That Mobile Programming Mess

Adrian Kosmaczewski

2009-11-25

Let's be honest. 3 months ago I said that it wasn't a good time to be an iPhone developer[1]. Today, it looks like being one sucks every day a bit more. This article will dive into the alternatives, horrors, breakpoints, misconceptions, IDEs, sadness, hope, USB cables, and all those different factors that are shaping this thing called mobile software development market.

1) iPhone

First we had Rogue Amoeba[2] and Joe Hewitt[3] leaving the App Store, and then we have the autobot in the review process[4] rejecting apps including Joe's Three20 library[5] (which is a highly ironic fact, if you think about it), and then rejecting apps with method or property names apparently similar[6] to those of private APIs, then app rankings going bezerk[7] for a little while, then the release dates having hiccups[8], and so on, and so on.

All of this sucks. Ask Paul Graham[9] if you don't believe me. And check the improvements suggested by Enormego[10]. There is a consensus about the fact that things aren't working – at least from the developers' point of view.

However, let's be even more honest, things don't look better on the other side of the fence.

[1] /blog/risk-management-in-iphone-projects/

[2] http://www.rogueamoeba.com/utm/2009/11/13/airfoil-speakers-touch-1-0-1-finally-ships/

[3] http://www.techcrunch.com/2009/11/11/joe-hewitt-developer-of-facebooks-massively-popular-iphone-app-quits-the-project/

[4] http://gizmodo.com/5405978/iphone-apps-have-to-be-approved-by-robots-now-too

[5] http://groups.google.com/group/three20/browse_thread/thread/c442af6e39a918b0/6d5046771539d139

[6] http://twitter.com/zenoc/status/5856233410

[7] http://www.tuaw.com/2009/11/04/developers-report-a-moment-of-upside-down-app-rankings-now-retu/

[8] http://www.theiphoneblog.com/2009/11/24/itunes-app-store-release-date-sorting-sorta-broken/?utm_source=feedburner&utm_medium=feed&utm_campaign=Feed%3A+TheIphoneBlog+%28The+iPhone+Blog%29&utm_content=Google+Reader

[9] http://paulgraham.com/apple.html

[10] http://developers.enormego.com/view/appstore_improvements?utm_source=feedburner&utm_medium=feed&utm_campaign=Feed%3A+EnormegoDevelopers+%28Enormego+Developer+Blog%29&utm_content=Google+Reader

2) Web Apps

According to Peter-Paul Koch[11] (of Quirksmode[12] fame, all in all a highly respected web developer), the problem of the App Store is not Apple, but the stupid developers who insist in writing native apps when they could just write web apps using good ol'HTML 5 and CSS.

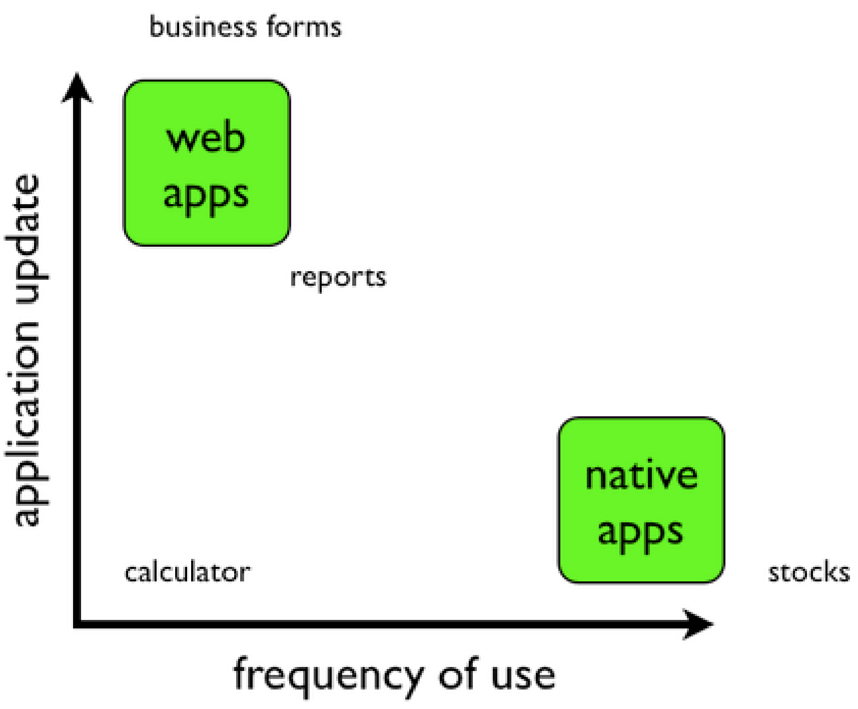
I agree that for a small range of applications, this approach would work, going from simple to-do lists, and even some social networking applications[13]. I've enumerated the option of web apps in a recent article of mine[14] about alternatives to the native SDK.

