Comparison of Cross-Platform Mobile Development Tools

Manuel Palmieri
Innovation, Design and Engineering
Mälardalen University
Västerås, Sweden
palmierimanuel@gmail.com

Inderjeet Singh
Innovation, Design and Engineering
Mälardalen University
Västerås, Sweden
inderjeetuiet@gmail.com

Antonio Cicchetti
Innovation, Design and Engineering
Mälardalen University
Västerås, Sweden
antonio.cicchetti@mdh.se

Abstract*—Mobiles are an integral part of daily life. With time, customers are expecting good and very versatile applications in less time. It is a big challenge to develop high performance mobile applications in this competitive market that would meet the expectation of customers. Mobile operating systems vendors are giving their best available resources for making applications in more convenient ways, although the development of new applications for each mobile operating system in short time is fairly a problem. Cross-platform mobile application development tools contribute in solving this problem largely. This paper presents a pragmatic comparison among four very popular cross platform tools, which are Rhodes, PhoneGap, DragonRad and MoSync. One of the main focuses of the comparison is to provide an overview on the availability of application programming interfaces, programming languages, supported mobile operating systems, licences, and integrated development environments. Furthermore, it also presents some critical points such as the factor of extensibility in tools and the effects that they may bring on market share. The comparison is aimed at supporting developers to make the right choice with respect to their needs/constraints.

Keywords: mobile application, cross-platform development tool, operating system, native api, smartphone.

I. Introduction

Nowadays, mobiles are more a necessity than a luxury. Besides making calls there are many other features which are gaining popularity like Camera, Music, Global Positioning System (GPS), Accelerometer, etc. These kinds of built-in features are provided by all major available mobile Operating Systems (OS's), such as Android, BlackBerry, iPhone Operating System (iOS), Symbian, Windows Mobile/Phone. All these mentioned mobile OS's are very popular in the market because of their uniqueness, for example Android is based on Java and freely available, iOS provides innovative features and quality, and BlackBerry is the most known in the corporate world. Furthermore, there are some other factors such as licenses, libraries of support, native features, etc., which are affecting their growth on the market [7]. The competition among them is a heads up, both for adding new features to OS's and feasibility to develop additional applications. However, although these OS's are so rich in libraries and built-in features, they still face the heat of the market to match customer's high expectations. The basic architecture and support of programming language of OS's is very different from each other. Developed applications for a certain OS are not compatible for other OS's, indeed, they force developers to rebuild the same applications for them.

Cross-platform mobile development tools are gaining popularity in the world due to their characteristic to compile the application source code for multiple supported OS's. Such tools are mainly depending on web programming languages like HyperText Markup Language (HTML), JavaScript and Cascading Style Sheets (CSS) with some native wrapper code for accessing native Application Program Interfaces (API) like Camera, Contacts, etc. The application development is very easy and time saving with these tools. For example, DragonRad is providing Drag and Drop (D&D) features, which require reduced programming skills to develop applications [12]. There are plenty of such tools available now on the market, thus creating confusion among developers on which one to embrace and which one to skip. In the future, cross-platform tools can bring a drastic change in the business model of mobile OS's, especially due to the fact of reduction of development costs of new applications. Therefore, this paper proposes a survey on four major available cross-platform development tools on the market which are Rhodes, PhoneGap, DragonRad and MoSync.

The popularity of tools demands more research on this field and developers are expecting clear information about these tools before opting for one of them. The information is available at different chunks. Some books and papers in the past have provided some information from different sources. In [7], Dern has given some information with small comparison among tools, whereas in [22], Allen et al have given detailed description about mobile OS's and four mobile development tools. Another relevant paper by Hammershoj et al [2] provides good comparison among different available mobile OS's by keeping business model as important factor. Apart from above mentioned related works, there are books available for specific tools, for example *Beginning PhoneGap* by Myer [17].

In this paper we go further providing collected information about the four selected tools with the aim to help developers to evaluate and understand what is the more appropriate tool to use to match their requirements.

The structure of the paper is as follows. Section II presents

preliminary ideas about mobile OS's, market status, effects on existing business model and benefits using a cross-platform development approach. Section III shows the criteria used for tools selection and comparison. Sections IV, V, VI and VII provide information about each specific tool. Section VIII shows the comparison about tools that have been presented in previous sections. At the end, Summary and Conclusion is presented in Section IX.

