Xamarin or Native?
An “angel vs devil on your shoulder” dilemma

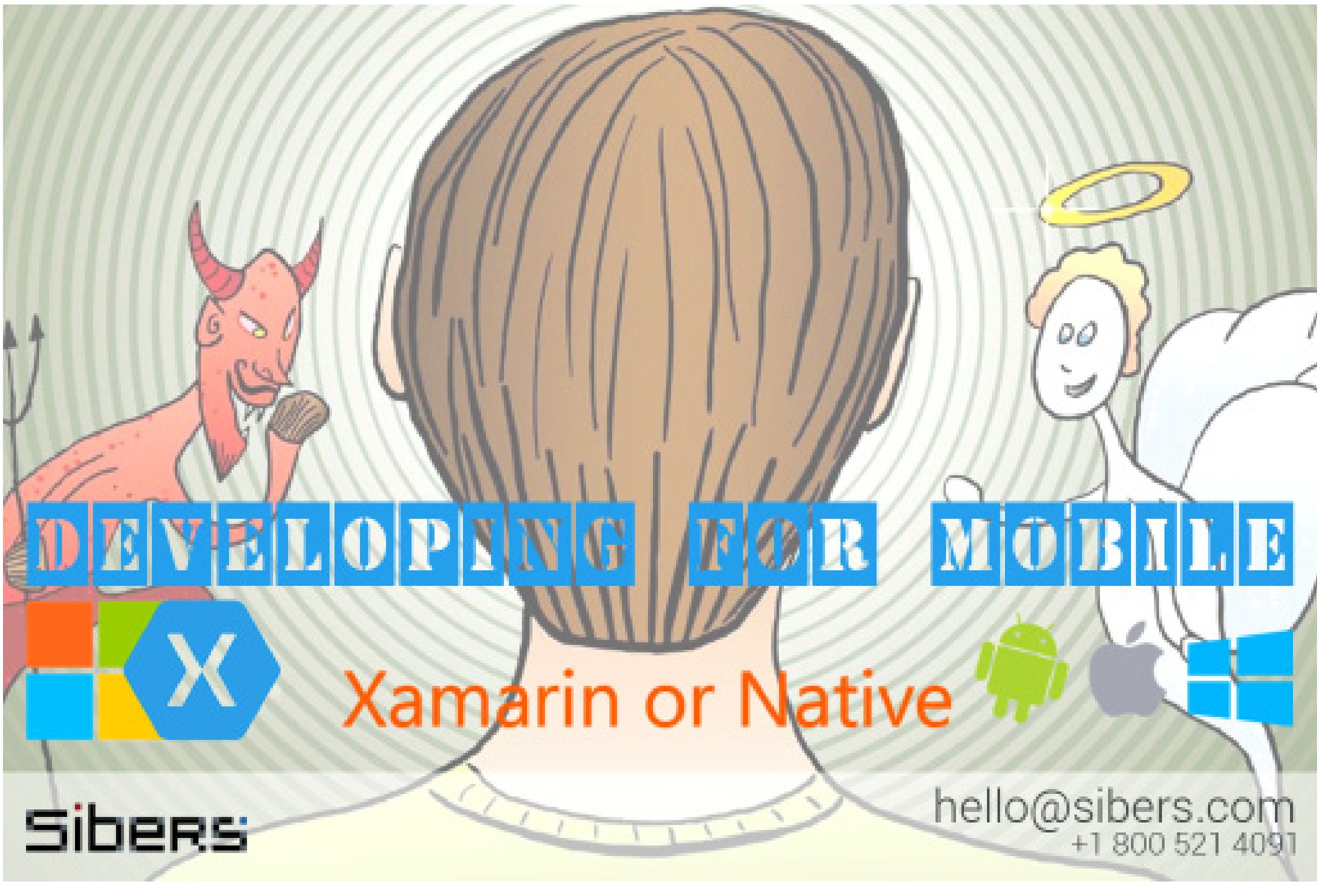
Article by Sibers at AppFutura Blog, June 2016

Xamarin or Native?
An “angel vs devil on your shoulder” dilemma

June 2016, AppFutura Blog
https://www.appfutura.com/blog/xamarin-or-native-an-angel-vs-devil-on-your-shoulder-dilemma/

“Multipurpose Xamarin”, whispers a tiny red-clad figure on your shoulder, and the advice is hard to ignore — especially since, as of March 2016, Xamarin is strongly supported by the Microsoft name and the entire community around it. This of course carries significant weight.