But, there's always a "but":

- The money is in native iPhone applications. My clients, at least, they want them, they are ready to pay for them, just like they paid for websites back in 1997. And the monetization of iPhone apps is simply brilliant. Submit your app, cross your fingers, charge a few bucks, and you have potentially 50 million clients ready to buy it. Try finding a monetization model for your web app based entirely on the iPhone.
- Games: although I'm not a game developer (yet) at least for the moment it's plain unthinkable to create fun, appealing games using HTML and JavaScript.
- Hardware access: you can access the GPS information, but that's more or less everything you can do. Try to access the camera, the local network stack (think Bonjour-based apps), or the microphone, or the speakers, or the iPod music library, or the address book of the user, or the accelerometer, or complex UI animations or transitions, and you're toast. The point is, many complex apps are based on those elements. Complex user experiences involve some or all of these elements, not just plain data. I do hope that Apple will allow access to some of these native features in the iPhone soon, particularly the accelerometer and the camera, which would be in my opinion perfect candidates for a JavaScript API.

There certainly is a business case for web apps, as I explained in my presentation last year at Geneva's iPhone Conference 2008:[15]

[11] http://www.quirksmode.org/blog/archives/2009/11/apple_is_not_ev.html
[12] http://www.quirksmode.org/
[13] http://googlemobile.blogspot.com/2009/07/google-latitude-now-for-iphone.html
[14] /blog/iphone-apps-without-objective-c/
[15] /blog/iphone-conference-2008/



In short: yes, web apps are a perfect alternative in some cases; yes, Apple is unpredictable and the App Store must be improved; and finally no, developers aren't stupid. There's a great deal of personal taste in the choice of a development platform, and in this case, the tradeoff is sometimes worth the pain.

John Gruber says it even better than me[16]:

> But the best proof is what I pointed out above: Apple itself created almost no iPhone web apps. Successful iPhone developers don't just want to write software that works on the iPhone. They want to write software for the iPhone that's just as good as Apple's. Today that means using Cocoa Touch and the native SDK.
>
> When you write a Cocoa Touch app for the iPhone, you're not starting from scratch. You're starting with the Cocoa Touch framework. As Faruk Ateş astutely points out in his response to Koch, to discount the framework is to discount everything that sets the iPhone apart as a development platform. Not only are native iPhone apps faster and more capable than their web-app equivalents, but they're easier to write.

Anyway.

[16]http://daringfireball.net/2009/11/iphone_web_apps_alternative

3) Android

I think, without any doubt, that Android is one of the most important platforms of the past few years (I agree with Tim Bray[17] in this point). Android has many characteristics that are, at first sight, extremely appealing:

- It's open source, based on open standards and proven technologies;
- It's backed by a company known by its technical expertise and innovation;
- It's growing faster and faster every day;
- It is proving popular among users, too – and this is not a minor element!

But, of course, not everything is green: Android basically trades off a shitty review process with a shitty programming environment (at least until IntelliJ IDEA 9[18] ships*). Not only that: the whole platform is threatened by Google releasing its own device[19], or by the fragmentation of the platform[20] by the device vendors, or by the low numbers of app sales[21] (which is somewhat surprising, given the apparent strong sales of some devices like the Droid[22]).

Of all the drawbacks I enumerated above, I want to focus in one aspect: the problem of platform fragmentation is, in my opinion, so big, so important, that voices are raising to explain[23] that this single factor might blow the whole platform:

> Can you write an .apk application that runs on all devices? Theoretically, yes. But not without testing on an ever-increasing number of gadgets. This is the problem that Symbian and J2ME phones have, and the road that Android is headed down if Google doesn't reign in control and quickly. Differing OS versions, different manufacturer and carrier customizations, and various app stores are going to hobble the OS before too long.
>
> All that said, I still believe it's got a real future after using it regularly on my Archos, but Google needs to get control quickly. I had originally suggested using the Android logo and trademark (which they may or may not own) as a way of ensuring compatibility, but it seems the logo is creative commons. So maybe they need to come up with an "Android Approved" logo or something.

