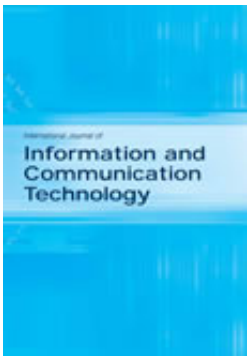


International Journal of Information and Communication Technology

ISSN online: 1741-8070 - ISSN print: 1466-6642
https://www.inderscience.com/ijict

Classification of existing mobile cross-platform approaches and proposal of decision support criteria

Ayoub Korchi, Mohamed Karim Khachouch, Younes Lakhrissi, Nisrine El Marzouki, Aniss Moumen, Mohammed El Mohajir

DOI: 10.1504/IJICT.2024.10060618

Article History:

Received:	29 May 2020
Last revised:	04 October 2021
Accepted:	16 December 2021
Published online:	05 December 2023

Classification of existing mobile cross-platform approaches and proposal of decision support criteria

Ayoub Korchi*, Mohamed Karim Khachouch, Younes Lakhrissi and Nisrine El Marzouki

Intelligent System,
Georesources and Renewable Energy Laboratory,
FST of Fez,
Sidi Mohamed Ben Abdellah University, Morocco
Email: ayoub.korchi@usmba.ac.ma
Email: mohamedkarim.khachouch@usmba.ac.ma
Email: younes.lakhrissi@usmba.ac.ma
Email: elmarzoukinisrine@gmail.com
*Corresponding author

Aniss Moumen

Engineering Sciences Laboratory,
ENSA of Kenitra,
IbnTofaïl University, Morocco
Email: amoumen@gmail.com

Mohammed El Mohajir

Faculty of Sciences,
Abdelmalek Essaâdi University,
Tetouan, Morocco
Email: m.elmohajir@ieee.ma

Abstract: The smartphone market has known an exponential growth since 2007, with the apparition of the first Apple phone. Nowadays, developing an application that targets all existing mobile platforms, becomes a tedious task for developers, due to the diversity of mobile platforms, their tools. For that, cross-platform approach with its various sub-approaches has shown its strength in reducing project's cost and time respecting the slogan 'develop once and deploy everywhere'. This paper aims to compare the mobile app development's approaches and suggests a decisional framework to choose the adequate one to get an application respecting the client's need with a low cost and time. This framework's criteria have a huge impact on the eventual cost, time, and success of an application building. If developers fail to match the demands of an app with the right development approach, it can turn their project into a certain failure.

Keywords: OS comparative study; mobile cross-platform; MDE; MDA.

Reference to this paper should be made as follows: Korchi, A., Khachouch, M.K., Lakhrissi, Y., El Marzouki, N., Moumen, A. and El Mohajir, M. (2024) 'Classification of existing mobile cross-platform approaches and proposal of decision support criteria', *Int. J. Information and Communication Technology*, Vol. 24, No. 1, pp.86–111.

Biographical notes: Ayoub Korchi is a Software Engineer from National School of Applied Sciences of Fez, Morocco in 2018, and since that he has been an SAP Technical Consultant and has worked for the account of many multinational enterprises. He is a PhD student at Sidi Mohamed Ben Abdellah University. His research interests include models, cross-platform approaches, MDA, mobile code. He is a SIGER Laboratory Fellow.

Mohamed Karim Khachouch is a Software Engineer from National School of Applied Sciences of Fez, Morocco in 2018, and since that he has been a Dotnet Developer and has worked for the account of many multinational enterprises. Actually, he is a PhD student at Sidi Mohamed Ben Abdellah University. His research interests include models, cross-platform approaches, MDA, mobile code. He is a SIGER Laboratory Fellow.

Younes Lakhrissi is a Professor of Computer Science, in National School of Applied Sciences-Fez, Sidi Mohamed Ben Abdellah University, and Doctor of Computer Science – Software Engineering from Rabat-Agdal University and Toulouse University. He is a member of SIGER Laboratory, FST of Fez, Morocco.

Nisrine El Marzouki is a Doctor of Riga Technical University Latvia and Sidi Mohamed Ben Abdellah University-Fez Morocco. Currently, she serves as a Senior Developer at ALTRAN in France. Her research interests include models specification and composition, constraints resolution, and MDE.

Aniss Moumen is a Professor of Computer Sciences in National School of Applied Sciences, University of Ibn Tofail Kenitra. He is a member of Engineering Sciences Laboratory. He is interested in artificial intelligence, machine learning and neural network. He is an editor of *MJQR* journal.

Mohammed El Mohajir is a Doctor of Science from the University Catholic of Louvain, Belgium. He is a Professor at the Faculty of Science Dhar Mahraz, Sidi Mohamed Ben Abdelah University. He lectures databases, information systems and business intelligence for both undergraduate and master levels.

This paper is a revised and expanded version of a paper entitled 'Classification of existing mobile cross-platform approaches' presented at International Conference on Electrical, Communication, and Computer Engineering (ICECCE), Istanbul, Turkey, 12–13 June 2020.

1 Introduction

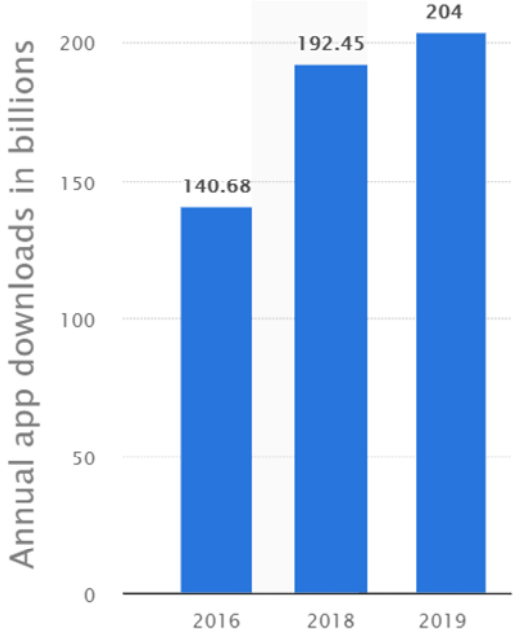
Since 2007, the use of smartphones had increased due to the variety of mobile applications proposed in each operating system provider such as Play Store for Android, App store for iPhone, Windows store for Windows phone, etc.; whether in the personal or professional purpose, covering several daily live domains such as: social media, lifestyle,

payment, utility, games and entertainment, productivity, news and information outlet, enterprise specific app, etc.

As it shown in Figure 1, we conclude that the use and the download of mobile applications are still growing up with over than 140 billion applications downloaded in 2016, and still growing up to 204 billion in 2019. (Pinto and Coutinho, 2018)

In this subject, statistics made by Gartner show that the smartphone sales had increased by more than 65% between the first quarter of 2012 and the third quarter of 2018 (OS smartphone: key values), with overall revenues in 2018 reached 365.2 billion US dollars (mobile app revenue) and numbers are still breaking the records.

Figure 1 Number of mobile app downloaded 2016–2019 (see online version for colours)



According to the same statistics mentioned above (OS smartphone: key values), since 2012, android has monopolised the mobile market with a part of 88% in 2018, the rest was taken by IOS and other OS. This variety of mobile distribution causes problems for enterprises, whom have the intention, to provide mobile applications targeting all smartphone's platforms.

Developing an application, having all device features access, native look and feel, targeting all the existing mobile platforms, becomes a really difficult task, due to the variety of mobile operating systems, their programming languages, their IDEs, and the rest of specific tools of each platform.

For this purpose, cross-platform's solution has been proposed by researchers with several approaches such as: compilation, component-based, interpretation, cloud-based, merged and MDA. All these approaches lead to facilitate the mobile application's development to target all existing mobile's platform. In this paper, we present the six approaches mentioned above, with their pros and cons, then we propose a decisional framework based on several questions in order to pick up the right approach that will be

used in development's process. Since each approach of them can deal with a specific project's needs, depending on the application and the target platforms specifications, so it is rather to proceed with this proposed decisional framework to choose the right approach based on a deep study of project's needs and specifications, than choosing an approach randomly taking only in consideration the facility of the approach and not comparing its advantages and its disadvantages. This decisional framework is based on multiple criteria taking the form of questions that can be used in totality or just some of them depending on the project's priorities, then giving a weight number to each question based on the importance of each criterion used in this decisional framework. More clearly, to take the right decision, the project manager must make a deep study depending on many constraints by answering multiple basic questions relative to the project nature, about the targeted platforms, the native look and feel, the access to device features, etc. This study must be done taking into consideration obligatory the budget and the time granted to the present project.

