Decision Framework for Mobile Development Methods

LACHGAR Mohamed
Laboratory of Applied Mathematics and Computer Science (LAMAI), Faculty of Science and Technology (FSTG),
Cadi Ayyad University
Marrakech, Morocco

ABDALI Abdelmounaïm
Laboratory of Applied Mathematics and Computer Science (LAMAI), Faculty of Science and Technology (FSTG),
Cadi Ayyad University
Marrakech, Morocco

***Abstract*—Recently, the mobile applications have emerged with the uprising smartphone trend. Now-a-days, a huge number of mobile operating systems require more developments, in order to achieve that, Open source cross-platform mobile frameworks came up, in order to allow importing the same code on various operating systems. In this paper, the focus is made on commonly used mobile development methods, and a process that selects the most suitable solution for a particular need is proposed. Eventually, a new framework that helps to choose the appropriate approach and tool respectively is suggested, according to a convenient survey based on binary questions, in addition to certain criteria.**

***Keywords*—Mobile development approaches; Mobile development tools; Cross-platform mobile; Mobile OS**

I. Introduction

Mobile devices, applications and associated services are being radically reshaped by user's behavior and corporate organizations as well, either business models, or business strategies and also the way employees work.

Since the release of the first iPhone in 2007, smart mobile devices occupied an important role in the world economy, so we talk more often about digital economy.

Worldwide mobile phones sales reached nearly 478 million units during the third quarter of 2015, so an increase of 3.7 percent compared to the same period in 2014. The figures and the trends presented in the following study confirm these facts [1].

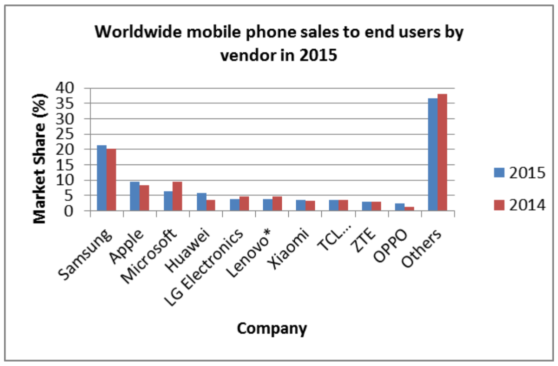

Fig. 1. Worldwide mobile phone sales to end users by vendor in 2015

This evolution is due to the growth the smartphone market, as those consumers abandon more and more "dumb" or "less smart" phones [1]. The following figure shows the evolution of smartphone sales compared to classic mobiles sales.

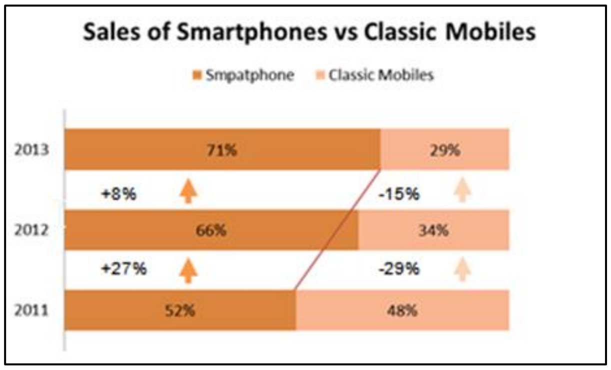

Fig. 2. Sales of smartphones vs classic mobiles [2]

Between 2011 and 2013, the share of smartphone sales increased by 37%. Now-a-days, about 71% of mobiles in markets are smartphones.

The market of tablets and smartphones is dominated by Android [1]. The choice of Android is justified by its constantly innovative technology, open and less expensive compared to iOS.

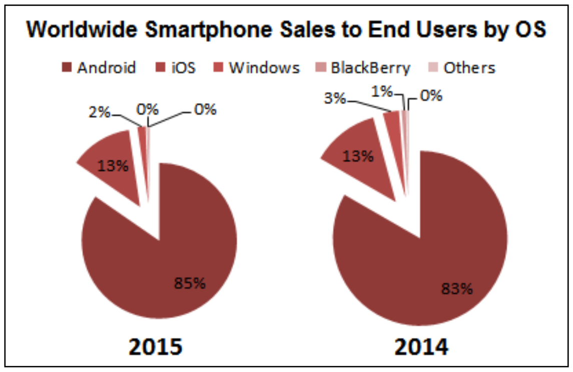

Fig. 3. Worldwide smartphone sales to end users by OS [1]

Each platform indeed requires different development tools. If we want to deploy an application on different platforms, it seems necessary to consume as much time as the sum of the time needed for each application; But there are some solutions to not allow the development of the application once, and then deploy it on other platforms. The aim of this article is, on one hand, to present these solutions and then to make a comparison between them, each one has its advantages and drawbacks; on the other hand, to provide afterwards an ideal approach for deciding which solution should be adopted for a given case.