II. Mobile Operating Systems and Cross-Platform Benefits

The intensive growth in mobile industry is demanding high performance for mobile OS's, so technology giants, such as Apple, Microsoft, Nokia, Symbian and Google are playing an important role. Each of them has introduced its own product to fit on the market. All OS's are exciting and providing something unique to attract customers. Some years ago, Symbian was very popular and one of the best OS's for developers, but currently Symbian's market share is dropped to 50 percent from an earlier share of 70 percent [12]. One reason of the market share loss has been the introduction of other mobile OS's from other market giants. Most of the newly introduced OS's are very rich in features and convenient for developers to create and deploy new applications. The major advantage in these OS's is the built-in features, such as Wi-Fi, Gallery, Bluetooth, Contact, etc.; developers do not need to develop these from scratch, which is a big relief and timesaving. Although all new OS's are very efficient, they still require much time and investment to develop new applications. The intensive competition does not give much space for companies to be slowed down in launching new applications, so all these OS's have definitely made developers' a life comparatively easier, but simultaneously have arisen many challenges to be competitive on the market [11].

Currently, the major mobile OS's are Android, Bada, BlackBerry, iOS, MeeGo, Symbian, webOS, Windows Mobile/Phone, but globally the family of mobile OS's is more extended. For developers that would develop applications on different OS's is really problematic because each OS has its own language, different API's, different Integrated Development Environments (IDE), etc. To meet the needs of developers, cross-platform mobile development tools have been developed with the purpose to give them the possibility to write the application source code once and run it on different OS's. Benefits that these tools have brought are:

- **Reduction of required skills** for developers to develop applications due to the use of common programming languages;
- **Reduction of coding**, because the source code is written once and it is compiled for each supported OS;
- **Reduction of development time and long term maintenance costs**;
- **Decrement of API knowledge**, because with these tools is not needed to know the API's of each OS, but only the API's provided by the selected tool;
- **Greater ease of development** compared to building native applications for each OS; and
- **Increment of market share** for the corresponding business model with the advantage to raise the Return On Investment (ROI).

The business model includes all activities related to commercial transaction [2]. The increased usage of cross-platform mobile tools can have some effects on the respective business model of each mobile OS, such as App Store for Apple, Google Play for Android, etc. One of the bigger effects of these tools is to expand the application sale on more markets as much as possible with the aim to increase the gain by both parties, business model owners and developers. Furthermore, especially for companies that are looking for these tools, the ROI could play a very important role. In fact, developing an application and selling it on multiple markets with these tools will decrease investment costs. Then the reduction in capital investment for applications could encourage developers and companies to invest more in developing new applications and register them in the respective business model. Finally, also another important variation related to the business model might come out by the use of these tools due to the support of a subset of OS's.

III. Criteria used for tools selection and comparison

This section explains the reasons behind tools choice and illustrates predetermined criteria decided for comparison among them. The selection of tools was done by considering frameworks that can generate applications at least on main mobile OS's, like Android, BlackBerry, iOS, Windows Mobile/Phone. Nowadays, there are many tools available on the market, but only few can generate applications for the defined mobile OS's. For example, other tools such as Corona, WidgetPad, Sencha, Titanium, and TotalCross do not have the versatility of supporting a wide range of mobile OS's.

To maintain a logical thread on the comparison, criteria that have been chosen could be helpful for developers to understand which tool could be appropriate for their purposes. Some of the properties provided by each tool that have been taken in consideration are:

- **Mobile Operating Systems** supported to understand possible effects on respective business models;
- **Tool licences** offered to evaluate the terms and conditions of use;
- **Programming languages** offered to developers for building applications;
- **Availability of API's** provided with the aim to get an idea of different hardware parts accessible in the OS;
- **Accessibility to native API's** to compare how it is possible to access them from each tool;
- **Architecture** provided for the development process of the application; and
- **Integrated Development Environments** available for developing applications.

