Cross platform approach for mobile aplication development: a survey

Mounaim LATIF
ERSI laboratory
Sidi Mohamed Ben Abdelah University
Fez, Morocco
mounaim.latif@usmba.ac.ma

Younes LAKHRISSI
ERSI laboratory
Sidi Mohamed Ben Abdelah University
Fez, Morocco
younes.lakhrissi@usmba.ac.ma

El Habib NFAOUI
LIIAN laboratory
Sidi Mohamed Ben Abdelah University
Fez, Morocco
elhabib.nfaoui@usmba.ac.ma

Najia ES-SBAI
ERSI laboratory
Sidi Mohamed Ben Abdelah University
Fez, Morocco
sbainajia@yahoo.fr

***Abstract*— Nowadays, the use of mobile technologies is rising at an alarming scale. Due to this, more powerful and efficient mobile applications are needed in order to keep up with this trend. Since there exists several mobile platforms (iOS, Android, etc…), each one with different SDK (Software Development Kit) tools and specific development capabilities, application development becomes more complicated and expensive. The challenge is to come up with a solution that allows us to deploy in different platforms using a single SDK tool and maintaining the same performance as the native application. A suitable solution is cross-platform. In this paper, we present a survey of cross-platform creation approaches with an emphasis on the MDA (Model Driven Architecture) approach as it is one of the most promising cross platform approaches. We also identify and discuss the main desirable requirements of any cross-platform technology.**

Keywords—Cross-platforms; mobile application development; Model Driven Engineering;

I. INTRODUCTION

Over a one-year period, the market for mobile cross-platform development has grown substantially. Companies have begun to adapt to new mobile trends by recognizing the need for smartphone access to business applications. This also creates the requirement to rapidly develop and deploy applications, which eventually puts under question the native approach consisting of the development of mobile applications separately for each platform [1][2][3]. The following table presents the differences between mobile platforms from a developer's perspective.

TABLE I. DIFFERENCES BETWEEN MOBILES PLATFORMS FROM A DEVELOPMENT POINT OF VIEW

OS	Programming Language	Development Environment	Application Store
Google's Android	JAVA	Android Studio, Android SDK	Play Store
Apple's iOS	Objective-C/Swift	XCode	Appel-iTunes
Microsoft Windows phone	Visual C#, C++	Visual Studio	Window Phone Market
RIM BlackBerry OS	JAVA	BlackBerry Plug-in for Eclipse	BlackBerry Apps World

The proposed solution to overcome the above issues in the native mobile application's development is the Cross-platform. This solution allows the code to be written once and deployed in several platforms. Through this, time and cost are reduced since the code is written to target multiple devices. In creating a cross-platform solution, several approaches are proposed. The ultimate purpose of these approaches is to achieve native app performance.

In this paper, we will give a comparison of existing approaches for cross platform based on different criteria. The first part is about the existing approaches for mobile cross-platforms. Then, we will introduce the desirable requirements of cross platform tools, and we will conclude by discussing the perspectives of cross-platforms for mobile development' application.

II. CROSS-PLATFORM APROACHES

A. Introduction

In application development, developers use the software development kit (SDK) to implement an application for a specific target platform. The application then becomes linked to that singular environment alone. For example, Android applications, which are programmed using Java. Android's framework allows access to the platform functionality and thus employs platform elements to render its user interface. iOS, on the other hand, uses Objective-C (or Swift, the new programming language developed by Apple) and Apple's framework Fig.1. The difficulty to develop mobile applications using many SDK's and frameworks motivate the implementation of cross-platform software development environments that could make the development easier and more efficient.