This paper is structured as follows: the first section presents the mobile development methods, followed by a comparative study of mobile development approaches. The second section shows an ideal approach for deciding which solution should be adopted for a given case. The last section concludes the paper and presents some future works and perspectives.

II. Related Works

Several studies have been carried out on mobile development methods, in which researchers presented the advantages and drawbacks of each approach. In [3] the authors presented a comparative study of multi-platform mobile development tools (PhoneGap, Titanium, Sencha Touch and jQuery Mobile). While the paper [4], has shown the advantages and drawbacks of various methods of mobile development and proposed technologies for each case, based on qualitative properties. However, Charland and Leroux [5] present an in-depth comparison of Native apps and Web apps development. Heitkotter et al. [6] present a comparative study between some cross-platform mobile tools based on several qualitative factors such as licensing costs, look-and-feel, supported platforms, development environments, maintainability and scalability. In this approach, the cross-platform perspective is not taken into account.

Veldhuis [8] present a comparative analysis about the performance of various mobile development tools, based on a simple numerical calculation.

In [9] the authors formulate a method to evaluate and select the best cross-platform development tools for a developer and also evaluate cross platform tools using time, technology, maturity, and cost aspects of mobile apps development. In contrast, this work is focused on the cross-platform development tools and doesn't present a process to assess the appropriate development method to adopt (native, hybrid or web).

In this paper we presented the architecture and features of each method, and an approach that could be adopted to choose an appropriate method and tool is proposed, in order to develop a mobile application.

Our framework focuses on the improvement of decision making in the mobile applications domain, taking into account several qualitative factors such as development rate, documentation, look and feel, popularity, learning curve and graphical tool for GUI. The mentioned framework can be divided into two stages; the first one allows deducing the mobile development method while the second one allows selecting the right tool for each method whose precision exceeds 50%.

III. Mobile Development Methods

The cross platform mobile applications are widely meant to provide mobile apps developers with means for writing once, and deploying everywhere. Currently, the market is full of dizzying array of cross-platform development tools [4].

Several studies on approaches to build cross-platform mobile applications are produced [4], [9], [12], [13]. Conclusively, a classification of these approaches into three categories is made:

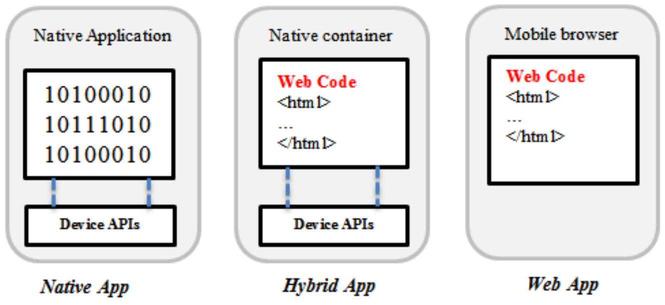

Fig. 4. Mobile development method

These types will be explained in the following sub-sections.

A. Native Approach

The Native applications have the highest performance, native look and feel, has full access to the device capabilities, they use the most updated hardware resources, in order to improve performance. The applications are built in languages that the platform supports, as a consequence it has access to IDEs, which provides the best tools for development, as well as a fast debugging of the project. Android apps can be built in Java on Android Studio, and iOS apps can be built in objective C on XCode, which have all the tools either to debug, or to design the interfaces, and then check the performance using instruments. Yet, the development of the native App needs initial time to learn the languages and tools provided by the platform-specific vendor, then develops the App. Also, the App will run on only one specific-platform [4], [9], [10]. The figure below shows native apps architecture:

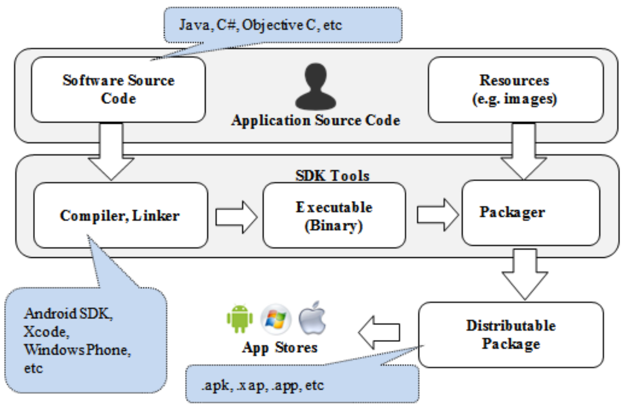

Fig. 5. Native app development

B. Web Approach

The mobile web Apps are developed using standard web technologies—typically HTML5, JavaScript and CSS. These apps are easy to develop, although cannot use device-specific hardware features such as camera or GPS sensor, and the lack look and feel of the native App [11].

