Desirable Requirements of Cross Platform Mobile Development Tools

Lamia Gaouar, Abdelkrim Benamar, Fethi Tarik Bendimerad
University Abou Bekr Belakid Tlemcen
Algeria
lamia.gaouar@gmail.com, abk_benamar@yahoo.com, ftbendimerad@gmail.com

ABSTRACT: *The ubiquity of mobile devices is the result of increasing popularity of mobile applications that are becoming more diverse and intuitive. The mobile device market is composed of several mobile platforms such as Android and iOS, so it is important to be able to deliver applications available for more than one platform in less time. But it is becoming increasingly difficult with the incompatibility between platforms and their SDK. To solve that, cross platform mobile development solutions have been investigated. Although each of such tools allows developing cross-platform mobile applications, the result can sometimes be unsatisfactory in comparison to a native application. In this paper, we present existing cross platform approaches. We discuss the general architecture of cross platform mobile applications. Then, we propose the desirable requirements in cross platform frameworks. Some existing solutions will be evaluated based on these requirements. We conclude the paper by some future perspectives.*

Keywords: Cross-platform Tools, Mobile Application, Cross-platform Development, Requirements

Received: 1 October 2015, Revised 29 October 2015, Accepted 4 November 2015

1. Introduction

Our lives have become more mobile and the near ubiquity of mobile devices (e.g. smart phones and tablets) and the internet have contributed to this effect. Due to the features they offer (GPS, accelerometer, Music, Camera, etc.), mobile devices have becoming more a necessary than need. These kinds of functionalities are provided by all the major mobile Operating Systems such as Android, iOS and Windows Phone. In addition, the many and varied applications available make smart devices the engine for innovation. The increasing demand for mobile applications requires to developers and companies several points to consider. One influencing point is the fragmentation of the landscape of mobile platforms. Android and iOS are the leaders with a large part of the market, while other platforms as Blackberry and Windows Phone are less present (Gartner, 2014). Thus, it is of prime importance for applications vendors to provide their solutions as many platforms as possible in order to attract more users. Implementing applications for multiple mobile platforms requires considerable efforts in term of, development time, resource, tools, maintenance and deployment. Indeed, programming language and development environment of each OSs differ from one to another (Charland and LeRoux, 2011). Then, application developed for specific OS cannot be deployed on all other OS, which forces developers to rebuild the same application for other OS using, for each one, its own SDK.

To address platform and device coverage problems, cross-platform mobile development tools allow producing mobile applications which can be deployed on multiple platforms. They reduce time and cost development by allowing developers to write

applications once, using adaptable cross platform framework. These tools are based on different cross platform approaches that have emerged in recent years. They differ in term of implementation technology on which they are based. When some of such tools use web technologies as HTML, CSS and JavaScript (e.g. Sencha Touch), others use some abstraction layers as JavaScript API to access device specific features (e.g. PhoneGap). More and more cross-platform olutions appearing on the market and it is no wonder they are gaining popularity knowing the customer expectations are becoming increasingly challenging. Cross platform solutions can be recommended in general, but they differ in term of performance, usability and development environment provided.

After introduction, we present in this paper the five approaches of cross platform mobile development in section 2. Then, section 3 gives the general architecture of a cross-platform mobile application. After what, we describe some related works in section 4. Section 5 provides the definition of the desirable requirements of cross platform mobile development tools. Based on these requirements, we analyse five selected frameworks (one per approach) in section 6. The paper concludes in section 7 with some future perspectives.

2. Existing Approaches of Cross Platform Mobile Development

There are two ways to develop mobile applications for mobile devices: the native and the cross-platform approach (Charland and LeRoux, 2011. The native approach which permits to create native applications consists of developing the same application as many as there's platform using, for each platform, its own Software Development Kit (SDK) and frameworks. For example, applications for Android are programmed in Java and XML, the platform functionality is accessed using the framework provided by Android and the user interface is rendered using platform provided elements. In distinction, applications for iOS are developed using the language programming Objective-C and Apple's frameworks.