The remaining of this paper is organised as follow: Section 2 contains some related works with a discussion about each work's study. Section 3 presents a quick overview of the three most known operating systems. Section 4 compares all the mobile development approaches that exist in the market. Then, a framework decision metric that aims to help a project manager to take the right decision concerning the approach will be found in Section 5. Finally, Section 6 concludes our paper and gives an overview of our future works.

2 Related works

In Meirelles et al. (2019), the authors classified mobile applications into native and cross-platform applications. Cross-platforms itself, is divided into web application when the source code runs in a remote server, and hybrid applications in case of a web app installed in a mobile device using a container wrapper to access native features. The authors then tried to compare these approaches depending on a questionnaire. This classification does not take into consideration all cross-platform approaches such compilation, component-based, interpretation, cloud-based, merged and modelling, and consider only web and hybrid approaches.

In Korchi et al. (2020), the authors classified cross-platform into six approaches: compilation, component-based, interpretation, and MDA approaches. They gave a description, pros, and cons of each approach. However, this classification remains incomplete and not really detailed since it lacks some other cross-platform approaches such as the merged and cloud-based approaches.

In Latif et al. (2016), the authors classified the cross-platform to five approaches: web, hybrid, interpreted, cross-compiled, and model driven approach. The authors compared the advantages and the disadvantages of each one, presented the requirement of cross-platforms applications. However, the classification was too much limited forgetting some other approaches such as component-based approach, trans-compilation approach, which is a sub approach of the compilation one.

In Latif et al. (2017), the authors kept the same five approaches classified in paper (Latif et al., 2016) and gave a quick view on some solutions to develop an application using each one of those five approaches presenting the advantages of each solution, but no demonstration were provided.

In Ebone et al. (2018), the authors classified applications into native and cross-platform types, and then classified the cross-platform into web/hybrid apps, proxy native apps, and cross-compilation apps. They gave an example of existing solution for each platform; Apache Cordova for web/hybrid apps, Appcelerator Titanium for proxy native apps, and Xamarin for cross-compilation apps. The authors used Android and IOS platforms to compare each solution's final app with its native equivalent. This comparison was based on building time, rendering time, total UI response time, memory storage, and app size. This classification remains incomplete since, it does not regroup the rest of cross-platforms approaches.

In Yang et al. (2017), the authors compared and classified mobile application development into three types: native, web and hybrid. The comparison was based mainly on cross-platform and equipment capability, development difficulty, application experience, backward compatibility. They concluded that the hybrid remains the best solution using multiple frameworks and development languages such Ionic Angular, Cordova, HTML, CSS. This solution may be the best for this case, but it needs a high level of development learning efforts and can be changed if the project's needs change.

In Lachgar and Abdali (2017), the authors enumerate in a first time the different existing development approaches to native, web, and hybrid approach, giving some examples of an app architecture development following the concerned approach. Then they compared every approach with another one following some criteria based on development environment needed for each approach. Finally, they proposed a framework that helps to pick up the most appropriate approach for a project's needs. The classification lacks the cross-platform approach which remains the best solution for reducing cost and time for mobile application development.

In Khachouch et al. (2020), the authors analysed different development approaches to classify them and identify the best one. To do so, they developed a decisional framework to help decision makers to choose the best and the most adequate approach. This framework was based on series of questions.

In El-Kassas et al. (2017), the authors in the first time classified the mobile application into three types: web, native, and hybrid type. Then they classified the cross-platform into six main approaches: compilation, component-based, interpretation, cloud-based, merged, and modelling approach. They gave the pros, cons and some solutions of each approach but no demonstration was provided. This classification was well-detailed and regrouped almost all cross-platform approaches, however we can't find even a paragraph talking about the MDA approach which is an OMG's particular vision and a subset of the MDD approach classified in the same article.

In Sabraoui et al. (2012), the authors proposed an approach that consists of three basic steps. Firstly, they defined a PIM using object diagram corresponding to the mobile application meta-model proposed in the same article, so as to model the application GUI. Then they transformed that model to a simplified XMI file using JDOM API, in order to use the MDA approach to transform that PIM to a PSM using meta-models of each platform for mapping, ATL languages for defining rules, and then generate the GUI code.

In Benouda et al. (2016), the authors proposed an approach to generate graphical user interfaces for an android application using an MDA approach. Their approach consists of two transformations; the first one is a model-to-model transformation using QVT language to transform PIM to PSM, and second one is a model to text transformation using acceleo tool to transform PSM to code in order to generate and facilitate the development of Android applications. Their approach concerns only UI generation and

lack access to resources, embedded sensors, events manager, generating complicated user interfaces, etc.

In Benouda et al. (2017), the authors proposed an approach that leads to generate a mobile phone application, they kept the same source meta-model, which is the same of a class diagram as the last article. They developed an Ecore corresponding to the source and target meta-model, and then implemented the transformation algorithm using QVT language. Their approach uses the 'translationist' approach which can translate a PIM into the final code using a sophisticated code generator which use the PSM as an intermediate step during the code generation process. In this article, the authors did not specify nor make a demonstration of this code generator.

In Salma et al. (2018), the authors proposed a cross-platform solution called TimPhoneGenerator based on MDA approach. This solution consists of five steps: firstly, they defined the targeted platform, then studied the basic features related to each platform such as data storage, software architecture, user interface, access to phone data, communication between applications, and access to phone features. The 3rd step consists of defining a PIM, in order to generate a PSM in the 4th step, and finally generate mobile code using eclipse modelling framework (EMF), which is used to design Ecore model using PSM generated and Acceleo to generate code from Ecore designed above. The solution proposed uses a PIM composed of 6 PIMs concerning data storage, Cross-app communication, software architecture without presenting the three others. However, this solution still has some lacks such as the transformation from PIM to PSM which is done manually, the code generated still need some manual modifications.

In Lachgar and Abdali (2014), the authors adopted the MDA approach to generate the graphical user interfaces using GUI DSL model as PIM which is implemented via Xtext solution. In a second time, they applied M2M transformation on the GUI DSL model to transform the PIM to PSM respecting generation rules. Finally, they generated a source code from the obtained PSM using a template developed with Xtend 2. This paper shows how to generate UI, without giving interest to other mobile development aspects such as sensors.

3 Three most used operating systems

In this section, we will present the three most used operating systems Android OS, iOS, and Windows phone OS.