B. Cross-platforms overview

Following the Native Approach previously described, this section will provide a general overview of cross-platform approaches. These approaches are based on development with one environment and deployment in many platforms. The design of these approaches can give a beneficial result in time and cost minimizing, as it allows developers to write with one of the languages only and use a single framework that would be

978-1-4673-7689-1/16/$31.00 ©2016 IEEE

translatable to many platforms. These approaches can be categorized as follows:

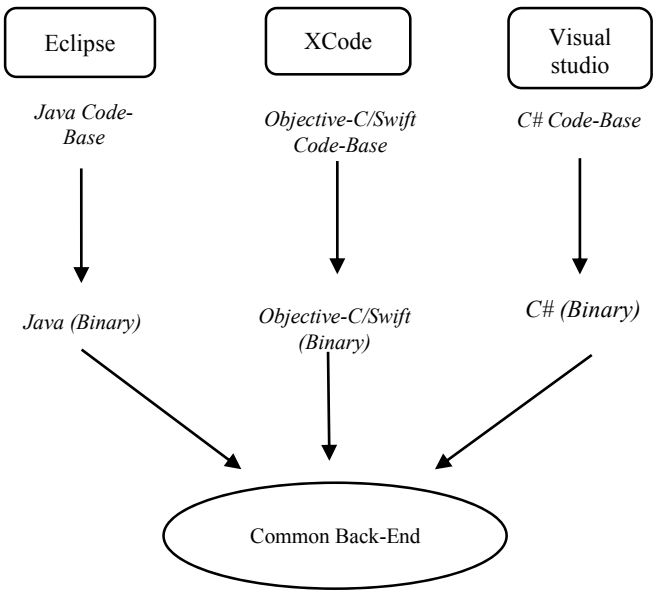

Fig.1. Native approach for mobile development

In the next sections, we will use the term '**Common Back-End'** to represent the application's backend including business logic and data processing/management, which is accessible via a standard fashion across all platforms.

1) Web Approach

The web approach is based on web browsers for mobile devices. Applications based on the web approach are implemented with the use of HTML, CSS and JavaScript; and rely on the browser as its runtime environment and benefit from the browser support of mobile platforms. Within this approach, the application is implemented as a single optimized website for mobile Fig.2. This optimization has to take into consideration the different screen sizes of the devices and their usage philosophy.

The advantage that comes of web mobile applications is that they exist in a similar fashion across mobile web browsers on all platforms. Thus, no mobile application updates are required. The drawback of the web approach is the fact that access to the device's native functionalities (such notifications system, GPS, Contact list, etc) is limited. A second disadvantage is that the time it takes to render the web pages by loading them from the network is longer than that of the native mobile user interface. Moreover, web applications are only accessible via a URL and cannot be made readily available on mobile app stores. This would have a diminishing impact on the approach's attractiveness.

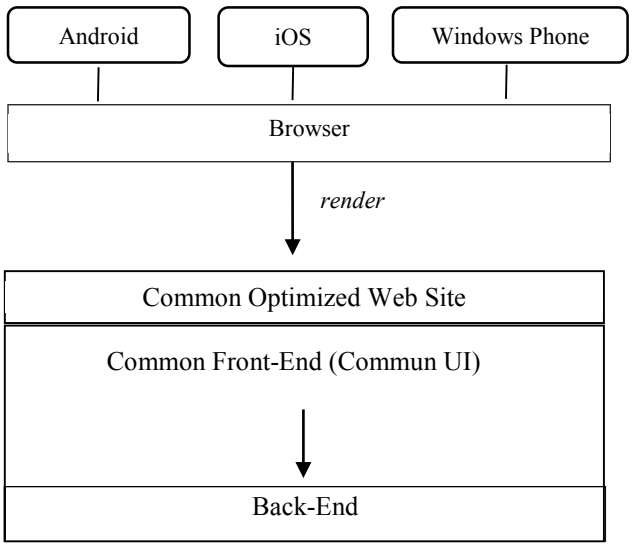

Fig.2: web approach

Jquery Mobile[4] is the most popular framework that use this approach to provide a single highly-branded responsive web site which is compatible with the most popular smartphone, tablet, and desktop platforms. Jquery Mobile is HTML5-based and it is built on the rock-solid jQuery and jQuery UI foundation. It offers Ajax navigation with page transitions, touch events, and various widgets. Its lightweight code that it is built with a progressive enhancement, and has a flexible, easily themeable design.

There exist other many solutions that allow the development of web application, for exemple Sensha Touch [5], jQTouch [6], HTML5 [7][8] .