IV. RHODES

Rhodes 3.3.3 is a cross-platform mobile application tool developed by Motorola Solutions Inc. under Massachusetts Institute of Technology (MIT). It is developed to rapidly build native applications for all major mobile OS's (iOS, Android, BlackBerry, Windows Mobile/Phone and Symbian). The main goal of Rhodes is to provide a high level of productivity and portability in programming. It is an open source Ruby-based mobile development environment. Thanks to this environment, developers can create and maintain enterprise applications and data based on single source code across different mobile OS's [22].

RhoMobile suite provides an IDE called RhoStudio which is an innovative solution dedicated to users that want to develop applications through a hosted IDE. This solution can be used across Linux, Mac, and Microsoft Windows OS's. Alternatively, RhoMobile offers the possibility to write applications with any other editor or IDE which supports HTML, HTML5, CSS, JavaScript and Ruby. The most popular editors are Eclipse, Visual Studio, Netbeans, IntelliJ and TextMate [16].

Rhodes provides native device applications to improve the end-user experience, which work with synchronized local data and take advantage of device capabilities, such as Barcode, Bluetooth, Calendar, Camera, Contacts, GPS, Menu, Near Field Communication (NFC), Screen Rotation, etc. [16].

Rhodes is the only framework that uses Model View Controller (MVC) pattern to develop mobile applications. The MVC pattern creates applications that separate data definitions (models) from business logic and (controllers) from interfaces (views), providing at the same time a point of connection between these elements [16]. Languages used in the view element are HTML, CSS and JavaScript to make mobile applications, whereas in the controller element is Ruby to make the backend support. Moreover, with MVC approach is also possible to write applications that use only the view element. Obviously, it is realizable for applications or sites that require a low level of complexity [20]. Rhodes provides mainly three possibilities to add extendibility in its framework, first can be done by adding external Ruby library to Rhodes, second by creating native extensions for specific Software Development Kit (SDK) of each OS and last by extending the already existing views available in Rhodes.

In Fig. 1 Rhodes architecture is shown. Controller, HTML templates and source adapter components are the parts which developers have to implement for the creation of applications, whereas other components are provided by Rhodes such as Rhodes App Generator which is an IDE than can be RhoStudio or another editor, Ruby Executor is the executor of the Ruby code, Device Capabilities are the API's, Rhom is a mini database ORM (object relational mapper) which provides a high level interface to make it quickly and easily (i.e. the database is SQLite for all mobile OS's except BlackBerry that is HSQLDB), RhoSync Client is a library to add sync data capability to your applications, and RhoSync simplifies the development of connectivity to enterprise backend applications. Moreover, performing the backend application integration between RhoSync Client and RhoSync Server is reduced by 50-80% the development effort [16].

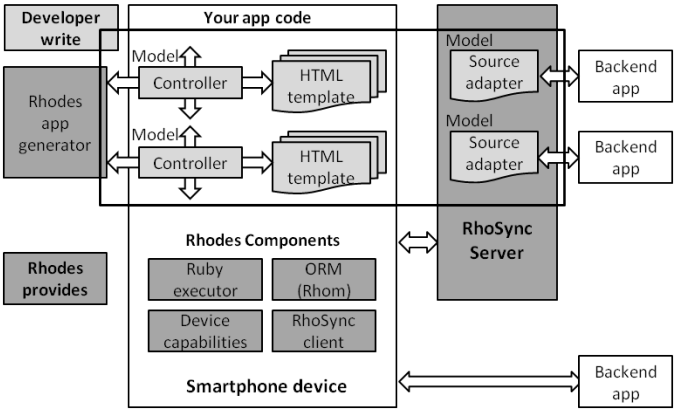

Fig. 1. Rhodes interfacing architecture between the Smartphone and Rhodes components [14].

Rhodes development files are compiled into native applications that can be executed on real or a virtual devices, indeed, this tool offers a desktop simulator where is possible to run applications. Applications developed with Rhodes are compiled into Java bytecode to be executed on BlackBerry OS, or compiled into Ruby 1.9 bytecode to be executed on all other OS's [22].

V. PHONEGAP

PhoneGap 1.9.0 is an open-source mobile development tool developed by Adobe System Inc. under Apache 2.0 license. PhoneGap allows developers and companies to build free, commercial and open-source applications, and give them also the possibility to use any licenses combination [19]. The development environment is cross-platform and permits the creation of applications for Android, Bada, BlackBerry, iOS, Symbian, webOS and Windows Phone OS's.