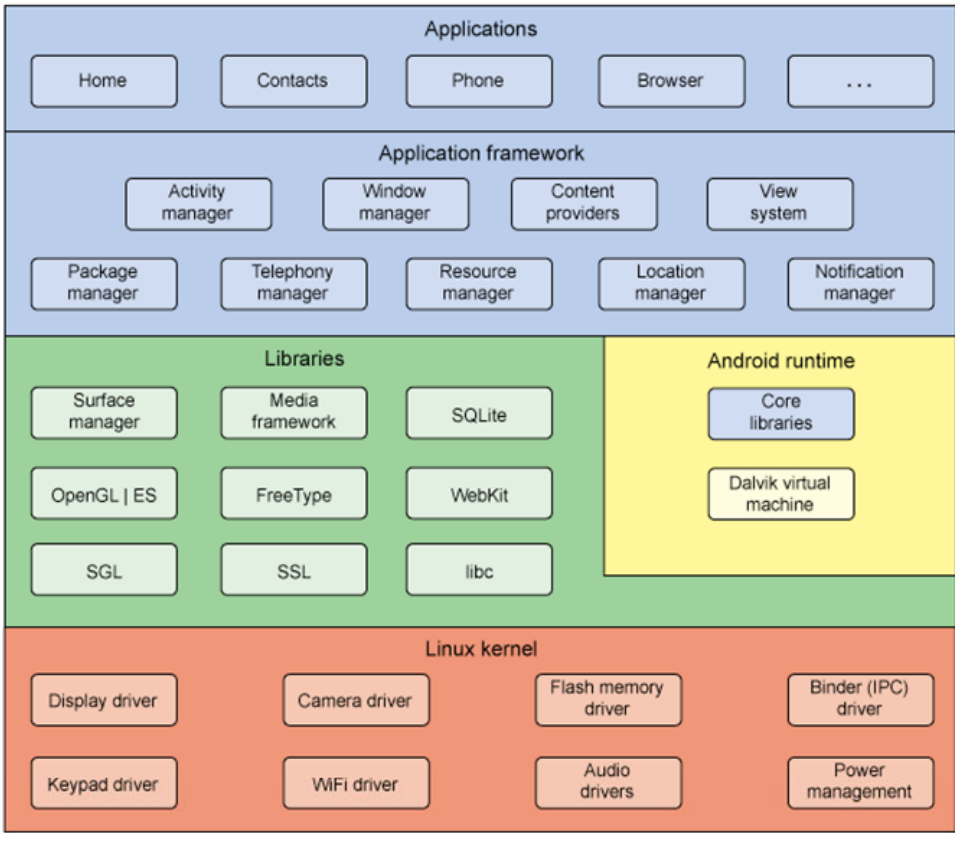
3.1 Android OS

Android is the monopolist of mobile operating systems with a market part of 86.7% in 2019 (OS smartphone: key values). Android was developed by Google, and designed firstly for only smartphones and tablets, but it has been then used also for televisions, cars, and smart watches.

As shown in Figure 2, android is based on Linux Kernel, which constitutes the first layer of the architecture and the abstraction one between the device hardware and the rest of the layers. It contains the entire important hardware drivers like Bluetooth driver, camera driver, Audio driver, etc. Above this layer, we find the libraries layer, which contains some C/C++ core libraries and Java based libraries such as SSL/SGL, media framework, graphics, SQLite, etc. used by some Android OS components. Within this

layer, we find a sublayer called Android runtime that contains some core libraries that allow developer to use java as the android programming language. It contains also the famous virtual machine Dalvik that is used as a just-in-time (JIT) process, in order to compile java classes to Dalvik byte code. This sublayer has brought a new method called ahead-of-time (AOT) compilation, which is performed at the application installation in order to reduce energy consumption by reducing resource cost and CPU usage. However, it needs more time and storage space for the Bytecode generated (Novac et al., 2017). The Third layer is the application framework. It provides many classes and interfaces required for creating an android application. It regroups Android API that can be required for the application development such as UI, telephony service, notification management, location service, etc. In top of the stack, we find the application layer, where all the application are be installed. (Divya and Kumar, 2016)

Figure 2 Android OS architecture (see online version for colours)



According to statistics, published in (number of applications available in leading app stores) 2.43 million android applications are available in the Google play store, which represent the android application market. All the applications were written in java only until 2017, when Google announced his preferred Android language, choosing the

Kotling-firt strategy, and actually more than 50% of android professional developers use it to develop their apps. The version 10 of the Android mobile operating system is the latest version. It was developed by Google and have been released since September 2019. This latest version came with new features and updates on:

- live caption with no Wi-Fi requirement
- smart reply that will suggest responses to messages
- sound amplifier by boosting sound and filtering background noise
- gesture navigation quicker than the previous versions
- dark theme that reduces battery consumption
- privacy controls by keeping data more private and let the user choose when and how the data can be shared
- security updates can now be sent directly to the smartphone from Google Play store, focus mode by pausing turning off the distracting application just by a simple tap
- family link by observing kids while using the phone, managing apps and content.

3.2 iOS OS

IOS is the second most used OS with a market part of 13.3% in 2019 (OS smartphone: key values), and the strongest one since the OS and the final products are developed and maintained by the same company. Apple developed IOS for only their devices. It was primarily designed for iPhone, but it has been then implemented on iPod touch, iPad, Apple TV.

IOS architecture is divided into six layers as illustrated in Figure 3. Core OS is the closest level to hardware with UNIX multitasking kernel called XNU. It contains the file manager, and series of drivers. It holds low level features that most other technologies are built upon such as Core Bluetooth framework, accelerate framework, external Accessory framework, security services framework, local authentication framework (Divya and Kumar, 2016).

Figure 3 iOS Os architecture (see online version for colours)

The second layer is core services which provides basic services such as: address book framework, cloud kit framework, core data framework, core foundation framework, core location framework, core motion framework, foundation framework, healthkit framework, homekit framework, social framework, storekit framework, that roughly manage contacts database, import/export data between app and iCloud, use hardware features like GPS compass accelerometer, handle health-relation information of the user, monitor the home's devices. Above this layer, we find media level, which is used for multimedia transfer and enabling graphics, audio, and video technology. It provides the following frameworks: UIKit graphics, core graphics framework, core animation, core images, OpenGl ES and GLKit, metal, that respectively gives support to design images, animate view, 2D vector, optimise the animation experience, play playlists, and use iTunes library, etc. The last level is Cocoa touch. It represents the iOS user interface that developers interact with while developing an application. It gives support for touch, motion event VoIP app, generating tweets and creating URL to access the twitter service, including scrollable map into app UI, sharing game related information online, using the following frameworks: ventKit framework, GameKit framework, iAd framework, MapKit framework, PushKitFramework, Twitter framework, UIKit framework.

According to the same statistics mentioned earlier (Delía et al., 2017), the IOS application market App store contains 1.8 million applications in the third quarter of 2019. Objective-C and swift are the two main programming languages for IOS applications, expect that swift is safer than objective-C.

IOS13 is the latest version of IOS, released on September 19, 2019. It brought some features presented in howtogeek website, such as:

- system-wide dark mode
- silence unknown callers, is an option for muting or blocking call from a number not existing in the contact list, which can be for example a robot-call
- new text-editing gestures, which facilitate text and text-entry cursor manipulation
- improved location permissions that allow user to give the application a temporary location access
- Bluetooth privacy controls, so like that all the applications will have a restriction to use Bluetooth.

3.3 Windows phone OS

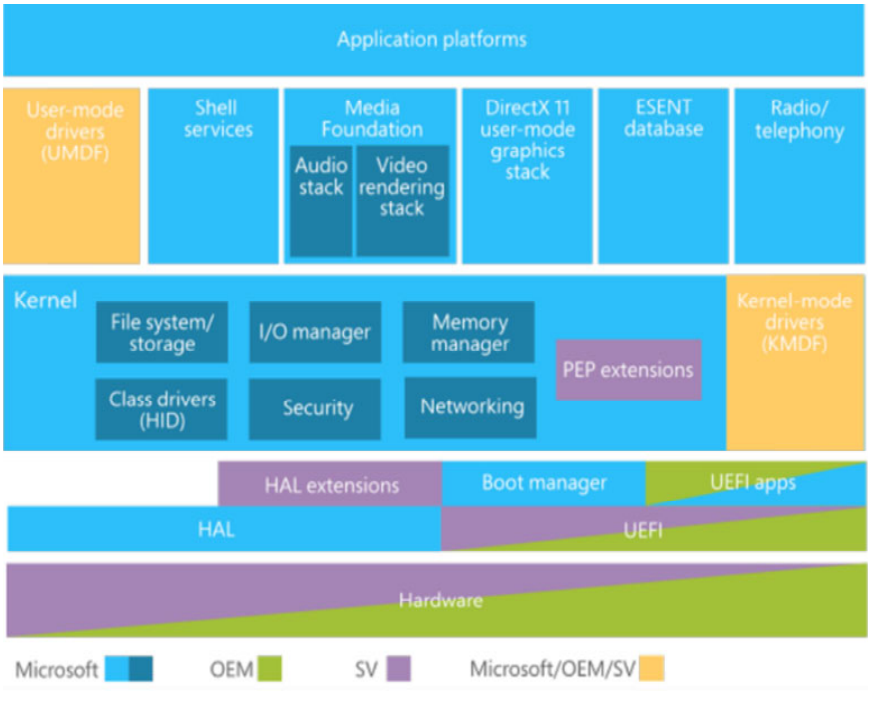
Windows phone is a mobile operating system developed by Microsoft, positioned as the third most used mobile OS after Android and IOS. It has the world's best user interface (Novac et al., 2017).

