This split into three teams is necessitated by the different languages, frameworks, tools and technologies used to develop for each platform.

This paper has shown that with WPF, Silverlight and WP7, the development framework for all three platforms becomes unified, with developers and designers able to use the same mature development tools to produce applications for each device. However, this unification goes one step further; not only can the same skills be used across each platform, but also *exactly the same code* can be used across each platform. This greatly reduces the need for coordination between the development teams working on each platform to ensure consistency, as well as reducing net development cost due to the development of duplicate functionality across platforms.

Looking towards the near future, I think that Silverlight adoption will continue to rise and will start to become the framework of choice for business applications. Businesses which choose Silverlight as their target platform will be able to draw on the existing talent pool of .NET developers. Whilst the current adoption levels of Silverlight (62%) might be deemed prohibitive for some business application developers, they should also consider that their applications deliver something of value to the end user; hence they are more likely to install a small plugin to use their application than the user of a casual consumer application.

It is envisaged that Flash will still dominate the web as a whole, but as Silverlight pushes into the business market, Flash will continue mostly as a tool for creating interactive adverts and banners. The convergence of WPF and Silverlight will continue, however, as Silverlight brings .NET applications to the web and mobile, it will be Silverlight that drives the adoption of its 'big brother' framework. Furthermore, Silverlight will start to appear on other platforms, with a Silverlight plugin for Nokia smartphones currently in beta [12], plans underway to bring Silverlight to set-top boxes and blu-ray players [13] and rumours abound that Silverlight will become available on the fast growing Android Mobile OS [14]. This all adds up to Silverlight/WPF becoming a powerful framework for cross-platform application development.

BIBLIOGRAPHY

1. **Eberhardt, Colin.** Templates, or Why I love WPF (and Silverlight Too!). [Online] Sept 2010. http://www.scottlogic.co.uk/blog/colin/2010/09/templates-or-why-i-love-wpf-and-silverlight-too/.

2. **Sneath, Tim.** Introducing the Third Major Release of Windows Presentation Foundation. [Online] May 2008. http://blogs.msdn.com/b/tims/archive/2008/05/12/introducing-the-third-major-release-of-windows-presentation-foundation.aspx.

3. **Wintellect.** Guidance on Differences between WPF and Silverlight. [Online] July 2009. http://wpfslguidance.codeplex.com/.

4. **Adobe.** Flash Player Version Penetration. [Online] June 2010. http://www.adobe.com/products/player_census/flashplayer/version_penetration.html.

5. **Eberhardt, Colin.** News on Silverlight adoption from the Silverlight 3 UK Launch. [Online] July 2009. http://www.scottlogic.co.uk/blog/colin/2009/07/news-from-the-silverlight-3-uk-launch-and-an-answer-to-my-question-on-silverlight-adoption/.

6. **riastats.** Rich Internet Application Statistics. [Online] Sept 2010. http://www.riastats.com/#.

7. **Canalys.** Majority of smart phones now have touchscreens. [Online] Feb 2010. http://www.canalys.com/pr/2010/r2010021.html.

8. **GadgetVenue.** Microsoft Announces Windows Phone 7 Hardware Manufacturers. [Online] July 2010. http://www.gadgetvenue.com/microsoft-announces-windows-phone-7-hardware-manufacturers-07233803/.

9. **Microsoft.** Windows Phone 7 Series UI Design and Interaction Guide. [Online] July 2010. http://download.microsoft.com/download/D/8/6/D869941E-455D-4882-A6B8-0DBCAA6AF2D4/UI%20Design%20and%20Interaction%20Guide%20for%20Windows%20Phone%207%20Series.pdf

10. **Smith, Josh.** WPF Apps With The Model-View-ViewModel Design Pattern. [Online] Feb 2009. http://msdn.microsoft.com/en-us/magazine/dd419663.aspx.

11. **Brown, Pete.** The Future of Client App Dev : WPF and Silverlight Convergence. [Online] Dec 2009. http://10rem.net/blog/2009/12/01/the-future-of-client-app-dev--wpf-and-silverlight-convergence.

12. **Nokia Beta Labs.** Microsoft Silverlight for Symbian. [Online] March 2010. http://betalabs.nokia.com/apps/silverlight.

13. **Microsoft.** Microsoft Silverlight Recap at NAB 2010. [Online] April 2010. http://team.silverlight.net/announcement/microsoft-silverlight-recap-at-nab-2010/.

14. **ZDNet.** Could Silverlight be Microsoft's next app for Android? [Online] March 2010. http://www.zdnet.com/blog/microsoft/could-silverlight-be-microsofts-next-app-for-android/5477.