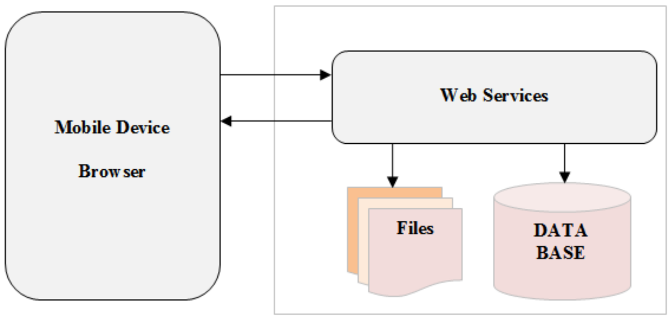

Fig. 6. Logical architecture of a mobile web application [4]

C. Hybrid Approach

The mobile hybrid apps combine between the web App and the native App. This type does not perform as well as the other programs that are based on native languages. Even though they are packaged natively, they are not native applications, they are executed on the platforms web engine, Webkit in case of Android and iOS, which is another layer between the user and the application, and so the performance can't match with the native apps [3], [12].

The below diagram depicts the high level of hybrid mobile application architecture:

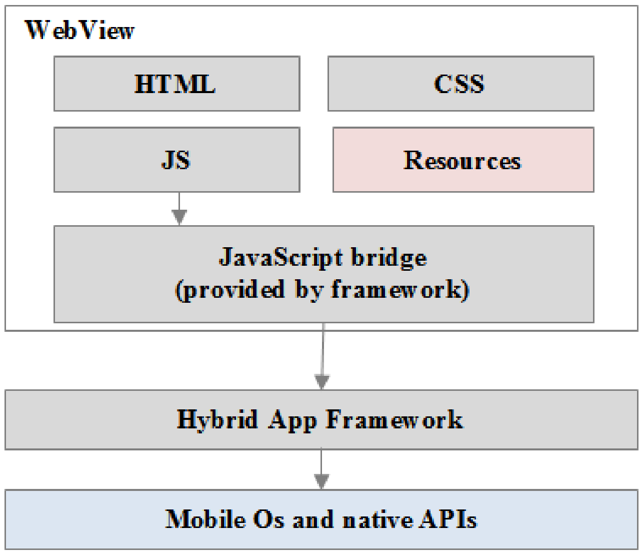

Fig. 7. Logical architecture of a typical hybrid application

D. A comparison of the three approaches

A comparison of the three approaches is structured in the following table.

TABLE I. MOBILE APPS DEVELOPMENT APPROACHES COMPARISON

	Native Approach	Hybrid Approach	Web Approach
Device Access	Full	Full	Partial
Speed	Very fast	Native speed	Fast
App Development cost	Expensive	Reasonable	Reasonable
AppStore	Yes	Yes	No
Approval Process	Mandatory	Low overhead	None
Quality of UX	Excellent	Not as good as native apps	Very good
Quality of apps	High	Medium to low	Medium
Security	High	Not good	Depends on browser security
Potential users	Limited to a particular mobile platform	Large – as it reaches to users of different platforms	Maximum including smartphones, tablets and other feature phones
Access device-specific features	High	Medium	Low
Development language	Native only	Native and web or web only	Web only
Skills/tools needed for cross-platform apps	Objective-C, Java, C, C++, C#, VB.net	HTML, CSS, JavaScript, Mobile development framework (like PhoneGap)	HTML, CSS, JavaScript

According to this study, native application turned out to be more improved, in terms of performance compared to other mobile application types (i.e., web and hybrid). Native applications are developed using a platform specific API compiled to run on the platform rather than an interpreted language code, such as, JavaScript. But the problem is that these native apps are more expensive to implement, limited to a particular mobile platform, require a collection of knowledge and languages to be realized.

The figure below shows the trend for native to cross platform development cost and time factors.