PhoneGap is a useful solution for building mobile applications using modern web programming languages, such as HTML, HTML5, CSS, CSS3 and JavaScript, and the functionality of SDK's instead to use less-known languages such as Objective-C or other languages [9]. It has the benefit to bring many advantages to skilled developers and especially to attract web developers [17].

Essentially, PhoneGap is a "wrapper" that allows developers to enclose applications written in known programming languages into native applications. Moreover, as each valid open-source software it is composed by many components and extensions. PhoneGap applications are hybrid, which means that they are not purely native or web-based. The meaning of "not purely native" comes from the layout rendering that is done via web-view instead of the native language of the OS, whereas "not purely web-based" comes from the lack support of HTML in some functions [22]. Besides, PhoneGap also

offers the possibility to extend the tool by developing own plug-ins.

Adopting a cross-platform approach the applications building and maintenance can be enhanced because developers have to write a single source code for any mobile OS supported by the tool. PhoneGap does not provide an IDE to develop applications, but developers have to write the source code with an IDE and port the it on other IDE's (e.g. Eclipse for Android, XCode for iOS, etc.). This approach does not allow developers to have a centralized development environment, so the effort required to compile the source code and produce the executable application (final product) is high. Thanks to the use of different IDE's for the development, PhoneGap can be performed on different PC OS's such as Mac, Linux and Microsoft Windows. Unfortunately, sometimes there are some exceptions because not all IDE's are compatible with all PC OS's.

A. PhoneGap architecture

The PhoneGap's architecture is composed mainly of 3 layers: Web Application, PhoneGap, and OS and native API's.

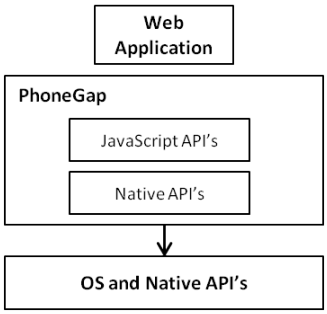

Fig. 2. Interfacing layers of the PhoneGap architecture [15].

In Fig. 2 the top layer represents the application source code. The central layer is composed by JavaScript and native API's. Mainly, this layer is responsible for the interfacing between web application and PhoneGap layers. Furthermore, it also takes care of the interfacing between JavaScript API's that are used by the application with native API's that are used by mobile OS's. The functionality of this layer is to maintain the relationship between JavaScript API's and native API's of each mobile OS. PhoneGap provides JavaScript API's to developers that allow the access to advanced device functionality, such as Accelerometer, Barcode, Bluetooth, Calendar, Camera, Compass, Connection, Contacts, File, GPS, Menu, NFC, etc. [19].

In Fig. 3 is shown a more detailed architecture schema provided by IBM. It represents all components about the web application, HTML rendering engine, PhoneGap API's and OS layers. Moreover, some different interfaces are shown in detail, such as the interfacing between PhoneGap API's and native API's layers.

VI. DragonRad

DragonRad 5.0 is a cross-platform mobile application development platform by Seregon Solutions Inc. and distributed

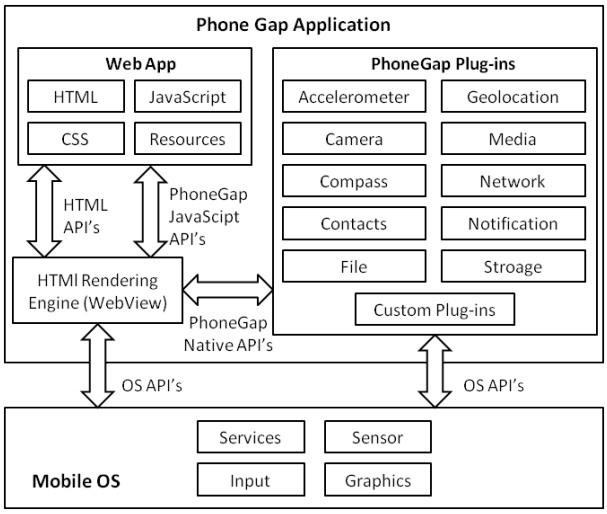

Fig. 3. Complete schema of PhoneGap architecture and interfacing among components [5].