As shown in Figure 4, windows phone OS architecture starts from the hardware layer, then we find the kernel layer, which is based on Windows CE kernel. It holds low-level device drivers access and provides some OS functionalities such as process management, thread management, memory management, and file management.

Above the kernel, we find the three libraries: App model, UI model, and cloud integration that are respectively responsible for application management, user interface management, and web search via Bing, locations services, and push notifications. Common language runtime (CLR) is used as virtual machine to compile the application files developed by C# which is considered as the main programming language for Windows phone OS (Okediran et al., 2014). According to the same statistics mentioned earlier (Number of applications available in leading app stores), more than 60,000 applications are available on windows phone market, which represents a small interval of applications compared to Android and IOS applications, but we can find the 50 most downloaded applications in either windows phone.

Windows phone 10 is the latest version of windows phone, released on March 17, 2016. According to Microsoft, phones working with Windows 10 will not receive security and quality updates anymore since June 11, 2019.

Figure 4 Windows phone OS architecture (see online version for colours)



3.4 OS comparative study

In order to sum up all information mentioned above, we set up the Table 1 to compare the three operating systems presented in this section. Table 1 regroups some parameters and specifications of each OS such as the founder, kernel, development language, virtual machine used to compile files, application market, and the last version of each OS.

Table 1 Summary comparative of the OS

	Android OS	*iOS OS*	*Windows phone OS*
Founder	Google	Apple	Microsoft
Kernel	Linux	Unix	Windows CE
Dev. language	Java/Kotlin	Objective-C/Swift	C#
UI	XML files	Cocoa touch	XAML files
Virtual machine	Dalvik	---	CLR
IDE	Android studio	Xcode	Visual Studio
App. market	Play store	App store	Windows phone Store
Last version	Version: 10	Version: 13	Version: 10
Last update	2019	2019	2016

Android OS takes a large part of market with a percentage of 88% in 2019, according to a study mentioned above (OS smartphone: key values). The free of charges of almost all applications available in play store is the main advantage that allows Android to win this marketplace. Also, the diversity of applications even out of the play store plays a role in favour of Android market.

4 Mobile application development types

In this section, we will explicit and give more details about the four known mobile development approaches counting the native, web, hybrid, cross-platform with their six approaches such as the modelling, cloud-based and merged approaches with their pros and cons.

4.1 Native applications

Native mobile applications are developed using platform specific tools and languages. To target multiple platforms, developers must make more efforts to learn and use specific languages and IDEs for each platform such as java with Android studio for Android, C# with Visual studio for Windows phone, objective C with Xcode for IOS. The result application will be available on the specific platform's stores such as Play store for Android, App store for IOS, and Windows phone store for Windows phone. Each platform's app stores have an audit process to evaluate the adaptation of the application with its specific platform on which it is installed (Delia et al., 2017).

This type of applications presents the advantage of providing the native look and feel of each platform, with high performance application, since it uses the specific elements of each platform. Then, this application has the access to all native features, it has also access to mobile device APIs, that allows application to use the various existing sensors, and some other mobile basic application services such as files, local storage, messages, contacts, etc.

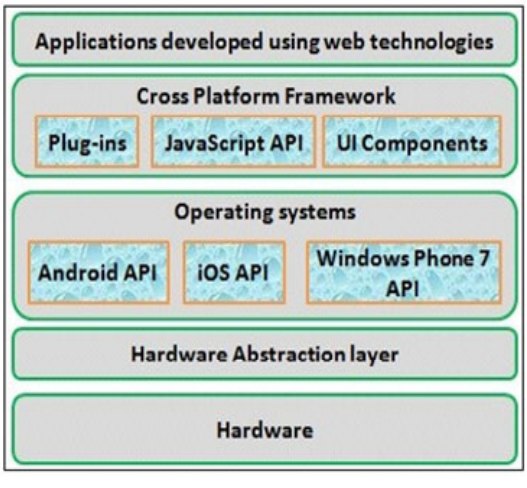
However, this type of application needs a lot of effort, time, and cost for developing the application, which can be multiplied by the number of targeted platforms, due to the steep learning curve.

4.2 Web applications

A web mobile app is built to be executed on a web browser. It is developed using web technologies like HTML, CSS, and JavaScipt. In this approach, none of the app components are installed on the mobile device. Rather than that, the app is accessible from a URL and the app data are manipulated by the server (Jobe, 2013). The following Figure 5 shows the architecture of a web application.

To access this type of mobile applications, there is no need to install it on the device, which present a storage advantage. In consequence, the application updates don't depend on the device updates. The web application may also present the advantage of reusing the user interface code, but it does not have the look and feel of native mobile applications, neither could it access to native features such as device sensors. The web application is not available in the app stores and relies on internet connection speed which can be a big problem of this type, especially when it concerns payment tasks. Also, to monetise a web application is not as simple as native one (Dalmasso et al., 2013).

Figure 5 Web application architecture (see online version for colours)



4.3 Hybrid applications

This approach combines between native and web approaches. It depends on web technology such as HTML5 CSS, JavaScript for user interface, and some native wrapped code to access mobile features. (Palmieri et al., 2012)

Figure 6 Hybrid app architecture (see online version for colours)

The native wrapper is the responsible of packaging, deploying, and distributing a mobile application across all targeted platforms. Also, the hybrid applications can run in the different platform's types counting the desktop, the web, and the mobile platforms (Pinto and Coutinho, 2018).

Figure 6 shows the hybrid app architecture (Dalmasso et al., 2013), which is based on Cordova framework, that we tried to schematise presenting the interaction between layers starting from the OS kernel to the developed application using the web technologies (Raj and Tolety, 2012).

This type of applications presents a lot of advantages in common with the two last types, since it is considered as a combination of native and web applications. Concretely, this type of applications could be available in all platform's app store, it relies on device computing capabilities in runtime. Hybrid application can access device features using a hardware abstraction layer as a bridge, and benefits from the use of UI code over multiple platforms but presents a modest native look and feel, especially for common UI elements and functionalities, and not for the newest ones. Its performances can't match with native apps, since they're only executed in the platform web engine, and suffering from platform specific behaviour of JavaScript and threading model incompatibilities with JavaScript. Hybrid application suffers also from high battery consumption which is 250% higher than its native equivalent application (Meirelles et al., 2019), with an amount of RAM used accounted to almost the double of the RAM used by its native equivalent. The same thing for CPU percentage used by hybrid application which expects to triple compared to its native equivalent. Native applications are also slower in terms of starting-up time and screen-transition compared to its native equivalent (Bosnic et al., 2016).

4.4 Cross-platform applications

In this subsection, we classify the cross-platform into six main approaches such as: compilation, component-based, interpretation, cloud-based, merged and modelling focusing especially on its famous method which is the model driven architecture (MDA), taking into consideration that each one of the cross-platform approaches derives from one of the three first mobile application development types which are: the native, the hybrid, and the web types.

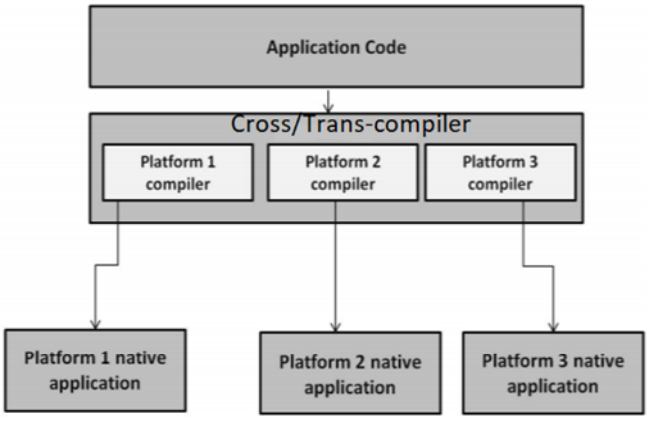
4.4.1 Compilation approach

This approach consists of a program called the compiler that transform a source code written in a high-level language to a target code written in low-level language. This approach has two sub-categories: cross-compilation and trans-compilation. When the OSs on where the compiler and the compiled program run are different then it is a cross-compilation approach.