In contrast to native approach, we present in this section an overview of the cross-platform approaches. Cross platform approach proposes to create single application which can be used across multiple platforms. They are designed to save time and costs by allowing developers to write an application once in a language they know and using a framework which is adaptable for multiple platforms. For that purpose, different ways can be borrowed (Heitkötter and Hanschke, 2013) (Friese, 2014):

The web approach consists to produce web applications, using web technologies as HTML, CSS and JavaScript, designed to be executed in the web browser of mobile devices,. Applications are accessed using an URL and the mobile device will not have any application specific components installed. Hence, web applications cannot access the mobile device features as calendar or GPS sensors and can suffer for a lake of performance due to connection and network delays. Also, web applications using this approach usually do not look and feel like a native application. Several mature tools are dedicated to mobile web applications development, we can find: jQuery Mobile (2014), jQTouch (2014), Dojo Mobile (2014), AppsBuilder (2014), iBuildApp (2014) and Sencha Touch (2014) which is the tool that we have selected for our study.

To address the lack of native features but to still allow using common web technologies, the hybrid approach has emerged as a combination of web technologies and native features. Hybrid approach uses the browser engine of the mobile device which renders and displays the HTML content in native container on the device that is a full screen Web view control. Yet the application may lack native look and feel. Hybrid applications are installed on the device and its capabilities are accessed through an abstraction layer such as JavaScript APIs. However, the execution through the browser engine can affect application performances. The most popular exponent of this approach is PhoneGap (2014b) that we have chosen for our study. Among the existing tools based on this approach we can also cite frameworks as MoSync (2014), mgwt (2014) or vaadin TouchKit (2014).

The Interpreted approach follows a different way. Application code is deployed in the mobile device and at runtime the interpreter interprets the source code across different platforms. The main advantage of this approach is that the user interface is made using platform-specific native elements. The device hardware and platform features are wrapped with specific framework API, providing access to the interpreted application. The main disadvantages of this approach are the runtime interpretation that could degrade the performance of the application, and development that generally depends on the set of features provided by the selected framework. Appcelerator Titanium (2014) is among tools based on this approach. We will describe it in the appropriate section. Other tools that help to create interpreted applications are Adobe Flash Builder (2014) and Rhodes (2014).

In the case of cross-compiled approach, the cross-compiler compiles the source code into particular native code generating the native binaries. A platform-specific version of the application is created for each target platform and runs with native performances. Device features are accessible and all the native user interface components can be used. The main inconvenient of this approach is the complexity to identifying and rectifying the cross-compilation phase issues. The cross platform solutions currently available in the market are not mature enough. A very prominent example for a cross-compiler tool is Xamarin (2014b) which is the framework that we chose for that approach. We can also add QTMobile (2014) as crosscompiler tool. The tools based on this approach are dependent on the efficiency and the reliability of their cross compiler.

The last but not least, model driven approach bases the development on models to describe the applications. The models are expressed using DSLs (Domain Specific Languages) [15] and UML (Unified Modeling Language) [32]. Automatic transformations generate source code for the supported target platforms from the defined models. The application generated is truly native and respect the native look & feel. In addition, it runs on native environment without intermediate layout.

Even if most of model driven tools are still in early stage and they are not yet very popular, we have still selected the AutoMobile Project (2014b) as representative of model driven cross platform development solutions for our study. Most existing model-driven solutions like canappi mdsl (2014) or applause (2014) have not been much development progress lately, or not relevant in general practice like mobl (2014), or also not active anymore like iPhonical (2014). However, model-driven approaches appears as the most promising approach for cross-platform mobile development and gives rise to various research projects such as Xmob (Le Goaer and Waltham, 2013), MD² (Heitkötter and Majchrzak, 2013) or AXIOM (Jia and Jones, 2012).