2) Hybrid Approach

The Hybrid approach is a combination between the advantages of web technologies and those of native functionalities. This approach uses the browser engine in the device and embeds the HTML content in the native web container (WebView in android, UIWebView in iOS…). The native functionalities are accessible through the use of an abstract javascript bridge Fig.3.

As opposed to web applications, Hybrid applications are distributable through application stores and the native features are available through the abstract layer. However, hybrid interfaces are inferior in performance compared to the native interfaces since the execution happens in the browser engine. Moreover, the interface does not have access to the native look and feel. To get this, it becomes necessary to use specific development libraries.

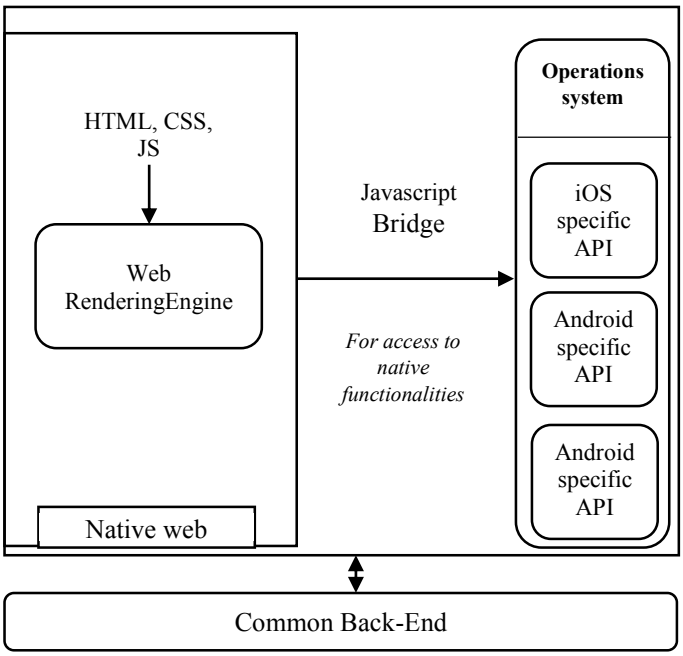

Fig.3. Hybrid Approach

Ionic Framework [9] is a recent hybrid solution for building native-feeling mobile apps based on HTML5, Ionic uses cordova plugins [10] to get mobile native features like notification, file storage, etc. It has more performance features than other hybrid solutions like Phonegap [11] which is the most popular hybrid framework -thanks to using an optimized new web technologies like Anglular JS [12] and SASS [13] There exist other hybrid solutions, such as MoSync[14].

3) Interpreted Approach

Interpreted approach use common language (like JavaScript or others) to write the code of user interface and generate the equivalent for native component for each platform. The native features are provided by an abstract layer that interprets the code on runtime across different platforms to access the native APIs Fig.4.

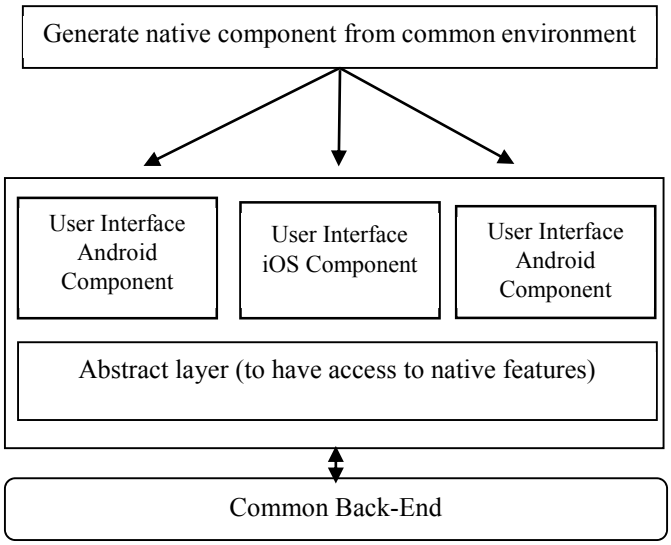

Fig.4. Interpreted Approach

The advantage of this approach is that it allows for native user interfaces. However, the downside is the dependence on the development environment. To be more exact, new platform-specific features such as new user interface features would not be made available to applications unless they are supported by the development environment. There is also an application performance degradation that is caused by calling the abstract layer on runtime.