When we transform a high-level language to another one, then it is a trans-compilation approach. In other words, the 'transpiler' is a set of tools that generates the targeted source code after a 'transpilation' process from the original high-level programming language (Andrés and Pérez, 2017).

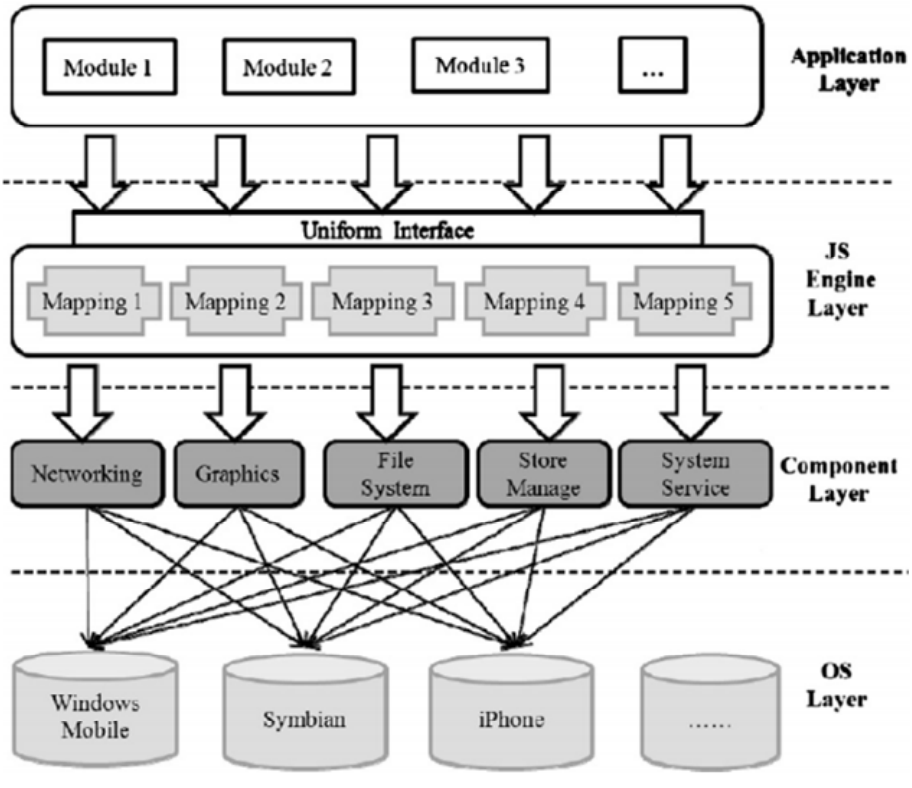
Figure 7 shows the architecture of a compilation approach with its two sub-approaches cross-compilation and trans-compilation, that depend on a third system to generate the native application code.

Figure 7 Cross/trans compilation application architecture



This approach benefits from the reuse of the existing code, to get and produce a native application in another platform, but it presents some difficulties in mapping between source and target languages because it focuses only on common elements of the platforms, and common APIs of the two languages.

Figure 8 Component-based application architecture



4.4.2 Component-based approach

The component-based approach consists of regrouping related system methods and functions into defined interfaces that must be implemented differently for each platform but keeping the same interface for all the platforms.

Figure 8 shows the architecture of a component-based mobile application (El-Kassas et al., 2017).

This approach simplifies the support of the new platforms by implementing a set of related functions of this one with their interfaces, but it adds a new difficult task to the developer who must learn how to use the component interfaces, and also focuses on only common functions belonging to all supported platforms.

4.4.3 Interpretation approach

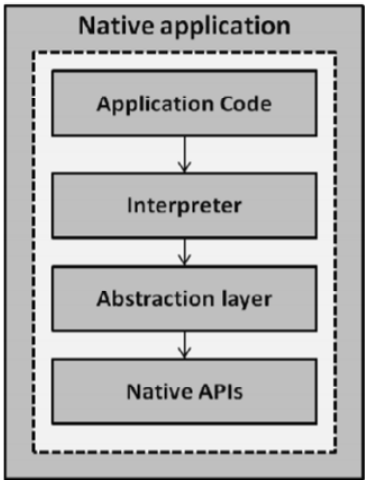
This approach uses a common language to write the UI code and then its equivalent will be generated for each platform by using an interpreter. It has two sub-approaches:

a Web sub-approach, which consists of creating a responsive web application that will be accessed through all platforms via URL, and the native features, will be accessible by implementing wrappers.

b VM-based sub-approach: that consists of two methods. The first one is developing an application with a platform-independent language and then deploy it in a virtual machine that is already installed in each platform. The second one is to translate the app code into bytecode that will be executed by a virtual machine at runtime. The best example to illustrate the second method is the MobDSL framework. With MobDSL, mobile apps are built using DSL and ran on a dedicated virtual machine. (Kramer et al., 2010)

Figure 9 shows the architecture of an interpreted application, which needs generally two other layers to interpret the code, then a bridge to access native features and APIs.

Figure 9 Interpreted application architecture



This approach is the easiest compared to the last approaches, in terms of development technologies, storage size and downloads times. However, the produced app does not have the native look and feel, presents performance degradation due to the abstraction layer called at runtime, moreover the Virtual machine is not available in App store for iOS (Raj and Tolety, 2012)

4.4.4 Cloud-based approach

This approach consists of hosting all the app backend on a server so that all the computations and processing are executed on a distant server. Many companies, such as Amazon Web Services, Heroku, offer that kind of services called Baas (Backend as a service). Well-known companies have chosen this approach for their products such as Facebook, Google (Google Drive for instance), Salesforce, etc. (Backend as a service). However, the applications developed by this approach need internet to be used, and often a high-speed network. Figure 10 presents the architecture of cloud-based approach (Choh et al., 2013).

4.4.5 Merged approach

As it is shown in another research paper (Ettifouri et al., 2017), this approach is based on combining two or more cross-platforms approaches as it shown in Figure 11, in order to benefit from the advantages and the strengths of each one of them, but it requires a lot of efforts from developers due to the use of multiple solutions according to the approaches chosen.

Figure 10 Cloud based architecture (see online version for colours)

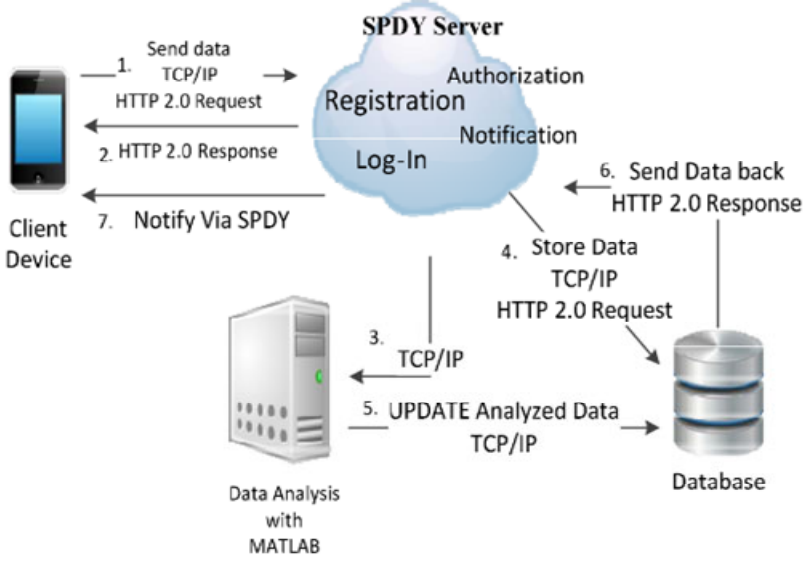
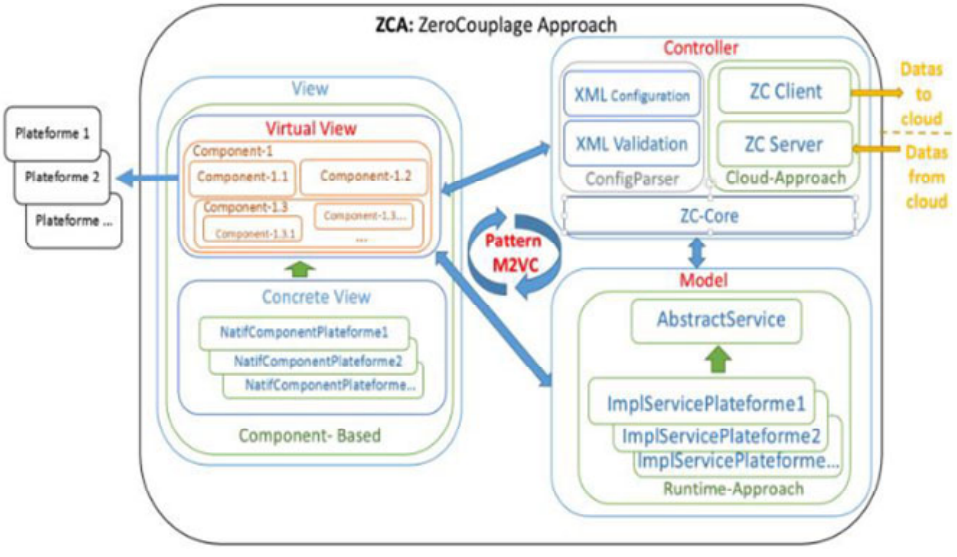