3. General Architecture of Cross-platform Mobile Application

In this section, we aim to propose a general architecture of cross platform mobile application, presented in the next figure.

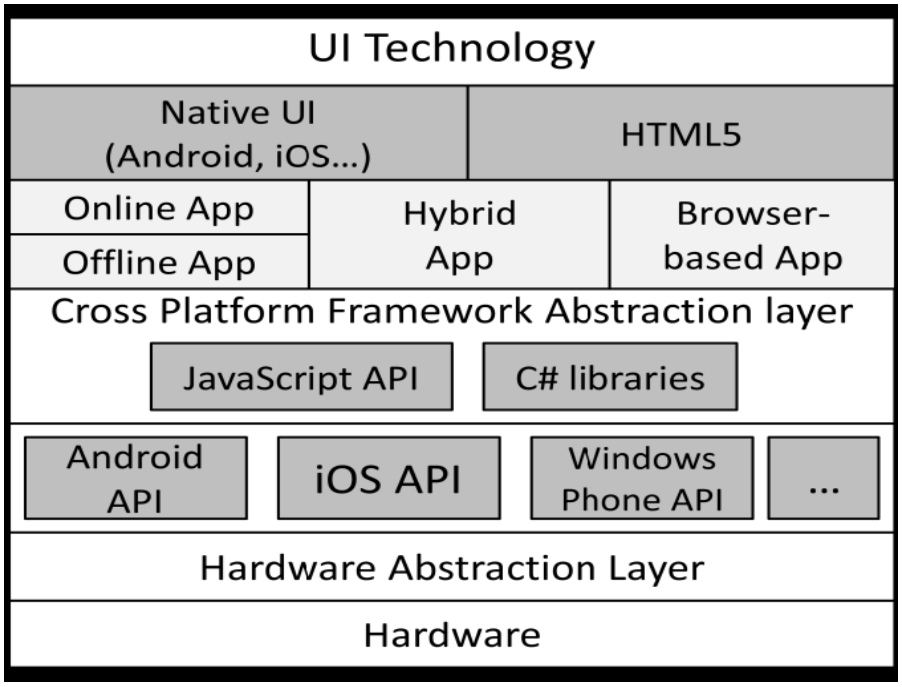

Figure 1. General architecture of cross-platform mobile application

Mobile application is kind of client-server applications where the user interface is hosted on a mobile device. In term of GUI components, we distinguish between applications that use native elements (specific for each platform), applications based on HTML5 technology who intended to run in the browser, and hybrid solutions thereof. Native applications can run online and offline too, in contrast to web HTML5 applications.

The cross platform framework allows access to device features as Camera, Contact, Sensors, etc., using JavaScript API or C#

libraries (depends on the framework used) which interact with the native API of a mobile platform. Then, the application is built separately, for each target platform, to generate the corresponding executables.

4. Related Work

Researches addressing the area of mobile development have evolved to head towards cross platform mobile development. Until recently, papers only discussed mobile platforms as we can see with the works presented by (Cho and Jeon, 2007) and (Lin and Ye, 2009). For existing work dealing with cross-platform mobile development, we can refer to (Anvaari and Jansen, 2010) that have compared mobile platforms with regard to the openness of their architectures. *Charland and LeRoux (2011)* compare the development of native applications and Web applications. Even if these works address more than one platform, they introduce the cross platform perspective only marginally.

While most articles dealing with cross-platform mobile development provide comparison criteria of crossplatform mobile development approaches or evaluation criteria for implementation technology of these approaches (Sommer and Krusche, 2013) (Allen et al., 2010), in this paper, we focus on how to define the desirable requirements for cross platform mobile development tools. Indeed, with the emergence of more than 100 tools for this field, we think it is time to ask the question about the requirements that such tools are required to provide.