Appcelerator Titanium [15] and Smartface App Studio [16] are the most poplar's interpreted environment using javascript to write user interface code.

Appcelerator Titanium is an open source tool based on this approach that allowing the development of mobile applications. It contains a framework called alloy, which provides an MVC (Model View Controller) development environment. On the other hand, Smartface App Studio presents more advantages with a WYSIWYG design editor, allowing for design of mobile apps and design fits. Moreover, it is the only cross-platform environment that allows the entire native iOS development process to be done on Windows (as an alternative to Mac-only Xcode).

4) Cross-Compiled Approach

In the cross-compiled approach (or generated approach), developers write codes with the use of any common programming language. These codes are transformed by cross compilers to a specific native code Fig.5.

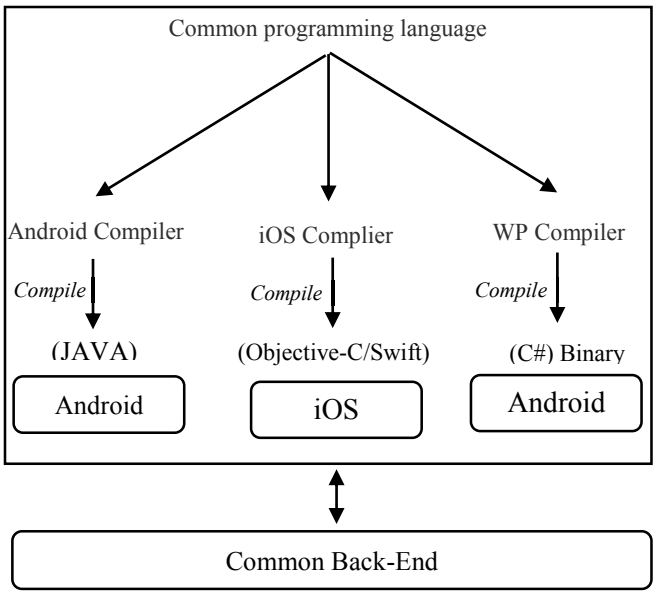

Fig.5: Cross-compiled Approach

The main benefit of this approach is that the applications are able to attain native performance and deliver all the features of native applications along with its native interface components. The disadvantage of this approach is that a few features cannot be used. Examples of these features are geolocation services and

camera access. This is because these functions are specific to that platform and access differs from platform to platform.

There exist two powerful platforms based on this approach; the first one is Xamarin [17], which uses c# shared codebase. Developers can use Xamarin tools to write native Android, iOS, and Windows apps with a shared code across a number of platforms and native user interfaces. However, Xamarin requires writing a specific code in order to benefit from the platform specific features; contrary to the second platform CodeName One [18] which gives a simple Java API that facilitates access to the platform specific features. However, this API is still in maintenance for the goal to take into account the platforms changes. Code Name presents more advantages by having a free and open source version, which is more generous than the free Xamarin version.

5) Model Driven Approach

This approach is based on Model Driven Architecture defined by the Object Management Group (OMG). The key of this solution is the fact that it is driven by modelling activities and a developer can describe an application on a high level, without having to deal with low-level technical issues such how to save data, system notifications, etc, at the core of the MDA [19][20] ,we have the three following models.

- Platform independent model (PIM) is a model of software that is independent of the specific platform used to implement it.
- Platform Specific model (PSM) is a model with the details that specify how that system uses a particular type of platform.
- Model transformation from PIM to PSM.

MDA is used to design the user interface once for multiple platforms (PIM) and then to generate the corresponding platform specific models (PSM) Fig.6.

The MDA approach provides the advantage of having the performance of native applications. The user interface is totally implemented with the native component without using a runtime intermediate. Developers have to maintain the model of the application in order to maintain the native code for each platform. The issue that arises with the MDA approach is the fact that it is limited to the application domain of the model language. The only applications that can be modeled are those that fall into the category supported by the model. On the other hand, the language and the code generator can be improved. The generated code remains incomplete and should be manually completed with the use of the native language and SDK tools. Thus, this code should be written individually for each platform, however, the manual integration of written code still a challenge.