Figure 11 ZCA approach as an example of merged cross-platform approach (see online version for colours)

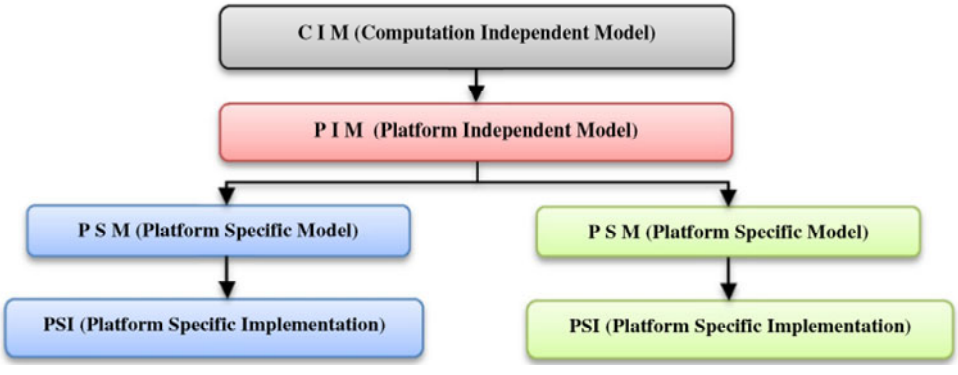


4.4.6 MDA approach (model driven architecture)

Modelling approaches generally are based on abstract models that describe the application functionalities using a textual and/or graphical concrete syntax (Biørn-Hansen, 2018).

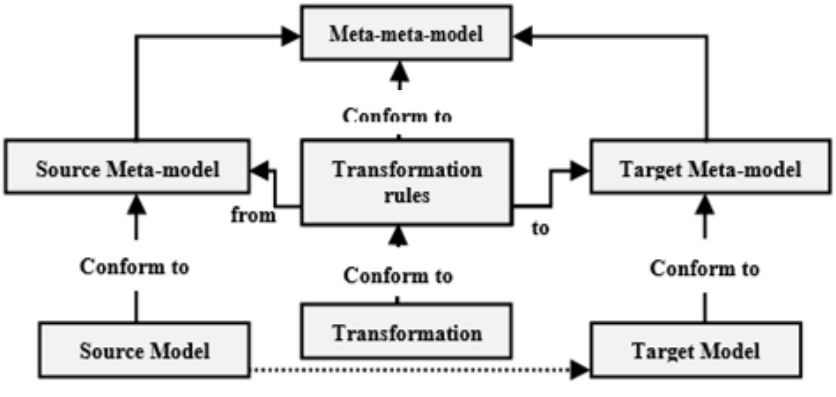
MDA approach is the most used modelling approach. It separates the business and the technical aspects of the application. Every aspect is represented by a set of models. MDA approach is based on three main model levels: computational independent model (CIM) which helps to present the system's functionalities. Platform independent model (PIM) which models the application design. Platform specific model (PSM) which concerns the application technical implementations on each platform (Charkaoui and Adraoui, 2014), and it leads to generate code for each platform, as illustrated in Figure 12 (Ebone et al., 2018).

Figure 12 MDA process (see online version for colours)



According to the MDA, a CIM will be transformed to PIM, which will be transformed to PSM, in order to generate final code. These transformations are applied to a source model which is conform to a source meta-model in order to generate a target model conforming to a target meta-model, using the QVT language (Benouda et al., 2017), as shown in Figure 13 (Lachgar and Abdali, 2014).

Figure 13 MDA transformation process (see online version for colours)



The main strength of this approach is based on models, which facilitate the task for developers who can focus on functional aspects instead of technical implementations. In other words, models take more attention in the development cycle. Until present, all the works have focused generally on simple user interface code without giving attention to some other aspects like sensors, etc.

4.4.7 Cross-platform approaches comparison

Compilation, interpretation, and component-based approaches focus globally on code, and from that code written in a particular programming language, we can generate other codes for each platform. However, the MDA approach has another strategy focusing only on models that represent a mandatory task before starting development of each application.

Every approach of the three first ones lack something related to code, for example: mapping between two codes, focusing on common elements and functions in existing platforms, having performance degradation due the use of VM or additional layer, etc. However, the MDA approach does not matter to these technical implementation problems, which constitute the main advantage of this approach.

5 Approach decision metrics

When evaluating the mobile apps development approaches, we often come across a comparison of the kind ‘hybrid vs. native’ or ‘low code vs. full code’. However, a project manager has many more constraints than that. Therefore, choosing the right approach depends on many more criteria than a pacific checklist.

We propose in our work a decision framework that can help anyone confronted with this situation.

5.1 Related works

In this article, tried to answer that problematic by setting a checklist to make the right choice (choose the right mobile development approach).

They propose to answer the following questions:

- Skilled team availability. The approach must be based on your team’s skill on the first place or hiring developers nearby. Choosing an appealing technology but not having the human resources can be a huge obstacle for the project success.
- Platform choice: Walled garden or open ecosystem. The term walled garden is used wisely. It referred to the low-code mobile apps development platforms such as Mendix (low-code mobile apps development platforms mendix) or Kony (low-code mobile apps development platforms kony). Everything is made available to the developers, from the front-end development with a library of components to the data management. They usually offer some third-party services such as Google Maps API. The inconvenient of those walled gardens is that you are limited to what they propose as development tools. You will never have full control of your app. Open ecosystems on the other hand are a good solution for those who do not want to be

limited by the platform. In fact, they offer you an SDK that can be used without any boundaries.

- Long-term partner. Choosing an approach will tie you to the development platform for a long time. The reason is the support needed in case of platform bugs occurrence.
- Design and UX consistency across teams and projects. In some contexts, there is a need of having shared UI library across all targeted platforms (mobile, desktop …). The origin of this problematic is because enterprises want to keep a unique design specification. Therefore, the solution is to choose an approach that is compatible with many frameworks.
- Ecosystem evolution. At this point, we need to shed light on the changing trend over the years. The project manager decision must be based on its maturity, then choose the right approach for it.

This checklist can help the developer to know more about his project, but the authors in Choose the right mobile development approach did not have an objective point of view because they were promoting for Ionic framework.

In the other hand, the authors in Decision framework of Anglin, and Todd proposed their own decision framework where they answered almost the same questions. However, we think that Todd's framework is not complete because it lacks some approaches such as: the cloud-based, merged, and cross-platform approaches.

5.2 Decision framework

To set a correct framework, we propose the following questions about the app:

- Could it target all platforms?
- How much is it easier to develop the app?
- How much is it easier to reuse UI code?
- Does it have a native look and feel?
- Does it have access to all mobile features?
- Does it have high performance?
- Can it be accessed in offline mode?
- Will it be published as free app or under a license?
- Will it be available on different platform's stores?
- How much is it easier to support it after publishing?
- How much is it easier to install updates?
- Does it have a high battery consumption?
- Does it have to implement an advanced security standard?
- How much it can reduce the project cost?