At least in terms of programmer experience, though, Android still has a long way to go. To begin with, it's based in Java; and I will never get tired of quoting this paragraph from His Highness Steve Jegge[24] (Google employee, by the way):

> I'll give you the capsule synopsis, the one-sentence summary of the learnings I had from the Bad Thing that happened to me while writing my game in Java: if you begin with the assumption that

[17]http://www.tbray.org/ongoing/When/200x/2009/11/20/Android-Splintering

[18]http://www.jetbrains.com/idea/nextversion/#Android_Development

[19]http://www.techcrunch.com/2009/11/17/thegoogle-phone/

[20]http://www.theiphoneblog.com/2009/11/17/fake-steve-android-fragmentation-harder-develop-iphone/

[21]http://us.mobile.reuters.com/m/FullArticle/p.rdt/CTECH/ntechnologyNews_uUSTRE5AJ1EU20091120

[22]http://www.pcmag.com/article2/0,2817,2355916,00.asp

[23]http://www.russellbeattie.com/blog/android-is-splintering-just-not-how-you-think-it-is

[24]http://steve-yegge.blogspot.com/2007/12/codes-worst-enemy.html

> you need to shrink your code base, you will eventually be forced to conclude that you cannot continue to use Java. Conversely, if you begin with the assumption that you must use Java, then you will eventually be forced to conclude that you will have millions of lines of code.

You can't be more clear. More code leads to higher maintenance costs, higher chances of bugs, and in general, harder-to-maintain code bases. The choice of Java[25] in the long run will hurt Android more than anything else. In a similar way, the current App Store policies are doing a similar harm, which is, as Graham said, to make developers run away from either platform.

Additionally, I might also point out the fact that currently the only way to write Android apps, is using Java (with a small subset of functionality available through the Android Native Development Toolkit[26]); compare this with the amount of alternative frameworks for the iPhone[27], and the discussion of the "openness" of the Android platform takes a different spin.

Finally, there's this factor Gruber discusses[28], this syndrom called the "year of Android", similar to the "year of Linux" promised since the early years of this decade… and which every year turns out, inexorably, in favor of other platforms.

> I'll just say that if the consensus winds up that the Droid isn't a great Android phone, this is the sort of attitude that'll sink Android. It's the same attitude desktop Linux has always had, that the future is going to be great, so don't worry about the present.
>
> Like a sports team that's always saying "Wait until next year", meanwhile, Apple has won another championship this year.

4) The Others

Just to name them, a bit further away in the finish line, we have a Windows Mobile[29] base of millions of devices without any serious roadmap, and which looks more like a neglected child than anything else. Or the Palm Pre which is, hum, nonexistent (at least in this side of the world). Or BlackBerry, which is, hum, off the radar (but which I think is the real third alternative to the iPhone and Android). Or Symbian and J2ME, which are, well, Symbian and J2ME.

So, in short, this means that programming for mobile platforms is both shitty and complex, no matter how you look at it.

Great.

Meanwhile, in a small town in Switzerland…

akosma software does and will concentrate mobile development efforts on the iPhone and Android platforms. We think, given all the facts we could chew so

[25]http://twitter.com/stevedekorte/status/5902464622
[26]http://developer.android.com/sdk/ndk/1.6_r1/index.html
[27]/blog/iphone-apps-without-objective-c/
[28]http://daringfireball.net/linked/2009/11/22/bray-android
[29]http://reddevnews.com/blogs/desmond-file/2009/01/microsofts-mobile-mess.aspx

far, that these two are more likely to dominate the mobile software development business in the years to come than all the others combined.

Nevertheless, in terms of absolute and relative numbers, current projects and cash at stake, iPhone is the clear winner at the time of this writing (something said a couple of months ago in an interview on the Swiss newspaper Le Temps[30]).

In terms of potential for growth, we cannot and we do not underestimate Android (hence our interest). Particularly, there are some factors that could boost Android's appeal, which are huge opportunities for growth, somewhat neglected so far:

- Google should promote the platform more, including ads on TV, and also allowing paid apps in more countries;
- Developers should be capable of creating Android apps in as many languages as possible. Compiled or not, scripting or not, object-oriented or not, it doesn't matter – I vote for C++ and Ruby;
- We need a better IDE, now (and a faster emulator, too. The current one sucks).

Above all, there's one important fact: programming is still fun, creative and interesting. And we'll keep on creating new stuff (and ranting about it), no matter what platform we choose.

(*) I have tried the current beta, and I don't think[31] it will change the game that much. No visual GUI editor (something that even Eclipse has, albeit in a primitive way), and the configuration of a single project, well, it's IntelliJ IDEA[32]. I couldn't even run a simple project with it. Let's see when the final release ships.

Update, 2009-11-26: This article in CNN[33] (French version)[34] talks about the problem platform fragmentation represents for small shops. It brings a strange feeling: the "open" Android platform might be only open to those with enough means to have a whole testing framework with the most popular devices, leaving out small shops who wouldn't be able to support their applications in all devices, given the famous fragmentation.

[30] /blog/interview-sur-le-temps/
[31] http://twitter.com/akosma/status/5949948423
[32] /blog/not-exactly-what-i-meant/
[33] http://edition.cnn.com/2009/TECH/11/17/android.wired/index.html
[34] http://www.pcinpact.com/actu/news/54322-android-croissance-inquietudes-developpeurs-smartphones.htm