XAMARIN BUILDING COST EFFECTIVE CROSS-PLATFORM MOBILE SOLUTIONS

epam

CONTENTS

INTRODUCTION

Cross-platform mobile development refers to a technique of writing common codebase applications which target different mobile operating systems. Major mobile operating systems, such as Android and iOS, are structurally different in their architecture. It is no wonder, therefore, that this causes massive trouble for developers trying to make apps work well and look nice on different platforms. For example, Google and Apple encourage feature implementations to be written in contradicting ways, making it very difficult for programmers to write a common codebase which will work on both operating systems. It also means that developers will have to duplicate feature implementations and continue tweaking them for each platform.

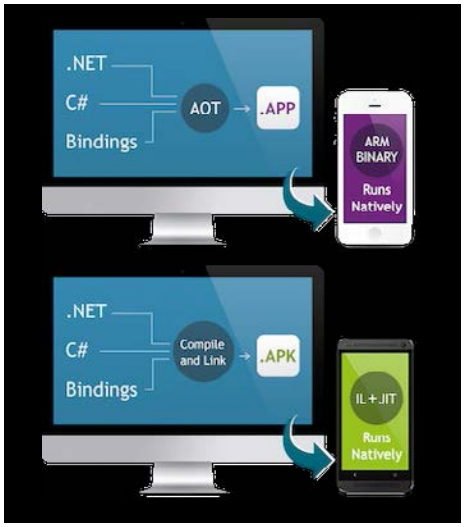
For these and many more reasons, the demand for cross-platform mobile development is constantly growing in the market. Maintainability and extensibility are becoming particularly more essential for larger multi-platform projects.

Using Xamarin as a baseline framework and toolset will improve the quality of resulting solutions and team efficiency. Utilizing .NET/C# capabilities, unified architecture and best of the best development IDE, while keeping all platform SDK and UI features directly accessible, makes solutions cost effective.

ABOUT XAMARIN

Xamarin is a multifaceted toolset for building native iOS and Android applications using C# as a cross-platform language and .NET as the base framework for both platforms. Xamarin opens up .NET Framework and C# capabilities for mobile development, including third party libraries. Most importantly, it creates a complete .NET wrapper over native iOS SDK and Android SDK, giving full access to all platform features and even the ability to interoperate with third party platform binaries (iOS, Android) using the same approach. Besides coding framework, Xamarin provides a choice of IDE and development tools, including the open-source Xamarin Studio, integration with Microsoft Visual Studio and allowing multi-platform mobile development on Windows and Mac machines.

Xamarin is a quality product that is progressing actively, getting more and more traction by providing unique capabilities to the mobile development technologies market.



- More than 600,000 developers are active in the Xamarin Community
- Three major releases within three years
  - v.1 – A stable platform and toolset that allows shared code in C# for iOS and Android
  - v.2 (February 2013) – Visual Studio integration, newest C# syntax supported, component store
  - v.3 (June 2014) – A new iOS visual designer, unified UI framework: Xamarin.Forms, Nuget support in Xamarin Studio
- Officially supported by major mobile platforms and services
- Tight partnership with Microsoft since 2013

> "Founded in May 2011, Xamarin has a history that dates back more than a decade. The principals had key roles on the Mono project, a free-of-charge and open-source initiative launched in 2001 that implemented the C# language, .NET framework and runtime environment on multiple platforms.
>
> Xamarin uses multifaceted technology built by a cohesive team with a solid long-term track record."