We can find some related works that can be identified whenever we look for the desirable requirements of the cross platform tools. Even if (Dalmasso et al., 2013) discuss the desirable requirements in a cross platform framework, their aim is different. In fact, they have developed Android application with four cross platform tools (PhoneGap only, PhoneGap + JQuery Mobile, PhoneGap + Sencha Touch 2.0, Titanium) to evaluate the performance of such tools. (Sommer and Krusche, 2013) present an evaluation of three frameworks (Titanium, Rhodes, PhoneGap + Sencha Touch) versus two native SDKs (Android SDK, iOS SDK) in criteria pertaining to functionality, usability features, performance and other categories. *Heitkötter and Hanschke (2013)* elaborate on a list of 14 criteria for evaluating cross-platform development approaches. These criteria have been structured into infrastructure and development perspective. "*The* infrastructure perspective sums up criteria relating to the life-cycle of an app, its usage, operation and functionality/functional range. The development perspective covers all criteria that are directly related to *the development process of the app, e.g. topics like testing, debugging and development tools*". These criteria can be seen as requirements of cross platform tools but this is not explicitly mentioned because it is not the purpose of this article. In addition, all approaches are not addressed in the evaluation (Just PhoneGap and Titanium are evaluated). Despite the topic seems to be similar to our work, our aim is different. In our work, we classify cross platform development approaches into five distinct categories. We include, in our analysis, one tool of each approach for an overall assessment. The evaluation of such tools will be based on the requirements that we define in the next section.

As we can see, previous research in this area is sparse and often focuses on comparing a set of existing frameworks. In this paper, we would like to provide the desirable requirements of cross platform tools. The aim is to identify the required elements in the process of developing a cross platform mobile application, and thus, such required elements that should provide a tool to claim the title of cross platform tool. Thus, our work can be considered as a complementary work.

5. The Desirable Requirements of Cross Platform Tools

The main "objective" of cross platform mobile development approaches is to provide a mobile application that can execute on multiple platforms. However, we must not lose sight that the main "motivation" is to provide a cross-platform application which is closest to the concept of native application.

The question of supporting multiple platforms is not new. The same problem occurred, 20 years ago, for PC platforms (Windows, Unix, Mac OS, etc.). With the omnipresence of mobile devices in our daily life and fragmentation of platforms, it makes sense that developers are turning to multi-platform development and thus that framework vendors provide appropriate solutions (Allen et al., 2010).

Based on literature and our own analysis, we have identified the desirable requirements of any crossplatform technology presented as follow:

5.1 Mobile Platform Supported

Cross platform approaches by definition must support several platforms. This requirement takes into account the number and importance of supported mobile platforms, e.g. iOS, Android and Windows Phone are practically mandatory since they are the largest shared mobile platforms. In addition, the equality of the support between the platforms should also be considered.

5.2 Development Environment

Development environment covers various parameters. In this requirement, we consider the features of the development environment offered with the framework such as an Integrated Development Environment (IDE,) debugger, compiler, emulator, etc. and functionalities as source code editor, intelligent autocompletion, which usually accompanies the IDE. In addition to source code editor, the opportunity to create the graphical user interface through a WYSIWYG editor and to test the application without having to deploy it to a device or an emulator constantly is greatly appreciated. Also, the maturity of the development environment reflects the maturity of the framework.

5.3 API and Documentation

In this requirement, we discuss the documentation quality of the framework and APIs available. The influence of the documentation on the quality and ease of learning is reflected in the progress of a developer during his training of a framework. The use of APIs determines the feature of the application developed. APIs are specific to devices/platform and their availability varies from one framework to another.

5.4 Security

Smartphone are considered as easy to trap objects whose flaws are already known and exploited (tracks conversations, data recovery, scams). Applications developed with cross platform tools are not highly secure (Dalmasso et al., 2013). When it is considered that each mobile OS and each mobile device has its own flaws, it is difficult to apply a security policy to an application designed to run on multiple platforms/mobile device. The ideal would be to introduce the concept of security in the heart of the development process of the application using for example the security by design concept (Denim Group, 2010) (Cavoukian and Chanliau, 2013) (Lin and Ye, 2009). Proper research needs to be carried out to secure the tools and applications.