“Native only!” whispers another tiny figure dressed in white on the opposite shoulder, who for illustrative purposes is best imagined as an iOS or Android developer.



Two effective solutions, but very different from one another. *Sibers presents us with this dilemma.* Which is the angel and which is the devil? Let’s consider their arguments and get to the bottom of this “moral dilemma”.

Before Xamarin appeared in 2011, there were only two mobile development paths, each occupying an opposite pole in terms of cost vs. quality balance: native iOS/Android/Windows Phone development, and cross-platform development using PhoneGap framework, which allowed for developing mobile solutions in JavaScript and HTML/CSS and reusing 100% of code across platforms. The latter path, which was meant to avoid paying double when developing for two platforms, resulted in a dramatic drop in performance. Word spread fast, and the word was, “Don’t do cross-platform. And don’t even think about using PhoneGap.”

But with Xamarin’s arrival and maturation, such foreboding advice no longer applies to cross-platform.

What is Xamarin?

It’s a platform founded on the basis of Mono, an open-source realization of .NET Framework. In plain language, Xamarin is a handy tool for translating code written in C# into native codes for popular mobile platforms.

On February 24, 2016, Microsoft acquired Xamarin, and soon after confirmed its inclusion in Visual Studio as a no extra-charge, open source program.

What does this mean?

It means that any software provider with a Microsoft technologies development team now has a Xamarin license as a part of its Visual Studio license, at no extra cost. To better understand the gravity of this occasion, consider that in 2014, a Xamarin license cost $999 per year for one developer per one platform. And three platforms cost a tidy $2,997.

Using Xamarin for development of several different mobile platforms simultaneously means that a customer works with one team, or even one developer (coding in C#), on all parts of the system: server-side, and numerous clients (websites, Windows and OS X desktop, and mobile applications for Android, iOS, and Windows Mobile). And most importantly, it means code-sharing across apps, since about 70% of the code running across Android, iOS, and Windows can be shared, thus saving time and money.

Sounds great, right? Especially for enterprise solutions: business logic is coded in C#, bugs are fixed when found, libraries are developed and then updated, and the system works consistently well on every platform. Also not to be overlooked is the fact that going open source means the whole .NET community should conceivably have an opportunity to improve Xamarin, fueling its rapid evolution and refining its stability.

To date, Sibers’ Microsoft development team has created more than a dozen solutions using Xamarin, and we firmly believe that C# has a bright future in mobile development. However, there is also a soft spot: a user interface (UI) cannot be reused across platforms and therefore must be created separately for every platform.

Let's imagine you're going to develop a mobile application of average complexity and you ask your development partner to provide you with a ballpark development estimate for two native apps for iOS and Android, in Objective-C and Java respectively, developed using Xamarin. Most likely you'll hear that in both cases the quality and performance will be identical, and the development time (and thus the cost) will differ only minimally. For instance, each native app would require, for each platform, 50 hours on logic and 150 hours on the UI, which equals 400 total development hours spent on applications for two platforms.

Now, if we use Xamarin, we will develop one code for the app's logic for both platforms (50 hours total), and two UIs at 150 hours each. This adds up to 350 hours, and we'll add another 25 hours to cover cross-platform compliance (insurance, fixes, etc.). The grand total is 375 hours, pretty much the same as the native scenario. And since Xamarin is free, the cost is pretty much the same too. So right now this is a minor soft spot, nothing more, and made up for by the fact that you only need to develop one code for the app's logic.

In sum, choosing between native and Xamarin development is a matter of a particular project's functionality and performance requirements. If the software development provider has mobile and Microsoft development teams, the customer should provide a detailed description of requirements in order to receive reasonable advice that results in the most inexpensive solution for developing a stable and user-friendly application. If you need an enterprise solution with server-side, website, and mobile apps, or you plan to port your app to all existing platforms, Xamarin is much preferred, and you will probably come out ahead. However, if you desire a complicated app that's targeted at iOS adepts, native would probably be better since you'll be certain that each and every interface and logic element strictly follows Apple's guidelines.

So which one is the devil and which one is the angel? The joke is there's no definitive right answer. In this scenario, Sibers thinks there is no absolute good or evil. Xamarin and native mobile development each have their own niche and range of use. All you need to do is rely on your experience and that of your trusted developer, and choose wisely.

You can get more information about Sibers visiting their AppFutura company profile.