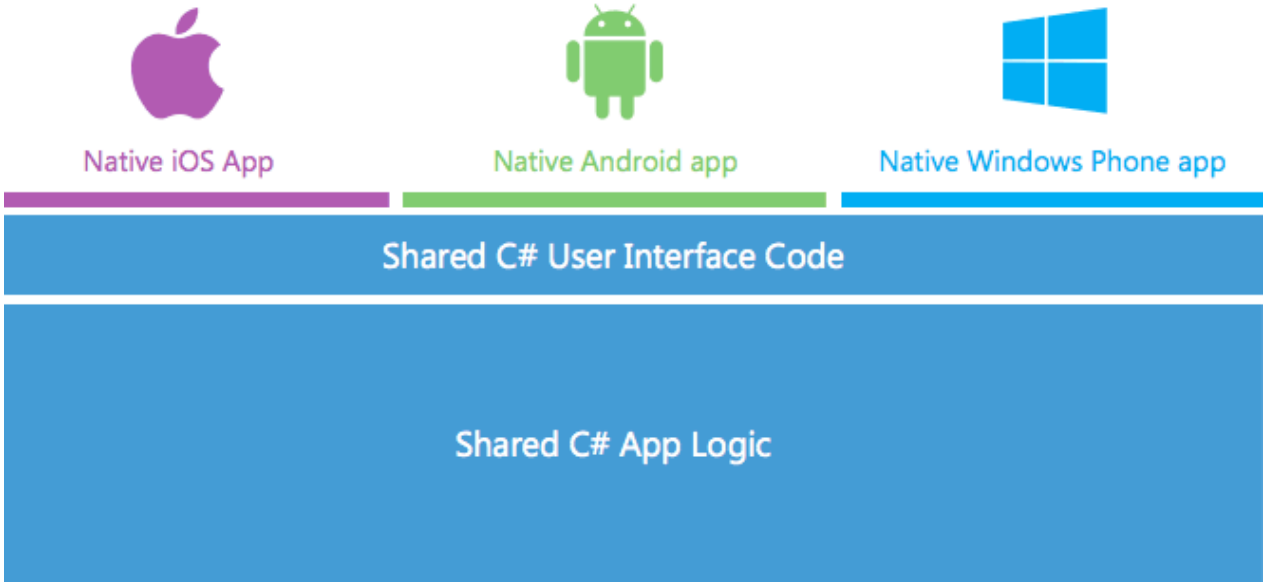
© Gartner, August 2013

XAMARIN VALUE PROPOSITION

A XAMARIN-BASED SOLUTION HAS SEVERAL SIGNIFICANT ADVANTAGES

- A multi-platform mobile product is implemented as a **unified solution using a single programming language**, which results in a more coherent team and approach to a product as one solution targeting multiple platforms from an architectural and coding perspective, instead of having two or three separate teams and projects independently building platform apps.
- The developed product is a "native" application, as if it were developed using platform tools: **Xamarin provides complete non-restricted access to native iOS SDK, Android SDK** and the ability to interoperate with third party platform components. It gives the same high performance of native apps, and in some cases is even more efficient by using .NET libraries (collections, string operations). At the same time, it allows reusability of existing .NET based components, which reduces the solution's cost.

- A big part of the architecture becomes **shared code: a reusable library** responsible for business logic, communication and other shared functionality. It is developed once and then reused in all target apps, which saves 15-25 percent of development time on average.
- The biggest practical value of using Xamarin is in its **more simplified maintenance and support** of mobile apps, because it is done once, using one programming language and with one approach. It automatically affects all platforms, without the chance for human error to creep into only one platform, making the problem difficult to locate.
- **The effectiveness of IDE exceeds platform tools by a generation – Visual Studio and ReSharper.** C# language is supported to the full extent of v.5 and is more efficient and versatile than Java and Objective-C. Developers can utilize all the best features of the language: anonymous methods and classes, real generics, lambdas, LINQ, yield, and even async/await. Using Visual Studio as an IDE and the ReSharper add-in bring the coding experience to its highest efficiency and ensures code quality guidelines are met.
- Throughout its history, Xamarin has kept its **policy of "available the same day**," meaning that whenever a new version of iOS or Android is released, a new version of Xamarin is generated by a special automatic tool and released the same day.
- It has become possible to apply the **architectural pattern MVVM to an entire solution**, making it testable on all levels without requiring a native test runner thanks to the clear separation of concerns that the patterns provide.



- In the recent 3.x versions of Xamarin, completely new unified user interface technology is introduced. **Xamarin. Forms allows one time UI development against an abstract library of components**, and framework will build the corresponding native UI that is appropriate for each platform (iOS, Android and Windows Phone). For several specific cases, that could bring up to 100 percent code reuse between across platforms, which simplifies maintenance of the UI as well.
- Additional value comes from **easy-to-integrate Component Store and Nuget** packages that make using third party components a very easy and reliable process.

XAMARIN CONSIDERATIONS

Xamarin is not a silver bullet though. There are some considerations that developers need to address before selecting Xamarin as a development platform of mobile apps. Here are some of them:

- Xamarin adds additional costs to development: $999 per platform, per developer, per year (business level). A typical **Xamarin-based team will require** $6,000-$8,000 per year just in licensing costs. Having a MSDN subscription provides a 30% discount, but it still amounts to **$4,200-$5,600 per year**.
- **There is a risk that some approaches will not work, and time will be lost finding a workaround.** There are a number of (minor) things that will not work in Xamarin by design, and we have encountered a few issues from time to time in the platform. We always find workarounds, but it still wastes time. In general, this is a rather minor issue for large projects.
- A complicated development environment is required for maximum efficiency: **Mac and Windows machines for each developer** (or virtualization technology to run Windows on Mac).
- The highest level of team efficiency requires a unique skillset from the team members: solid .NET skills combined with knowledge of iOS and/or Android SDK and generic mobile development approaches. It is **difficult to find specialists with the ideal skillset**. We recommend including part-time Mobile Solution Architects in every cross platform development team to manage the skill gaps.