5.5 Access to Device Specific Features

The kind of application determines its capacity to access the features of the mobile device. There is a difference between the features according to native application and web site application. The functionality requirements of an application can be identified as follow: 1) informational requirements, where the user primarily consumes content 2) transactional requirements, where the user primarily interacts with the application to accomplish a task and 3) Device-specific requirements, e.g. offline transaction entry or file access. Most frameworks support standard device features, e.g. camera, GPS, Accelerometer, etc. and provide access to such features through intermediate layer as APIs.

5.6 Resource Consumption

Resource consumption relates to the application developed with the cross platform framework. This requirement include in order: memory usage, CPU usage and power consumption (Dalmasso et al., 2013). Mobile phones like other pervasive devices suffer from resource shortages. These resources can influence each other, for example CPU utilization affects battery consumption. Several studies have been undertaken on this subject (Yuan and Nahrstedt, 2003) (Rawassizadeh, 2009) (Murmuria et al., 2012). The memory usage may increase for various reasons, this can be due to the addition of features to generate user interface or to the use of HTML and JavaScript files. Several research works (Datta, 2012) (Bala and Garg, 2013) (Datta, Bonnet and Nikaein, 2012) (Lee et al., 2012) have appeared recently dealing with the power consumption of mobile applications. In mobile devices, the power is the most important resource (Kravets and Krishnan, 1998) and the applications developed using cross-platform tools must use the battery of mobile devices effectively.

5.7 Look and Feel

Success on an application depends in large part on user experience. Currently, the cross platform tools try to reproduce as closely as possible the look and feel of the native application. Most users seek applications that resemble to native applications, in term of graphic user interface and reactivity of the application. Indeed, mobile is a device when applications have to be interrupted by event as a call or SMS. When users want to react to this event, after that, he wants returning to the application where he left it. For that purpose, the support for backend communication protocols and data formats becomes mandatory.

5.8 Visibility

The way to distribute the application will determine the visibility of this application. Generally, users are turning to app stores of mobile platforms to obtain an application, when web site applications are accessible only through URL and internet connection. In addition, this requirement determines also the way of how to update and maintain the application, i.e. any application update in the mobile device is required for web application since data and application are hosted on the server.

6. Evaluation of Cross Platform Tools

In this section, we have selected five cross platform tools, one per approach, in order to provide a description of such tools, according to the desirable requirements that we have detailed in the previous section. The main criteria of selecting these tools were their popularity and extensive use, especially for the first four tools. For the last one, even if model driven tools are not mature enough, Automobile Project was selected because it appears as the most promising tool in its category.

We do not aim to compare these selected tools. But, our objective is to provide, for each approach, an overview of the existing cross platform tools based on what we have presented as the necessary requirements that must be met by any cross platform tool.

6.1 Example of Web Approaches: Sencha Touch

Web applications are most often obtained through frameworks such as *Sencha Touch (2014)* that allows creating free web applications. Sencha Touch is an HTML5 mobile application framework for building web applications. Sencha Touch 2.3.1, which is the latest version at the time of this writing supports Android browser, Google Chrome for Android, BlackBerry 10, Bada Mobile Browser, Kindle Fire Browser, Windows Phone 8 and Mobile Safari. In fact, this version of Sencha Touch only targeted at webkit browsers.

Sencha Touch can be used with Apache Cordova/PhoneGap or Sencha's native packager. Either enables to package the application in a native container and enables access to select device-level APIs unavailable to traditional web applications. However, the application suffers from the lack of performances due to the execution on the browser.