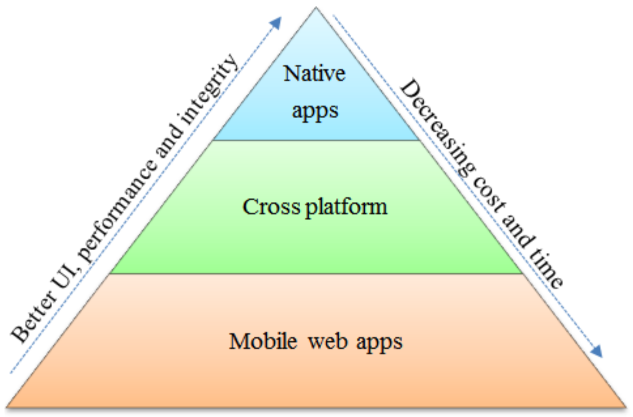

Fig. 8. Native vs. Cross platform development

IV. Decision Framework for Adopting the Appropriate Development Method and Tools

We have shown that the three solutions have advantages and inconveniences. The question now that arises is: which are the approaches that can be adopted to develop a mobile cross-platform application? And what tool can be used to implement the solution?

To answer these questions, a tool to provide answers based on the nature of the application to develop is proposed. The architecture of this tool is presented below (See Figure 9 for more details).

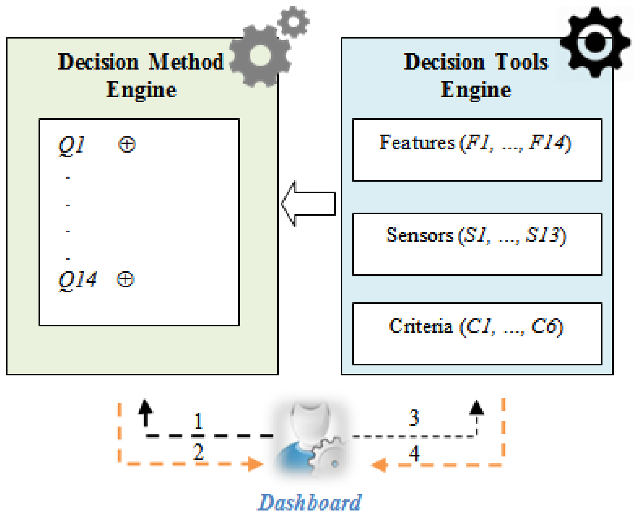

Fig. 9. Framework architecture

In more detail, this architecture consists of four key steps:

1) *The first step :* consists of filling a survey, then sending the answers to the decision engine.

2) *In the second step :* the decision engine analyzes the responses and transmits the appropriate mobile development method to the customer, and also determines the percentage of completion of each method.

3) *In the third step :* according to the received method, the customer must complete a survey, then forward it to the decision tool engine.

4) *In the last step :* the decision tool engine analyzes the responses and sends the right tool to be used in the implementation of the solution to the customer. In the hybrid case the tools are classified according to the features desired in the application to develop.

The next subsections show how each block is implemented.

A. Decision Method Engine

For this, we propose a set of questions in order to single out the correct approach to develop a very specific mobile application.

Q 1 : Should it be published on the main AppStore?
Q 2 : Does it operate in offline mode?
Q 3 : Do you want to sell it?
Q 4 : Is it a simple application?
Q 5 : Will it be frequently used by the user?
Q 6 : Is there an immediate need to deliver the app to the market?
Q 7 : Do you have separate budget for developers in each OS?
Q 8 : Do you need a lot of native features in the Mobile App?
Q 9 : Is app security a high priority?
Q 10 : Should it be very fluid?
Q 11 : Do you want a lot of animations?
Q 12 : Are we building application that needs a lot of algorithmic computation?
Q 13 : Do you want to be always up to date with the latest versions of OS?
Q 14 : Do you want to have the best user experience?

The table below gives the answers to these questions for each mobile development approach (native, hybrid and web).

TABLE II. Mobile Development Methods Decision Framework

	Native	**Hybrid**	**Web**
Q 1	✓	✓	✗
Q 2	✓	✓	✗
Q 3	✓	✓	✗
Q 4	✗	✗	✓
Q 5	✓	✓	✗
Q6	✗	✓	
Q 7	✓	✗	
Q 8	✓	✗	
Q9	✓	✗	
Q 10	✓	✗	
Q 11	✓	✗	
Q12	✓	✗	
Q 13	✓	✗	
Q 14	✓	✗	

In this perspective, we present the selection criterion established in a decision tree represented in the figure 10 below.

B. Decision Method Engine implementation

The decision tree shown in Figure above is used to determine the mobile development approach to be taken within a given situation. The decision method engine will also determine the percentage of completion of each method. To do this, we have adopted the following approach.