under a commercial license. It allows developers to design, manage and deploy mobile applications once and use it across iOS, Android, BlackBerry and Windows Mobile [3]. The tool focuses on database driven mobile enterprise applications with easy and wide range of databases support. It provides the D&D environment which help developers to save programming time and to create logics. DragonRad provides their own built IDE, that can be configured for different simulators like iOS, Android, BlackBerry, Windows Mobile etc. As DragonRad has host-client architecture, it is required to setup server and database based on the needs of developers but it also comes in complete package with all prerequisites of server and database like Tomcat, MySql etc. DragonRad is commercial tool with the support to its own language D&D, the possibilities of extension in terms of adding plugins and other support to the framework are quite limited.

DragonRad facilitates the integration and synchronization of database system with native functions of above defined mobile OS's, such as Contacts, Calendar, Geolocation, Menu and Storage. The Architecture of DragonRad mainly composed of three major components [8]:

A. DragonRad Designer

It is a D&D visual environment or GUI for developers to design, develop and install mobile applications. Features of D&D are not only helping developers to design applications, but also reduce the efforts for maintenance and coding [8].

B. DragonRad Host

DragonRad host component could be run on either Linux or windows server which fill the gap between database of enterprise and mobile applications. It helps to maintain the communication with mobile device, which also includes query of transaction during network unavailability. It also plays the role to establish problem free connection with database

access and updates with synchronization. The following list summarizes the most relevant features of DragonRad host [8]:

- Taking data query from the device;
- Executing data query on specific target;
- Sending data back to device based on request;
- Handling data updates from the devices and updating databases; and
- Compression and data checking of data packets.

C. DragonRad Client

This component behaves like a native application on device which helps to run and interpret code of the created application by designer. DragonRad has the emulator to run and debug the application. This component also has the feature to customize application like change icon, application name, DragonRad project and link for installation of DragonRad host. By changing the link to DragonRad host, the application automatically connects to the given host when it started. It would also help in updating the project when required. One advanced function provided by the DragonRad company is to compile the application online [3].

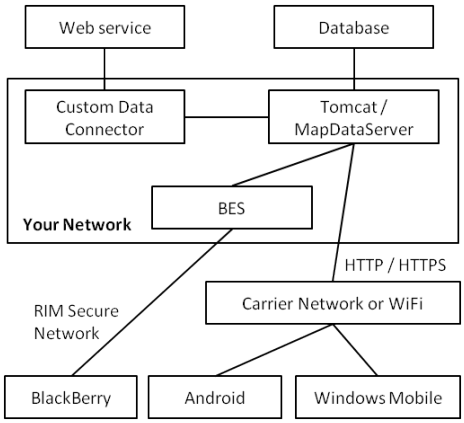

Fig. 4. DragonRad architecture with the connectivity of different components in it [8].

In Fig. 4, the area containing your network provides all backend support such as a custom data connector that could fit for any web service available, while Tomcat/MapDataServer is to support for the database. This full network is connected to mobile phone with different OS's with the help of the Wi-Fi. The other sub-part BlackBerry Enterprise Server (BES) is specifically for BlackBerry products, such as PlayBook. This tool allows creating application in very easy manner with having only three step, which are connect, build and deploy [8].

In DragonRad, it needs one connector that could be used to connect data source to DragonRad host web application. Therefore, it provides custom data connector to meet features. To handle the data transmission between back-end and device this connector is useful. As shown in Fig. 5 custom data connector is responsible to make connection of DragonRad host web application with database. DragonRad designer and

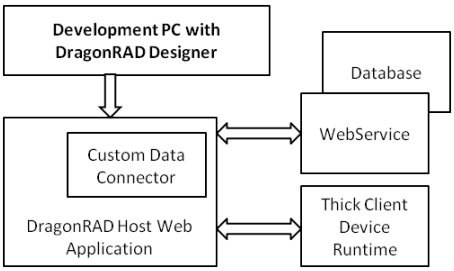

Fig. 5. DragonRad custom data connectors architecture [8].

device runtime are for respective work. There are few other features of this custom data connector [8].