Sencha Touch does not provide IDE for developing Touch applications but we can look to IDE as NetBeans, Webstorm and Aptana. But none of them provides a great experience programming. In addition, Touch provides a plugin for Eclipse environment.

6.2 Example of Hybrid Approaches: PhoneGap

This framework is based on the open source Cordova project. PhoneGap supports as for now (version 3.3.0) 7 mobile platforms: Android, iOS, BlackBerry, webOS, Windows Phone 7 and 8, Symbian and Bada. It allows developers to create mobile applications using modern web technologies as HTML5, CSS3 and JavaScript (PhoneGap, 2014a).

PhoneGap is a "web code wrapper", i.e., PhoneGap operates by packaging the web page with the specific PhoneGap engine which is specific for each supported platform. This engine displays the page in a regular web view. Additionally, PhoneGap provides an access to device functionality as Accelerometer, Camera, Contact, GPS, etc. through JavaScript API's. The API is implemented differently for each platform. This is why the application result is hybrid, i.e., it is not purely native neither purely web (Allen et al., 2010). The product result is a binary application archive that can be distributed through the application market.

PhoneGap does not allow having a centralized development environment because it does not provide an IDE to develop applications. Instead, it provides a service called PhoneGap Build that allows developers to compile their applications in the cloud. Developers can also choice an IDE to write the source code and take it to an appropriate IDE for the target platform, e.g. Eclipse for Android, to additional code modifications.

Even if applications developed with PhoneGap are elaborate, they have more look and feel as web application than native application. PhoneGap is more adaptable for projects where cross platform reach is more important than high performances and user experience (Allen et al., 2010).

6.3. Example of Interpreted Approaches: Appcelerator Titanium

Appcelerator Titanium (2014) is a development environment for creating native applications across different platforms,

developed by Appcelerator Inc. Titanium applications are developed using web technologies as HTML5, JavaScript and CSS3. It uses JavaScript APIs and platform APIs to build the interface and to access native device features.

Titanium links JavaScript to native libraries, compiles it to bytecode and then the platform SDK builds the package for the desired target platform. The code is then packaged with Titanium's engine. At runtime, this engine interprets the JavaScript code and creates the user interface. The final application resembles to the typical platform appearance due to using native elements to build the user interface. However, application performances can be disappointing due to the interpretation of the source code every time the application runs.

Titanium also includes the Titanium Studio which is an Appcelerator's IDE. It is allows the writing, testing, and debugging of mobile applications. It is also offer the possibility to run and deploy the applications (Hartmann et al., 2011) (Vision Mobile, 2012) (Appcelerator Titanium, 2014).

Small numbers of platforms are supported by Titanium: iOS, Android, Windows Phone, BlackBerry OS, and Tizen with, by the latest stable release version 3.2.0 of the framework. 6.4. Example of cross-compiled approaches: Xamarin *Xamarin (2014b)* is a development environment for cross-compiled applications that allows developing applications using the C# language. Xamarin 3, the latest version of the framework, supports iOS, Windows Phone and Android.

To run the application on the device, the cross compiler Xamarin translates the code of the application writing in C# to binary code that can run on the target platform. Xamarin applications are built with native user interface controls. Thus, applications not only look like native application, they behave that way too. Xamarin also allows the access to all native devices and platforms features by providing all the native APIs of the supported platforms as C# libraries. Xamarin applications leverage platform-specific hardware acceleration, and are compiled for native performance (Xamarin, 2014c).

Xamarin offers also its own IDE named *Xamarin Studio (2014)*. Studio provides a convivial development environment with functionalities as code completion, debugger, etc. Xamarin Studio allows also distributing the applications through the application stores.

6.5. Example of Model Driven Approaches: AutoMobile

AutoMobile Project (2014b) appears as one of the most active and most promising in its category. AutoMobile is funded by the European Commission. The AutoMobile project exploits the modern paradigm of Model-Driven Engineering and code generation. It is based on abstraction, modelling and code generation to represent applications in a platformindependent manner, and then, generate the code to deploy the application and interaction logic onto the target platform. AutoMobile relies on modelling languages such as *IFML (2014)* (Interaction Flow Modelling Languages) and on tools like *WebRatio (2014)*.