We have assigned, a decision factor, to each question, according to its importance. The selected intervals clarify these points:

- 8 : Very important
- 6 : Important

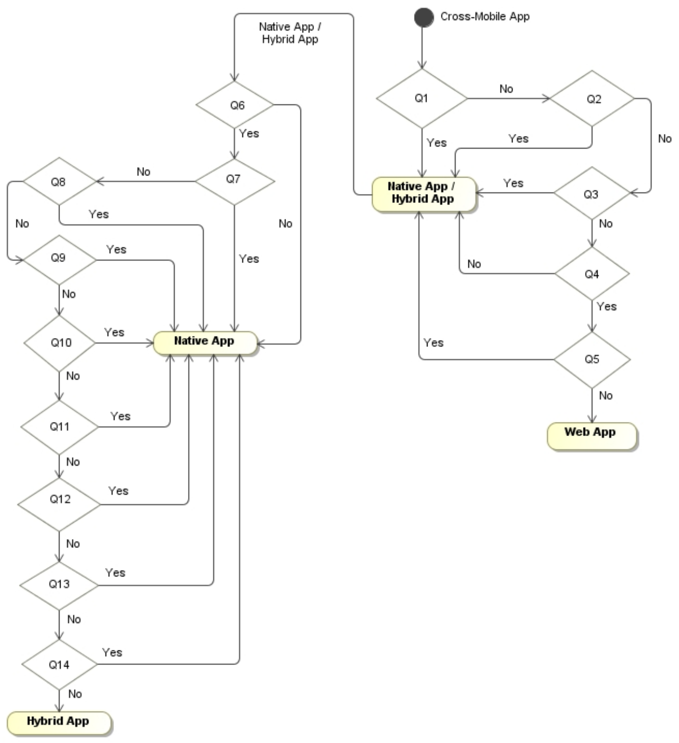

Fig. 10. Decision Tree for adopting the appropriate development method

- 4 : Not so important
- 2 : Not at all important

The chart below illustrates these assigned weights:

TABLE III. FACTORS ATTRIBUTED TO QUESTIONS

Question	Factor
Q1	8
Q2	4
Q3	6
Q4	2
Q5	2
Q6	6
Q7	8
Q8	6
Q9	6
Q10	4
Q11	4
Q12	6
Q13	6
Q14	6

The rationale for choosing various weights of factors is provided below:

- Question 1 is essential, to decide between web approach and the two other approaches. Question 7 is very important for choosing between native approach and the hybrid one, which explains the factor 8 as a decision factor.
- Question 2, actually is less important, especially with the web approach which allows saving data through offline mode according to the HTML 5 innovations.
- Question 3 depends on question 1; a mobile application for sale, must be published in the APPSTORE, thus, we have provided the decision factor 6.
- Question 4 not all important, a simple application can be developed even with all approaches.
- Question 5 not all important, a simple application developed with the web approach, without recourse to the native APIs, can also be used frequently by users.
- Question 6 is important, to decide between native and web approach. If the company has skilled human resources to develop the application within the deadlines set, will be interesting to adopt the native approach, which explains the factor 6 as a decision factor.
- Question 8 can make the difference between native and hybrid approaches. In order to implement an application, an access to several native APIs is required, so it is better to use native approach.
- Question 9 is important, if security is a priority, then it would be better to adopt native approach. Consequently, we assigned 6 as a decision factor for this question.
- Questions 10 and 11 are less important, according to the hybrid approach evolution that supports the implementation of some animations and fluidity depending on the JavaScript Framework evolution, therefore, for both questions, we have assigned the factor 4.
- Whenever the application requires a lot of algorithmic computation, then it is better to use native language for taking advantage of the methods already developed. That explains 6 as a decision factor for a question 12.
- Question 13 is important, if a mobile application has several native features it should benefit from the latest updates of the operating system, which explains the factor 6 as a decision factor.

The feedback is a strong point for this we assigned 6 as a decision factor for a question 14.

An extract of the used class diagram for implementing is shown below:

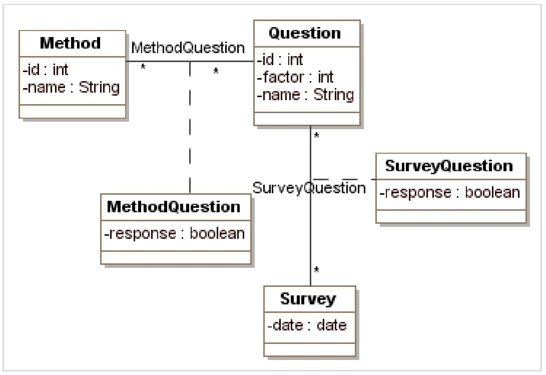

Fig. 11. Extract of class diagram of method engine

The precision is given by the following ratio:

$$\text{Precision (in \%)} = \frac{\sum \text{Factor}(Q_P)}{\sum \text{Factor}(Q_E)} * 100$$

Where:

$\sum \text{Factor}(Q_P)$: is the sum of the factors of the performed questions.