There exists a prototype framework for model-driven cross-platforms for mobile application development called MD2 [21][22]. It is Based on a textual DSL and allows to generate runnable native apps for Android and iOS. However, it is still an incomplete solution that needs to be improved in order to handle the drawbacks of this approach.

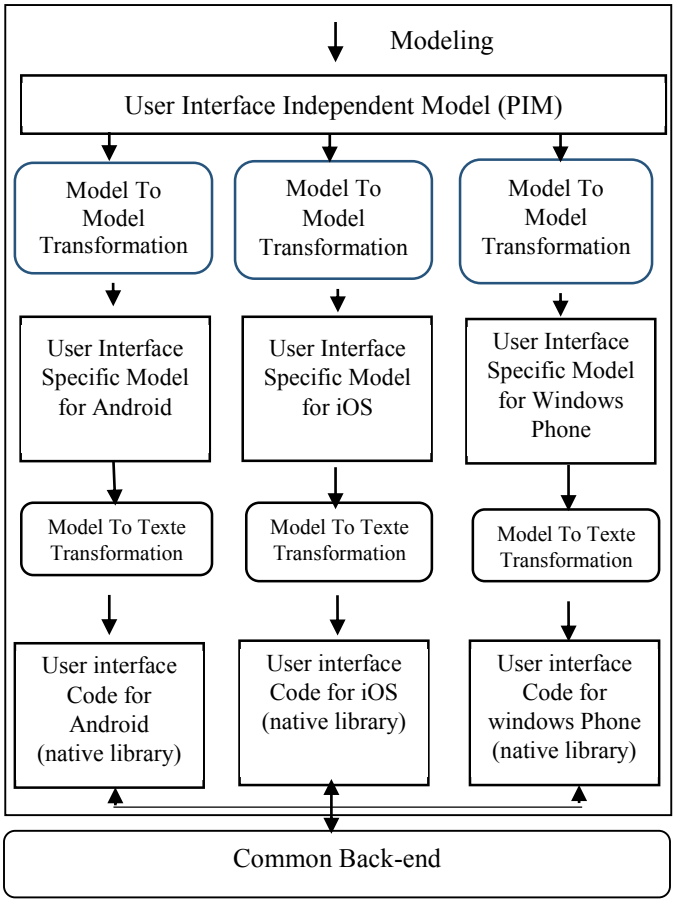

Fig.6: Model Driven Approach

III. THE REQUIREMENT FOR CROSS-PLATFORMS

The main objective of cross-platfoms is to produce mobile applications for multi-platforms, but it is important to take into account the requirements needed for producing and maintaining high performing mobile applications. Based on literature and our own analysis, we have identified the some desirable requirements of any cross-platform technology presented as follow:

A. Application Scalability and maintainability

The cross-platform must give the possibility to maintain and improve the application. For example, if new options need to be implemented in an application, the modifications have to always be in the cross platform level and thereafter deployed in deferent platforms.

B. Access to devices features

The application created using cross-platform must have all access to the device's features. It is preferable that this access be direct without an intermediate layer.

C. Resources consumption

Cross-platforms must take into consideration resource consumption including CPU, power consumption and memory usage. It is necessary for developers to take into account resource consumption, but the cross-platform must offer

maximum auto-optimization for mobile resources in the deployment for each platform. A number of studies have been done on this subject [23][24][25].

D. Security

The security issue is one of the most important criterions in mobile development. Part of this problem can be resolved by the developer. The developer must pay attention to which data is stored on the device, because devices may be stolen or otherwise fall into unauthorized hands. However, cross-platforms must have the capacity to respect the security policies defined by developers in all different OS [26] [27].

E. Devlopement envirenement

Development environment is a very important fact in cross-platforms. This is why we have to integrate all the needed debuggers, compilers, and intelligent auto-completion systems. But most importantly, we need to have multiple simulators with different systems (Android, iOS, windows Phone etc…) to get a good visibility on the application before deployment.