In the following Table 2, we give a rate to each question/approach in order to help the project manager to take his decision, depending on some criteria as question mentioned above. The rate will take 3 values: not good, middling, and very good.

- –1 : Not good
- 0 : Middling
- +1 : Very good.

In the Table 2, we consider the following:

- Native approach: N
- Web approach: W
- Hybrid approach: H
- Cross-platform approaches:
  a Compilation: C
  b Component-based: CB
  c Interpretation: I
  d Cloud-based: CL
  e Merged: M
  f MDA: model driven architecture.

According to Table 2, we conclude that a mobile application project depends on many criteria; in a general case; not only time and cost, but so many other criteria which are mentioned in the same Table 2. If there is no problem concerning cost and time, the native approach remains the best in terms of application look and feel, access to all native features even the newest ones, high performances, speed, etc. but still the maintenance and updates problems which will be multiplied by the number of targeted platforms each time. Otherwise, if the project manager decision depends only on cost and time, his decision will be oriented directly to web, hybrid, or cross-platforms applications.

To choose between these three approaches, the project manager will focus on the rest of the criteria. If there is no need to install the application in the device, nether has it access to native features, so it's rather to choose the web approach or cloud-based approach, taking into consideration that the targeted users must often have a good connection speed. Then, if it concerns a simple application that must be installed in the device but does not need a high performance in terms of speed especially, neither a native look and feel, such as payment, news applications, etc. it is rather to choose hybrid approach or interpreted approach.

When the decision goes towards a mobile application with the maximum respect of important criteria such as native look and feel, access to almost all native features, with the ease of the development, MDA approach remains the best solution in this case, since it depends only on models instead of technical implementation which needs a lot of effort for learning some new technologies which is the case of component-based or merged approaches. Also, it remains better than the compilation approaches since it could generate code for even the newest mobile UI elements or those that do not exist in all mobile platforms which is not the case of compilation approach.

Table 2 Weight attribution and approaches result to each question

Question	*Weight*	*Native*	*Web*	*Hybrid*	*Cross-platform*					
					C	*CB*	*I*	*CL*	*M*	*MDA*
Could it target all platforms?	X	+1	+1	+1	+1	+1	+1	+1	+1	+1
How much is it easier to develop the app?	X	–1	+1	+1	+1	–1	+1	+1	–1	+1
How much is it easier to reuse UI code?	X	–1	+1	+1	+1	+1	+1	+1	+1	+1
Does it have native look and feel?	X	+1	0	0	0	0	0	0	0	+1
Does it have access to all mobile features?	X	+1	–1	0	0	0	0	0	0	+1
Does it have high performance?	X	+1	0	0	0	–1	–1	0	0	0
Can it be accessed in offline mode?	X	+1	–1	+1	+1	+1	+1	–1	+1	+1
Will it be published as free app or under a license?	X	+1	–1	+1	+1	+1	+1	+1	+1	+1
Will it be available on different platform's stores?	X	+1	–1	+1	+1	+1	+1	+1	+1	+1
How much is it easier to support after publishing?	X	–1	+1	+1	–1	–1	–1	–1	–1	–1
How much is it easier to install updates?	X	–1	+1	0	0	0	0	0	0	0
Does it have a low battery consumption?	X	+1	+1	–1	+1	–1	–1	+1	0	+1
How much is easier to implement and advanced security standard?	X	+1	–1	+1	+1	+1	+1	+1	+1	+1
How much it can reduce the project cost?	X	–1	+1	+1	+1	+1	+1	+1	+1	+1
Score	$\text{Score} = \sum (X * Rate)$									

5.3 Simulation

Supposing that we would like to develop an absence management mobile application for a high school. Taking into account, the diversity of professors' phones, this application must run on different platform presenting the advantage of accessing mobile native features such as camera in order to scan some papers for example and others for special future needs. Also, it can be found in different platform's providers or stores, but the school have a limited budget which constitutes the major constraint. This mobile application should have some high performances especially in term of speed in order to facilitate the use for professors and avoid wasting their time. This application could run in offline mode in case the professors couldn't access to internet, and once connected the

data, can be transferred to school database. Finally, this application should have a low battery consumption in order to encourage professors to use it.

Table 3 Weight attribution and approaches result of the current case study

Question	*Weight*	*Native*	*Web*	*Hybrid*	*Cross-platform*					
					C	*CB*	*I*	*CL*	*M*	*MDA*
Could it target all platforms?	3	+1	+1	+1	+1	+1	+1	+1	+1	+1
Does it have high performance?	1	+1	0	0	0	–1	–1	0	0	0
Does it have access to all mobile features?	2	+1	–1	0	0	0	0	0	0	+1
Can it be accessed in offline mode?	3	+1	–1	+1	+1	+1	+1	–1	+1	+1
Will it be available on different platform's stores?	3	+1	–1	+1	+1	+1	+1	+1	+1	+1
Does it have a low battery consumption?	1	+1	+1	–1	+1	–1	–1	+1	0	+1
How much it can reduce the project cost?	3	–1	+1	+1	+1	+1	+1	+1	+1	+1
Score = $\sum(\text{Weight} * \text{Rate})$	10	–1	11	13	10	10	7	12	15	

In Table 3, we sum up all the constraint with a weight of each constraint that was communicated by the school, and we will use the framework method to choose the adequate approach for development process, taking into consideration the respect of all school constraints.

Applying the framework to the school specifications, we conclude that the best approach to use respecting all the school constraints is the MDA approach with a score of 15.

We suppose that this simulation is done by the school project manager, the choice of the enterprise that will take in charge the application development responsibility; could be the same or different maybe.

6 Conclusions

This paper presents a framework that helps to choose the best development's approach for a mobile project in each context. Each one of the approaches mentioned in this paper has shown its strength in serving the cross-platform purpose in terms of reducing cost and time, developing once, and deploying in all existing platforms. However, there is always a lack related to code and technical implementation.

This framework is based on several questions. The framework's result is influenced by the project's constraints presented as questions. Therefore, the framework returns a final decision for each mobile project context. However, when there are no constraints either in term of cost or human resources, the native approach remains the best solution to opt for because of its advantages in terms of quality, performance, ergonomics, look and feel, and access to native features.

Our works will focus on the MDA approach, which has more interest in models and functional aspect. Several research have been done on the same stream of works, many solutions based on MDA approaches have been proposed, but every solution lacks something such as: meta-model for remote service generation, meta-model for the transitions between screens, meta-model including all sensors even the recent ones such as fingerprint and face ID, applying MDA process for the future updates, and the big challenge still in having access to the entirely device features (Lachgar, 2017).

In our future works, we intend to propose an MDA approach's-based solution to generate mobile applications using a textual or graphical modelling language, an approach that respect mostly the constraints mentioned above.

In a first time, we will continue in the streams of works as Benouda et al. (2017), Lachgar and Abdali (2014), Sabraoui et al. (2012), Benouda et al. (2016) and Salma et al. (2018). We will propose a model to generate UI code even for the complex ones, using basic modelling languages such as UML to make our solution easier to be used by a large number of mobile developers who are mostly familiarised with UML, and enjoying its graphical advantage that are mostly understandable for non-developers. This advantage can facilitate needs communication between developers and the clients. Then, we will evolve our model by enriching it with element concerning transition between screens, sub-metamodel for sensors call, and accessing to mostly all device features.

References

Andrés, B.F. and Pérez, M. (2017) 'Transpiler-based architecture for multi-platform web applications', in *2017 IEEE: Second Ecuador Technical Chapters Meeting*, ETCM, New Jersey, USA, pp.1–6.

Backend as a service [online] https://www.g2.com/categories/mobile-backend-as-a-service-mbaas. (accessed 5 February 2021).

Benouda, H., Azizi, M., Esbai, R. and Moussaoui, M. (2016) 'MDA approach to automate code generation for mobile applications', in *Mobile and Wireless Technologies*, Springer, Singapore, pp.241–250.

Benouda, H., Azizi, M., Moussaoui, M. and Esbai, R. (2017) 'Automatic code generation within MDA approach for cross-platform mobiles apps', in *2017: First International Conference on Embedded and Distributed Systems*, EDiS, Oran, Alger, pp.1–5.