$\sum \text{Factor}(Q_E)$: is the sum of the factors of the expected questions.

With: (Q_P) are the performed questions and (Q_E) are the expected questions (according to TABLE II).

C. Decision Tools Engine

Once the development approach is selected, the next step will be to define the tools to use during the implementation phase. In order to achieve this, we evaluate the needs of the solution to develop towards some sensors and features available in the mobile phone.

The following features and sensors are integrated in many of the major smartphone devices:

TABLE IV. SMARTPHONE DEVICE STANDARD FEATURES

Code	Features	Definition
F1	Contacts	Does the solution supports CRUD functionality to access the contact list?
F2	Geolocation	Does the solution can be capable of using smartphone GPS?
F3	Ad hoc Wi-Fi	Does the solution capable of managing ad hoc Wi-Fi connections?
F4	Storage	Does the solution support CRUD functionality for Local Storage?
F5	SMS	Does the solution have an API to send SMS from the application?
F6	Telephony	Does the solution have an API to make calls from the application?
F7	Bluetooth	Does the solution supply an access to device Bluetooth?
F8	Audio (Recording)	Does the solution allow audio playback in the application?
F9	Audio (Reading)	Does the solution allow audio recording in the application?
F10	Camera (Take photo)	Does the solution allow taking pictures in the application?
F11	Camera (Video Recording)	Does the solution allow the recording of video in the application?
F12	Vibration	Does the solution allow making vibrate the device since the application?
F13	Multi – touch	Does the solution can be capable of capturing the "Gestures" or the "Multi-touch"?
F14	SOAP	Does the solution have an API to manage the SOAP protocol?
F15	Push Notification	Does the solution contain an API to manage "push notifications"?
F16	SQLite	Does the solution integrate the functionality Create, Read, Update, and Delete (CRUD) of SQLite?
F17	Network availability	Does the solution can be capable of checking the availability of the network?
F18	File System	Does the solution provide to access to the device's file system?
F19	Memory management	Does the solution allow to manually managing memory?

TABLE V. SMARTPHONE DEVICE STANDARD SENSORS

Code	Sensors	Definition
S1	Accelerometer	Does the solution allow to access to the accelerometer?
S2	Compass	Does the solution allow to access to the magnetometer or has it an API to create a compass?
S3	Orientation	Does the solution allow detecting the rotation of the device?
S4	Light sensor	Does the solution allow access to the light sensor?
S5	Gravity	Does the solution allow access to the gravity sensor?
S6	Pressure	Does the solution allow access to the pressure sensor?
S7	Gyroscope	Does the solution allow access to the gyroscope sensor?
S8	Proximity	Does the solution allow access to the proximity sensor?
S9	Temperature	Does the solution allow access to the temperature sensor?
S10	Ambient Temperature	Does the solution allow access to the ambient temperature sensor?
S11	Linear Accelerometer	Does the solution allow access to the linear accelerometer sensor?
S12	Magnetic Field	Does the solution allow access to the magnetic field sensor?
S13	Relative Humidity	Does the solution allow access to the relative humidity sensor?

Also, here are some criteria which may be useful in the selection process:

TABLE VI. SELECTION CRITERIA

Code	Criteria
C1	Development rate
C2	Documentation
C3	Look and feel
C4	Popularity
C5	Learning curve
C6	Graphical tool for GUI

The following sub-section describes and evaluates the mobile development tools, towards the different aspects identified above. These tools are classified in three categories: the platform specific development kit, the Cross-platform mobile development and the web tools.

D. Decision Tools Engine implementation

In the case of the web approach, the tools are defined namely HTML5, CSS and JavaScript.

In the native approach, according to the target platforms, tools to be used for each platform can be defined; therefore, the choice will be unique in this case.

In the case of the hybrid approach the decision tools engine must provide a score that will be calculated, based on the number of features and sensors required in the application that are supported by the tool.

Now, to choose the right tool for implementing a mobile software application, we have defined the following scale:

Rating API or Sensor needs:

- 4: Well Supported.
- 2: Supported.
- 0: Not support.

Development rate:

- 3: Very Fast.
- 2: Fast.
- 1: Medium.
- 0: Slow.

Documentation:

- 3 : Very Good
- 2: Good.
- 1: Fair.
- 0: Poor.

Look and Feel:

- 3: Very Good.
- 2: Good.
- 1: Fair.
- 0: Poor.

Popularity:

- 3: Very popular (Very High).
- 2: Popular (High).
- 1: Less popular (Medium).
- 0: Not popular.

Learning curve:

- 3: Very Fast.
- 2: Fast.
- 1: Medium.
- 0: Long.