What proposes AutoMobile: 1) a platform independent modelling language based on OMG standards (MDA, UML, IFML) for modelling applications 2) a set of software components and an architectural framework acting as technical building blocks based on HTML5 and also target native applications (iOS and Android) 3) a model-to-code generator, which consists of a set of model transformations integrated in the existing WebRatio platform (Automobile, 2014a). AutoMobile targeted only iOS and Android platform for now.

Table 1 summarizes the result of the evaluation. Our discussion let appear that although the user experience is not as good as native applications, the cross platform applications can be deployed in several platforms at once to reach out most of the potential users, which is essential for the application vendors:

7. Conclusion

The cross platform mobile development follows the concept: "Develop once and run anywhere". In other words, it promotes code reuse which reduces the development cost and time. Even if the user experience does not equal those of native applications, cross platform mobile applications can be deployed in several platforms. Given the competition that animates the market for mobile platforms, more and more developers are turning to cross-platform mobile applications solutions.

In this paper, we have distinguished between five cross platform development approaches. Then, we have described the general

	Sencha Touch	PhoneGap	Appcelerator Titanium	Xamarin	Automobile
Mobile platform supported	Only targeted at webkit browsers (Android browser, BlackBerry 10, WP8…)	Android, iOS, BlackBerry, webOS, WP 7 & 8, Symbian, Bada	iOS, Android, WP, BlackBerry OS, Tizen	iOS, WP, Android	iOS, Android
Development environment	NetBeans, Webstorm, Aptana, plugin for Eclipse	PhoneGap Build	Titanium Studio	Xamarin Studio	Not defined
API and documentation	Extensive documentation with large community of developers			Significantly smaller than for native and HTML5 based apps	Not defined
Security	Depends on browser security		Not good		Not defined
Access to device specific features	Access through Apache Cordova APIs	Access through JavaScript APIs	Access through JavaScript APIs and platform APIs	Access through C# libraries	Acces to all device features
Resource Consumption	Execution on browser engine may degrade the performances		Can be disappointing due to the interpretation of the code each execution	Significantly better with the use of native user interface controls	Same as native apps
Look & feel	More web application than native application		Resembles to the typical platform appearance due to using native elements		Pure native application
visibility	distributed through the application store				

Table 1. Tools evaluation based on desirable requirements

architecture of cross platform mobile applications. We have identified the desirable requirements of cross platform technologies. Based on these aspects, we have provided a detail survey that covers five cross platform tools (one per approach) allowing developers to gain insight about cross platform solutions. We have observed that the cross-platform tools can be recommended in general, even if none of such tool satisfied all the requirements.

We present this study as a first step toward defining standards, allowing the specification of cross platform mobile application development process. We are aware that there is still much work to do. More experimental studies are required to collect more data and support our findings. In the future studies, we will extend our analysis to additional cross platform tools. At the same time, we will further investigate requirements that call for more attention, such as security and resource consumption described in this paper.

References

[1] Adobe Flash Builder. (2014). Adobe Flash Builder. http://www.adobe.com/fr/products/flash builder.html.

[2] Allen, S., Graupera, V., Lundrigan, L. (2010). Pro Smartphone Cross-Platform Development: iPhone, BlackBerry, Windows

Mobile, and Android Development and Distribution. New York, USA: Apress.

[3] Anvaari, M., Jansen, S. (2010). Evaluating architectural openness in mobile software platforms. *In: Proceedings of the 4th European Conference on Software Architecture*, 85–92. ACM, Copenhagen, Denmark.

[4] Appcelerator Titanium. (2014). Titanium Mobile Application Development | Appcelerator Inc. http://www.appcelerator.com/titanium/. applause (2014). applause. https://github.com/applause/.