Biørn-Hansen, A., Grønli, T.M. and Ghinea, G. (2018) 'A survey and taxonomy of core concepts and research challenges in cross-platform mobile development', *ACM Computing Surveys*, Vol. 51, No. 5, pp.1–34.

Bosnic, S., Papp, I. and Novak, S. (2016) 'The development of hybrid mobile applications with Apache Cordova', in *2016: 24th Telecommunications Forum*, TELFOR, Belgrade, Serbia, pp.1–4.

Charkaoui, S. and Adraoui, Z. (2014) 'Cross-platform mobile development approaches', in *2014: Third IEEE International Colloquium in Information Science and Technology*, CIST, Tetuan-Chefchaouen, Morocco, pp.188–191.

Choh, Y., Song, K., Bai, Y. and Levy, K. (2013) 'Design and implementation of a cloud-based cross-platform mobile health system with HTTP 2.0', in *2013: IEEE 33rd International Conference on Distributed Computing Systems Workshops*, ICDCSW, Philadelphia, Pennsylvania, USA, pp.392–397.

Choose the right mobile development approach [online] https://ionicframework.com/resources/articles/ how-to-pick-the-right-mobile-development-approach (accessed 2 January 2021).

Dalmasso, I., Datta, S.K., Bonnet, C. and Nikaein, N. (2013) 'Survey, comparison and evaluation of cross platform mobile application development tools', in *2013: 9th International Wireless Communications and Mobile Computing Conference*, IWCMC, Sardinia, Italy, pp.323–328.

Decision framework of Anglin, and Todd [online] https://www.telerik.com/docs/default-source/whitepapers/choose-right-approach-mobile-appdevelopmentbb581d10116543e79a9febdb187fd0a3.pdf?sfvrsn=0 (accessed 5 March 2021).

Delía, L., Galdamez, N., Corbalan, L., Pesado, P. and Thomas, P. (2017) 'Approaches to mobile application development: Comparative performance analysis', in *2017 Computing Conference*, London, UK, pp.652–659.

Divya, K. and Kumar, V.K. (2016) 'Comparative analysis of smart phone operating systems android, apple ios and windows', *International Journal of Scientific Engineering and Applied Science*, Vol. 2, No. 2, pp.432–439.

Ebone, A., Tan, Y. and Jia, X. (2018) 'A performance evaluation of cross-platform mobile application development approaches', in *2018: IEEE/ACM 5th International Conference on Mobile Software Engineering and Systems*, MOBILESoft, Gothenburg Sweden, pp.92–93.

El-Kassas, W.S., Abdullah, B.A., Yousef, A.H. and Wahba, A.M. (2017) 'Taxonomy of cross-platform mobile applications development approaches', *Ain Shams Engineering Journal*, Vol. 8, No. 2, pp.163–190.

Ettifouri, E.H., Rhouati, A., Berrich, J. and Bouchentouf, T. (2017) 'Toward a merged approach for cross-platform applications (web, mobile and desktop)', in *Proceedings of the 2017: International Conference on Smart Digital Environment*, Rabat, Morocco, pp.207–213.

Jobe, W. (2013) 'Native Apps vs. Mobile Web Apps', *International Journal of Interactive Mobile Technologies*, Vol. 7, No. 4, pp.27–32.

Khachouch, M.K., Korchi, A., Lakhrissi, Y. and Moumen, A. (2020) 'Framework choice criteria for mobile application development', in *2020: International Conference on Electrical, Communication, and Computer Engineering*, ICECCE, Istanbul, Turkey, pp.1–5.

Korchi, A., Khachouch, M.K., Lakhrissi, Y. and Moumen, A. (2020) 'Classification of existing mobile cross-platform approaches', in *2020: International Conference on Electrical, Communication, and Computer Engineering*, ICECCE, Istanbul, Turkey, pp.1–5.

Kramer, D., Clark, T. and Oussena, S. (2010) 'MobDSL: a domain specific language for multiple mobile platform deployment', in *2010 IEEE: International Conference on Networked Embedded Systems for Enterprise Applications*, Suzhou, China, pp.1–7.

Lachgar, M. (2017) *Approche MDA Pour Automatiser La Génération De Code Natif Pour Les Applications Mobiles Multiplateformes*, Published PhD Thesis, CADI AYYAD University, Marrakech, Morocco.

Lachgar, M. and Abdali, A. (2014) 'Generating Android graphical user interfaces using an MDA approach', in *2014: Third IEEE International Colloquium in Information Science and Technology,* CIST, Tetuan-Chefchaouen, Morocco, pp.80–85.

Lachgar, M. and Abdali, A. (2017) 'Decision framework for mobile development methods', *Int. J. Adv. Comput. Sci. Appl.* (IJACSA), Vol. 8, No. 2, pp.110–118.

Latif, M., Lakhrissi, Y. and Es-Sbai, N. (2016) 'Cross platform approach for mobile application development: a survey', in *2016 International Conference on Information Technology for Organizations Development*, IT4OD, Fez, Morocco, pp.1–5.

Latif, M., Lakhrissi, Y. and Es-Sbai, N. (2017) 'Review of mobile cross platform and research orientations', in *2017 International Conference on Wireless Technologies, Embedded and Intelligent Systems*, WITS, Fez, Morocco, pp.1–4.

Low-code mobile apps development platforms kony [online] https://www.kony.com/ (accessed 2 March 2020).

Low-code mobile apps development platforms mendix [online] https://www.mendix.com/ (accessed 5 May 2020).

Meirelles, P., Aguiar, C.S., Assis, F., Siqueira, R. and Goldman, A. (2019) 'A students' perspective of native and cross-platform approaches for mobile application development', in *International Conference on Computational Science and Its Applications*, ICCSA Petersburg, Russia, pp.586–601.

Mobile app revenue [online] https://www.statista.com/statistics/269025/ worldwide-mobile-app-revenue-forecast/. (accessed 2 March 2021).

Novac, O.C., Novac, M., Gordan, C., Berczes, T. and Bujdosó, G. (2017) 'Comparative study of Google Android, Apple iOS and Microsoft Windows phone mobile operating systems', in *2017: 14th International Conference on Engineering of Modern Electric Systems*, EMES, Oradea, Romania, pp.154–159.

Number of applications availabre in leading app stores [online] https://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/ (accessed 2 February 2021).

Okediran, O.O., Arulogun, O.T., Ganiyu, R.A. et al. (2014) 'Mobile operating systems and application development platforms: a survey', *International Journal of Advanced Networking and Applications*, Vol. 6, No. 1, pp.2195–2201.

OS smartphone: Key values [online] https://www.zdnet.fr/actualites/chiffres-cles-les- os-pour-smartphones-39790245.htm (accessed 21 February 2021).

Palmieri, M., Singh, I. and Cicchetti, A. (2012) 'Comparison of cross-platform mobile development tools', in *2012: 16th International Conference on Intelligence in Next Generation Networks*, ICIN, Berlin, Germany, pp.179–186.

Pinto, C.M. and Coutinho, C. (2018) 'From native to cross-platform hybrid development', in *2018 International Conference on Intelligent Systems*, IS, Funchal, Madeira, Portugal, pp.669–676.

Raj, C.R. and Tolety, S.B. (2012) 'A study on approaches to build cross-platform mobile applications and criteria to select appropriate approach', in *2012 Annual IEEE India Conference*, INDICON, Kochi, India, pp.625–629.

Sabraoui, A., El Koutbi, M. and Khriss, I. (2012) 'Gui code generation for android applications using a mda approach', in *2012: IEEE International Conference on Complex Systems*, ICCS, Agadir, Morocco, pp.1–6.

Salma, C., Abdelaziz, M. and Issam, A. (2018) 'Cross-platform mobile development framework based on MDA approach', *International Journal of Technology Diffusion*, Vol. 9, No. 1, pp.45–59.

Yang, Y., Zhang, Y., Xia, P., Li, B. and Ren, Z. (2017), 'Mobile terminal development plan of cross-platform mobile application service platform based on ionic and Cordova', in *2017 : International Conference on Industrial Informatics-Computing Technology, Intelligent Technology, Industrial Information Integration*, ICIICII, Wuhan, China, pp.100–103.