Graphical tool for GUI:

- 2: Well supported.
- 1: Supported.
- 0: Not supported.

An extract of the used class diagram for implementing is shown in Figure 12 below:

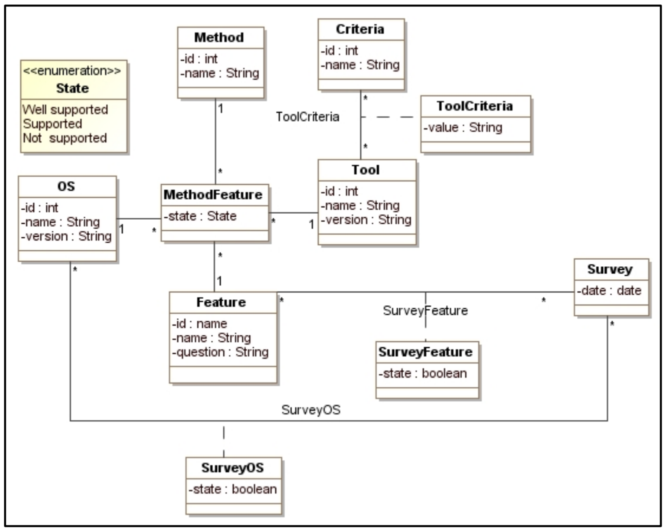

Fig. 12. Extract of class diagram tools engine

V. CASE-STUDY

A. Description:

The aim of this project is to develop a location-based app, this latter allows to locate the position of contacts in the phone book located within a given radius, using a Map, it also provides the ability to communicate with other people connected to the network with the same application, by exchanging text messages and media files (e.g. photo, video), and finally it gives the possibility to take pictures and transmit them via the application to other contacts.

B. Requirements:

- Available on Android and iOS.
- Access to the network.
- Notification Alert and Vibration.
- Access to Camera and video.
- Low costs development.
- Deployable on app stores.
- Access to media.
- Access to Smartphone GPS.
- Access to contacts list.
- Access to telephony.

C. Tools:

- F1 : PhoneGap + jQuery Mobile.
- F2 : PhoneGap + Sencha Touch.

- F3 : PhoneGap + Onsen IU.
- F4 : PhoneGap + Angular UI.
- F5 : PhoneGap + Ionic.
- F6 : Titanium Appcelerator.
- F7 : Xamarin.
- F8 : Flex + Air.

D. Result:

- Method decision:

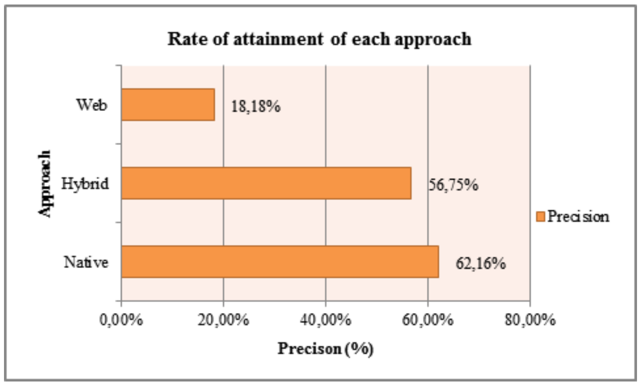

Fig. 13. Rate of attainment of each approach

62.16% of requirements need to adopt the native approach.

56.75% of requirements can be implemented with the hybrid approach.

18.18% of needs can be developed with the web approach.

- Tools decision:

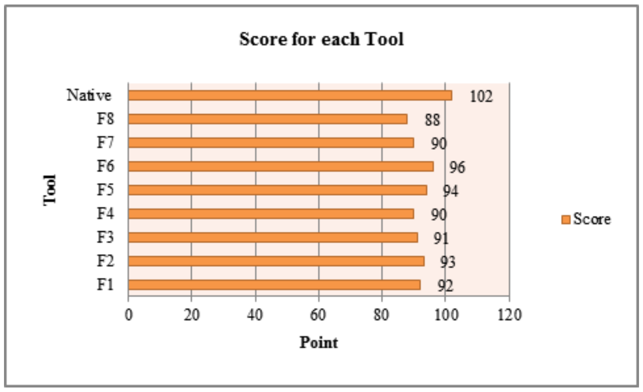

Fig. 14. Score for each tool

For this case-study, the platform specific development kits are among the best, Titanium Appcelerator in the middle followed by PhoneGap with Ionic framework and Sencha Touch, and Flex among the lowest-ranking.