IV. CONCLUSION

The cross-platform solutions are recommended when an application is devoted to more than one platform, with time and cost limitations. There exist several approaches based on the principle "Develop once and run anywhere". In this paper we presented an overview of cross-platforms approaches. And as discussed in the above sections, it is certain that cross-platforms is the best solution for producing mobiles applications and is better than the native approach in terms of cost and time. However, all approaches and tools presented in this paper have their own limitations that may differ from an approach to another. Due to this, most companies in mobile application development are still using the native approach and only a small percentage have had satisfactory experience using the cross-platform approach.

We have notice that the MDA approach is the most expandable compared to the other approaches. Furthermore, the MDA architecture is the global trend for software development and it is due to this that our future studies will be focused on this approach. The goal will be to present an MDA based solution, while taking into consideration all aspects of mobile application development.

REFERENCES

[1] Henning Heitkötter, Sebastian Hanschke and Tim A. Majchrzak, "Comparing cross-platform development approaches for mobile applications," Web Information Systems and Technologies. pp. 120–138, LNBIP, Springer, 2013.

[2] Andre Charland and Brian LeRoux, "Mobile application development: Web vs. Native," In Communications of the ACM, Vol 54, pp. 49–53, Issue 5, May 2011.

[3] Paulo R. M. de Andrade, Adriano B. Albuquerque. "CROSS PLATFORM APP A COMPARATIVE STUDY". International Journal of Computer Science & Information Technology (IJCSIT) Vol 7, No 1, February 2015

[4] JQuery Mobile . Last access : December 2015. https://jquerymobile.com/

[5] Sensha Touch . Last access: December 2015. https://www.sencha.com/products/touch/

[6] Jqt. Last access : December 2015. http://jqtjs.com

[7] HTML5. Last access : December 2015. http://www.w3.org/TR/html5

[8] Xing Jin, Tongbo Luo, Derek G. Tsui and Wenliang Du. "Code Injection Attacks on HTML5-based Mobile Apps".

[9] Ionic Framework. Last access : December 2015. http://ionicframework.com

[10] Apache Cordova. Last access: December 2015. https://cordova.apache.org/

[11] PhoneGap. Last access: December 2015. http://phonegap.com/.

[12] Angularjs. Last access: December 2015. https://angularjs.org/

[13] SASS. Last access: December 2015. http://sass-lang.com/

[14] MoSync. Last access: December 2015. http://www.mosync.com/

[15] Appcelerator. Last access: December 2015. http://www.appcelerator.com/

[16] Smartface. Last access: December 2015.https://www.smartface.io/

[17] Xamarin. Last access: December 2015. https://xamarin.com/

[18] Codename One. Last access: December 2015. https://www.codenameone.com/

[19] L'uboš stará˘cek and Valentino vrani´c. "MDA Based Multiplatform Mobile Application Modeling with Platform Compliant User Interfaces".

[20] Florence T. Balagtas-Fernandez. "Model-Driven Development of Mobile Applications," IEEE/ACM International Conference on Automated Software Engineering. pp. 509-512 ,2008 [IEEE Computer Society Washington, DC, USA] .

[21] MD2. Last access: December 2015. http://wwu-pi.github.io/md2-web

[22] Henning Heitkötter, Tim A. Majchrzak, Herbert Kuchen. "Cross-Platform Model-Driven Development of Mobile Applications with MD2". SAC '13 Proceedings of the 28th Annual ACM Symposium on Applied Computing, pp. 526-533, Coimbra, Portugal March 18 - 22, 2013.

[23] Rahul Murmuria, Jeffrey Medsger, Angelos Stavrou. "Mobile Application and Device Power Usage Measurements" .

[24] Cassandra Beyer "Mobile Security: A Literature Review".

[25] Soumya Kanti Datta, Christian Bonnet and Navid Nikaein, "Android power management: Current and future trends," int the First IEEE Workshop on Enabling Technologies for Smartphone and Internet of Things (ETSIoT), pp.48,53, 18 June 2012.

[26] Jaymin Lee, Hyunwoo Joe and Hyungshin Kim, "Smart phone power model generation using use pattern analysis," 2012 IEEE International Conference on Consumer Electronics (ICCE), pp. 412,413, 13-16 Jan. 2012.

[27] Denim Group, "Secure mobile application development reference", 2010-2011.