- Responsible for transferring data request and updates form device to database with keeping synchronization;
- Receiving data queries from device;
- Processing received data queries;
- Re-query data or resend stored data based on the query;
- Receiving notification form device; and
- Sending data to device.

VII. MoSync

MoSync 4.0 is an open source solution developed by a Swedish company targeted to mobile market. MoSync has fully fledged SDK which helps developers to build and package all type of mobile applications, such as simple, advanced and complex application that share the same code base. MoSync SDK is proving to be very powerful tool with many components tightly coupled together like Libraries, Runtimes, Device Profile Database and Compilers, and so on. It provides the full fledge Eclipse-based IDE and the use of standard C/C++. It also added the support with web-based language like HTML, HTML5, CSS and JavaScript. It provides well documented API's both in C/C++ and web-based. The idea involved to support multiple mobile OS's is different from other tools and also in very isolated way from other mobile operating code. Applications in MoSync are built to target a device profile by using GNU Compiler Collection (GCC) and pipetool. After writing the application, pipe-tool is used to compile the resources present in the application. Then GCC backend is called and path to target device profile passed to it. GCC uses it to produce MoSync intermediate language, which then fed in to pipe-tool. Then, pipe-tool behaves as the bridge between MoSync applications to target device profile. The profile database helps the application in ensuring that it has adapted correctly to the device. Runtimes are libraries which are bound to provide support related to all like regarding graphics, audio, communications, input, uniform interface to low level system API's and other device features [1]. MoSync is completely open source and based on the Eclipse for IDE, so it provides to add extendibility in the same way as Eclipse does, i.e plugins or adding external library.

A. MoSync Architecture

MoSync has mainly two architectures specific to each C/C++ and web based languages. The idea and most of the

components are same except few. Architecture mainly consists of eight components as shown in Fig. 6.

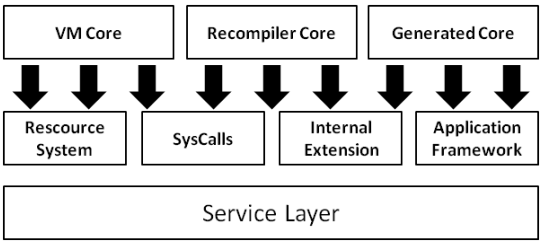

Fig. 6. MoSync Runtime Architecture, Showing the layered structured concept use in it [4].

1) Service layer: supports many functions like file I/O, threading, networking, memory management and many more functions [4].

2) Application framework: is responsible for runtime entry point. It is primarily to manage platform specific events like event management, initialization and destruction. The size and level of responsibilities varies across multiple platforms [4].

3) Syscalls: provides support to all features required by OS's, such as camera, contact, images, audio and networking etc which is specific to multiple platform. It is also responsible to interprets MoSync resource files and supports some like event management, initialization and destructions [4].

4) Resource System: manages resource objects such as images, sound and data blobs with the support of dynamic creation and destruction of resources [4].

5) Internal Extensions: specifies the design and configuration of each OS. Not all platforms contain the same type of features except few mandatory one, so missing features are implemented through single Syscall and known as numbered functions. When function is not found by Syscall it throws one error related to feature, which makes developers life little easy to determine a non-universal API is accessible in runtime [3].

6) Core: is responsible for the execution of MoSync programs by interoperating with Syscall and resource system, MoSync offers mainly three different types of core, which all share same common interface [4].

7) Virtual Machine Core: is the component that provides the support to load, interpret and run MoSync bytecode directly. The execution is taken care by single; small function that allows efficient Just in time (JIT) optimization. The whole structure is very similar to the core of Java Mobile Edition (Java ME) [4].

8) Recompiler Core: is the component that loads MoSync bytecode and recompiles it on the specific platform or typically Advanced RISC Machine (ARM) machine. After this recompilation, the generated code is executed. This core has many similarities with windows mobile and symbian [4].

9) Generated core: The core is responsible for the exhibiting interface with the generated native code. At this level it does not have any connection with the bytecode. The type of core is like iOS core. So the reason for having three different types of core in MoSync has versatile advantages. For example VM core is best for debugging and its dynamic approach makes it possible to load new code at runtime. This property proved to be very useful for many applications. Recompiler core is more efficient but less debugging support and its dynamic behavior also help in fast recompilation of some code. At last generated core have zero overhead for low end devices which are not able to load code at run time [4].