VI. Conclusions and Future Works

This work, presents a framework allowing to select the best technology to use for the development of a specified mobile application in a given context. This framework consists of two main stages, the first one determines the mobile development method (native, hybrid or web) with a completion percentage called precision, based on a set of relevant questions, the second one determines the appropriate tool for the implementation based on a set of relevant criteria.

In an ideal world of technology, without time constraints and money, it would be obviously more interesting to move to a native solution. The result has advantages in terms of ergonomics, performance and integrity.

This study allowed us to understand in which case it is interesting to turn to the web and hybrid solutions. A timely simple and unconstrained performance gain has to be a hybrid or web approach.

Consequently, so as to remedy to native approach's shortcomings, we suggest setting up a solution based on the Model-driven Engineering, allowing developers to generate native applications from the UML diagrams or by using DSL [14], [15].

We are currently working on the development of solutions for reverse engineering, aiming to transform the hybrid code and the web one, into native code. Thus, it will use the native applications advantages and extend them with other native features, which aren't supported now-a-days in the hybrid and web methods.

References

[1] Gartner, "Gartner Says Smartphone Sales Surpassed One Billion Units in 2014", http://www.gartner.com/newsroom/id/2996817, March 3, 2015 (Accessed on December 3, 2015)

[2] Kerensen Consulting, "Evolution des usages Mobiles, prévision 2015".

[3] I. Dalmasso, S. Datta, C. Bonnet and N. Nikaein, "Survey, comparison and evaluation of cross platform mobile application development tools", Proceedings of the 9th International Wireless Communications and Mobile Computing Conference (IWCMC), IEEE Xplore press, Sardinia, pp. 323-328. 2013. DOI: 10.1109/IWCMC.2013.6583580.

[4] R. Raj and S. B. Tolety, "A study on approaches to build cross-platform mobile applications and criteria to select appropriate approach", India Conference (INDICON), IEEE Xplore press, Kochi, pp. 625-629. 2012. DOI:10.1109/INDCON.2012.6420693.

[5] A. Charland and B. Leroux, "Mobile application development: web vs. native", Communications of the ACM, vol. 54, no 5, pp. 49-53. 2011.

[6] H. Heitkötter, S. Hanschke and T. A. Majchrzak, "Evaluating cross-platform development approaches for mobile applications", Web information systems and technologies. Springer Berlin Heidelberg, pp : 120-138. 2012.

[7] L. Delía, N. Galdamez, L. C. Corbalán, P. J. Thomas and P. M. Pesado, "Un análisis comparativo de rendimiento en aplicaciones móviles multiplataforma", XXI Congreso Argentino de Ciencias de la Computación. 2015.

[8] M. M. O. Veldhuis, "Multi-Target User Interface design and generation using Model-Driven Engineering", Unpublished dissertation in partial fulfillment of the requirements for the degree of Master of Science in Computer Science and Master of Science in Human Media Interaction Faculty of Electrical Engineering, Mathematics and Computer Science University of Twente, the Netherlands, 2013.

[9] P. Smutny, "Mobile development tools and cross-platform solutions". Proceedings of the 13th International Conference on Carpathian Control (ICCC), IEEE Xplore Press, High Tatras, pp. 653-656, 2013, DOI:10.1109/CarpathianCC.2012.6228727.

[10] S. Xanthopoulos and S. Xinogalos, "A comparative analysis of crossplatform development approaches for mobile applications", Presented at the Proceedings of the 6th Balkan Conference in Informatics, Thessaloniki, Greece, 2013.

[11] L. Corral , A. Janes and T. Remencius, “Potential Advantages and Disadvantages of Multiplatfonn Development Frameworks-A Vision on Mobile Environments”, Procedia Computer Science, vol. 1 0, pp. 1202-1207, 2012.

[12] M. Palmier, I. Sing and A. Cicchetti, “Comparison of cross-platform mobile development tools”, Proceedings of the 16th International Conference on Intelligence in Next Generation Networks (ICIN), IEEE Xplore Press, Berlin, pp. 179-186, 2012, DOI:10.1109/ICIN.2012.6376023.

[13] N. Serrano, J. Hernantes and G. Gallardo, “Mobile Web Apps”, IEEE Software. pp: 22 – 27, 2013, DOI:10.1109/MS.2013.111.

[14] M. Lachgar and A. Abdali, “Generating Android graphical User Interfaces using an MDA approach”, Proceedings of the Third International Colloquium of Information Science and Technology (CIST), IEEE Xplore Press, Morocco, pp. 80-85, 2014, DOI:10.1109/CIST.2014.7016598.

[15] M. Lachgar and A. Abdali, Abdelmounaïm, “Modeling and generating native code for cross-platform mobile applications using DSL”, Intelligent Automation & Soft Computing, pp. 1-14, 2016.