[5] Appsbuilder. (2014). Appsbuilder - Créez des apps et des sites mobiles. http://www.appsbuilder. com/fr/accueil.

[6] Automobile. (2014). Results | The AutoMobile Project. http://automobile.webratio.com/results/.

[7] Kravets, R., Krishnan, P. (1998). Power management techniques for mobile communication. *In: Proceedings* of the Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking *(MobiCom)*, October 1998, Dallas, Texas.

[8] Le Goaer, O., Waltham, S. (2013). Yet Another DSL for Cross-Platforms Mobile Development. *In: Proceedings of the 1st Workshop on Globalization of the Domain Specific Languages*, July 2013, Montpellier, France.

[9] Lee, J., Joe, H., Hyungshin, H. (2012). Smart phone power model generation using use pattern analysis, In: Proc. of the IEEE International Conference on Consumer Electronics (ICCE 2012), 412-413, Berlin, Germany.

[10] Lin, F., Ye, W. (2009). Operating System Battle in the Ecosystem of Smartphone Industry. In: Proc. of the International Symposium on Information Engineering and Electronic Commerce (ISIEEC 2009), 617-621,Ternopil, Ukraine.

[11] mgwt. (2014). mgwt - Making GWT work with mobile. http://www.m-gwt.com/. mobl (2014). mobl. http://www.mobl-lang.org/.

[12] Mosync (2014). Create native mobile apps for multiple platforms | HTML5/JavaScript and C++ developer tools. http://www.mosync.com/.

[13] Murmuria, R., Medsger, J., Stavrou, A., Voas, J. (2012). Mobile application and device power usage measurements. In: Proceedings of the IEEE Sixth International Conference on Software Security and Reliability (SERE 2012), 47-156, Gaithersburg, MD, USA.

[14] PhoneGap. (2014). PhoneGap | About the project. http://phonegap.com/about.

[15] PhoneGap (2014). PhoneGap | Home. http://phonegap.com/.

[16] Qt (2014). Qt - mobile app development. http://www.qt.io/mobile-app-development/.

[17] Rawassizadeh, R. (2009). Mobile Application Benchmarking Based on the Resource Usage Monitoring. *International Journal of Mobile Computing and Multimedia Communications*, 1 (4), 64-75.

[18] Rhodes. (2014). Rhodes - Motorola Solutions. http://www.motorolasolutions.com/USEN/ RhoMobile+Suite/Rhodes.

[19] Sencha Touch (2014). *Mobile App Development Framework*. http://www.sencha.com/products/touch/.

[20] Sommer, A., Krusche, S. (2013). Evaluation of cross-platform frameworks for mobile applications. *In: Proceedings of the 1st European Workshop on Mobile Engineering*, February 2013, Aachen, Germany.

[21] Vaadin TouchKit. (2014). Vaadin TouchKit - vaadin.com. https://vaadin.com/add-ons/touchkit.

[22] Vision Mobile. (2012). Cross-Platform Developer Tools 2012: Bridging the worlds of mobile apps and the web. Vision Mobile. http://www.visionmobile.com/product/cross-platform-developer-tools-2012/.Webratio (2014). WEBRATIO. http://www.webratio.com/.

[24] Xamarin Studio. (2014). Advanced IDE For iOS & Android Programming - Xamarin Studio. https://xamarin.com/studio.

[25] Xamarin. (2014). *Mobile* App Development & App Creation Software. http://xamarin.com/.

[26] Xamarin. (2014). Mobile Application Development to Build Apps in C#. https://xamarin.com/platform.

[27] Yuan, W., Nahrstedt, K. (2003). Energy-efficient soft real-time CPU scheduling for mobile multimedia systems. *In: Proc. of the nineteenth ACM symposium on Operating systems principles*, October 2003, Bolton Landing, NY, USA.