COMPARISON MATRIX

XAMARIN, PLATFORM SDK, MOBILE APPLICATION DEVELOPMENT PLATFORMS			
	PLATFORM SDK	**XAMARIN**	**MOBILE APPLICATION DEVELOPMENT PLATFORMS**
DESCRIPTION	Development using an out-of-the-box SDK platform, provided by a platform vendor (Android SDK, iOS SDK, Windows Phone SDK) to build native apps. Various mobile services usually provide a native connector.	Xamarin defines a single development framework and tools on top platform SDKs (iOS, Android) to build native apps. Xamarin opens up a full .NET components world to mobile developers and is able to cover most of the mobile services in .NET.	Each platform (Kony, Appcelerator, IBM Worklight, etc) defines its own specific approach to building and managing apps. They usually use web technologies to develop multi-platform apps. Such platforms often come with a wide range of out-of-the-box well-integrated services.
KEY BENEFITS	The original platform approach. Allows platform features to be developed in the way they were initially designed and intended to be developed.	Xamarin is a functional wrapper over Platform SDKs and allows full access to SDK features, while enabling the apps to be built using a single programming language (C#) and utilizing the full power of the .NET world in a cross platform way.	These platforms focus on rapid development and add a spectrum of built-in tools and services to support enterprise and consumer focused apps. They usually utilize web technologies to enable cross platform benefits. UI is either generalized to support framework abstractions, or mimics the native approach via web technologies.
KEY DRAWBACKS	Each platform (iOS, Android, Windows) has a completely unique approach to mobile app development. Building for multiple platforms results in multiple independent projects with almost no room for sharing code and architecture.	Xamarin has significant developer license costs. Developing for Xamarin requires strong .NET skill mixed with knowledge of iOS and Android platforms which is a rare combination.	Very high costs for larger projects with high usage. Lower performance and visual glitches in web-based mobile UI. Very limited UI ability to customize UI for Kony and Appcelerator UI frameworks. Requires specific platform skill (Kony, Appcelerator).

PRODUCT / APP PERFORMANCE	**Native / Great (5/5)** Full control over the application UI and performance	**Native / Great (5/5)** Same level of control as the Platform SDK apps	**Appcelerator: (4 / 5)** Native / Good **Kony: (4 / 5)** Native / Good
			PhoneGap: (2 / 5) Hybrid / Below avg (due to use of low-performant frameworks)
			IBM Worklight: (3 / 5) Hybrid / OK **Telerik Platform: (3 / 5)** Hybrid / OK
COMPONENTS AVAILABLE FOR INTEGRATION	**Minimal + Manual 3rd party (3 / 5)** Only SDK is available out of the box. Third party open source components and libraries	**Great (5 / 5)** Full SDK support .NET BCL and third party libraries Nuget and Xamarin Component Store for easy integration.	**OK-Good (3-4 / 5)** Depending on the platform, developers can access a variety of features out of the box, which usually cover most scenarios. There are components stores and libraries. The platforms are closed and component reuse is usually restricted within the platform only.
CUSTOMIZABLE UI	**(5 / 5)** Complete low level access to UI SDK with a full range of capabilities. Native UI can be customized in a way that is allowed by the system, there are third party native controls and all platform controls can be extended.	**(5 / 5)** Relatively the same access to the platform and native third party components, though it requires some integration work. On the plus side, the new Xamarin 3.0 adds multi–platform controls and a third party library of such UI elements is being developed to extend the initial set.	**Appcelerator: (2 / 5)** Great for basic quickly built User Interfaces, but very difficult to add platform features. **Kony: (2 / 5)** Very similar to Appcelerator, which has the same problem.
			IBM Worklight: (4 / 5) HTML-based UI with a good collection of components. **Telerik Platform: (4 / 5)** Kendo UI is a powerful HTML based framework.
			PhoneGap: (3 / 5) Many approaches available, but a rather fragmented approach using HTML/JS.

MULTI-PLATFORM: DEVELOPMENT (DIFFICULTY, SPEED, QUALITY OF RESULT)	**(2 / 5)** iOS and Android development tools are far from being perfect, but good enough for development. The platforms are rather complex and require a deep understanding of the capabilities.	**(5 / 5)** Visual Studio and a single programming language make the life of developers much easier. Though understanding the platform specifics is required to implement device features, most of the code will be part of a shared core library and will use the power of C# / .NET to connect to services, manage data and objects, do calculations, etc.	**(4 / 5)** All these platforms are focused on rapid development and usually provide WYSIWYG interface. Some platforms require learning of their framework on top of the native SDK, while others rely on HTML and JS. In either case, the speed of development is very good, but the tools are not perfect and customizing will increase solution difficulty.
MULTI-PLATFORM: ARCHITECTURE	**(1 / 5)** Each platform defines architectural best practices. Following them brings decent results. But they differ between platforms, eliminating reuse. Feature implementation will vary between platforms, which creates some unexpected differences between apps.	**(5 / 5)** MvvM is a single proven architectural approach for all targets and specifically tailored to UI and Logic separation. It enforces a very productive approach to development, and it is testable. It removes the chance for the apps to accumulate many minor differences throughout the development stage.	Usually a platform will define architecture for the app. It is designed to be flexible for cross platform development with a specific tool. **Appcelerator: (4 / 5)** **Kony: (4 / 5)** **IBM Worklight: (4 / 5)** **Telerik Platform: (4 / 5)** **PhoneGap: (3 / 5)**
MULTI-PLATFORM: SHARED CODE	**(1 / 5)** Sharing will happen within the same platform (iOS Phone and Tablet for example).	**(4 / 5)** 40-60% of the code will be shared in the core library. Xamarin Forms also allow increasing that ratio by adding cross-platform UI. The shared .NET library can even be easily reused in other mobile apps and non-mobile projects due to the nature of .NET.	**Appcelerator: (5 / 5)** Can have very high code reuse **Kony: (5 / 5)** Can have very high code reuse **IBM Worklight: (5 / 5)** Can have very high code reuse **Telerik Platform: (5 / 5)** Can have very high code reuse **PhoneGap: (4 / 5)** Can have high code reuse, but plugins require native coding
MULTI-PLATFORM: MAINTENANCE	**(1 / 5)** The actual updates and fixes are not too difficult from a maintenance perspective, but the fact that the development team needs to apply them in 2-4 places each time makes the support of the multi-platform apps rather difficult.	**(5 / 5)** Most of the issues that require code changes during maintenance are the changes in the shared code, which allows the changes to be applied only once and work on all targets.	**Appcelerator: (5 / 5)** **Kony: (5 / 5)** **IBM Worklight: (5 / 5)** **Telerik Platform: (5 / 5)** **PhoneGap: (4 / 5)** Decent, but it depends on the HTML framework chosen. Plugin maintenance is just as difficult as native development.

TEAM STAFFING	**(3 / 5)** The required skill for each team member is adequately defined, but each platform requires a separate team of skilled engineers.	**(4 / 5)** The team's main skill should be .NET/C# and the team works together on a single solution that results in multiple apps. Though the team size could be ~15-20% less, the skill set of the developers should include platform SDKs, which is a very rare combination.	**Appcelerator, Kony: (3 / 5)** The team will require a skill in a specific framework that is not usable anywhere else and fairly rare. **IBM Worklight, Telerik Platform, PhoneGap: (5 / 5)** HTML/JS based framework projects require a very common set of skills that can easily be staffed.
OVERALL ADDITIONAL COSTS	**(5 / 5)** Native development requires no specific costs besides the development program fee (~$100/platform).	**(4 / 5)** License costs are applied and are generally high: it usually costs ~$1,398/developer/year + MSDN Subscription. But no upkeep costs are incurred other than the hosting option, which is up to the client to choose.	**Appcelerator, Kony, IBM Worklight: (2 / 5)** For larger apps, the costs will grow very fast and may reach millions of dollars per year. But these costs are for the various backend services that the platform provides. However, similar services are available from cloud vendors at a much lower rate. **Telerik Platform, PhoneGap: (4 / 5)** These platforms usually require a developer license, like Xamarin.

SUMMARY

Platform selection nowadays requires making several decisions. Major companies still prefer to choose native development over HTML-based or Framework-based cross platform development, because that is the only feasible way to get the most out of the application in terms of UX and performance.

Xamarin allows developers to combine the best of both worlds: the benefits of a single codebase, cross-platform components, reuse, a unified team and reduced development costs, while still enabling all the options that are allowed by native development.

Established in 1993, EPAM Systems, Inc. (NYSE: EPAM) is recognized as a leader in software product development by independent research agencies. Headquartered in the United States, EPAM serves clients worldwide utilizing its award-winning global delivery platform and its locations in 19 countries across North America, Europe, Asia and Australia. EPAM was ranked #6 in 2013 America's 25 Fastest-Growing Tech Companies and #3 in 2014 America's Best Small Companies lists by Forbes Magazine.

FOR MORE INFORMATION, PLEASE VISIT EPAM.COM

GLOBAL

41 University Drive, Suite 202
Newtown, PA 18940, USA

P: +1 267 759 9000
F: +1 267 759 8989

EUROPE

112/114 Middlesex Street, 3r Floor
London, E1 7HY, United Kingdom

P: +44 20 7377 2864

CIS

1/1 Academician Kuprevich Street
Suite 110, 220141 Minsk, Belarus

P: +375 17 389 0100
F: +375 17 268 6699