VIII. Comparison of Tools

This section presents the comparison made among the tools based on the comparison criteria explained in Section III. The parameters defined are explained here in details. Table I shows the different kinds of mobile OS's supported by each tool. Furthermore, on the right side of Table I OS's supported by respective tool are shown [16], [21].

Tool Name	Mobile OS Support	OS Support
Rhodes	Android, BlackBerry, iOS, Symbian, Windows Mobile, Windows Phone	Linux, Mac, Windows
PhoneGap	Android, BlackBerry, iOS, Symbian, WebOS, Windows Phone	Linux, Mac, Windows
DragonRad	Android, BlackBerry, iOS, Windows Mobile	Linux, Mac, Windows
MoSync	Android, iOS, BlackBerry, JavaME, Symbian, Windows Mobile	Linux, Mac, Windows,

TABLE I
COMPARISON ON MOBILE PLATFORMS COMPATIBILITY AND DEVELOPMENT ENVIRONMENTS OS'S SUPPORT.

From Table I, it is easy to understand that PhoneGap offers more compatibility for development on different mobile OS's, which is optimal to gain the maximum profit for both developers and business model owners because one built application can be registered at all respective business models, whereas IDE's hosted in personal computer is the same for each platform.

Table II describes several characteristics of the tools presented in the paper, such as supported programming languages, accessibility to native API's, IDE and plug-in extendibility. Starting from the programming languages, Rhodes, PhoneGap and MoSync are the only tools that have support for web programming languages, such as HTML, HTML5, CSS and JavaScript. Differently to other tools, PhoneGap provides support for CSS3, whereas MoSync for C and C++. Instead, DragonRad has support for its own language D&D.

As mentioned in the previous paragraph, since Rhodes, PhoneGap and MoSync are web oriented development tools, the access to native API's is done through JavaScript API's. In Addition, MoSync also offers the accessibility through C/C++ API's. On the other hand, DrangonRad provides its own API's to interface the application layer with the OS layer.

With respect to available IDE's, Rhodes has two useful ways to develop applications; in fact it provides some different solutions as RhoStudio IDE to develop in locale and RhoHub IDE to develop via remote connection. Furthermore, it provides the

possibility to use alternative IDE's, such as Eclipse, Visual Studio, Netbeans, IntelliJ, Textemate, etc. MoSync offers an Eclipse based IDE, whereas DragonRad offers its own IDE solution. PhoneGap is the only platform that has different approaches among tools, indeed it offers an extension that could be applied on all native IDE's. Some examples of native IDE's are XCode for iOS, Eclipse for Android, Visual Studio for Microsoft, etc. This kind of solution is fine but limits the liberty for developers to use the IDE of their choice. A careful reader would argue, what are the advantages of this approach if developers have to create applications on different IDE's. It is worth nothing that PhoneGap, Mosync and Rhodes are the tools that support HTML, CSS and JavaScript languages; since these languages are not native the source code will be the same for all platforms. The only effort required by developers in using PhoneGap is to develop applications on an IDE and perform a simple porting of the source code in other IDE's [6], [21].

Apart DragonRad, other tools provide the possibility to add plug-ins in the IDE and also give the feasibility to develop plug-ins from scratch and then add it to the IDE.

Name	Language	Accessibility to native API's	IDE	Plug-in Extendibility
RhoMoblie	HTML, HTML5, CSS, JavaScript,	JavaScript	RhoStudio RhoHub, *	Yes
PhoneGap	HTML, HTML5 CSS, CSS3 JavaScript	JavaScript	IDE native of the mobile OS (e.g. Eclipse, Xcode)	Yes
DragonRad	D&D	na	DragonRad Designer	No
MoSync	HTML, HTML5, CSS, JavaScript C, C++	JavaScript, C, C++	Based on Eclipse	Yes

TABLE II
COMPARISON ON DEVELOPMENT FEATURES.

* others alternative IDE's, such as Eclipse, Visual Studio, Netbeans, IntelliJ, Textemate, etc.

The license of tools is another helpful parameter for the comparison. As shown in Table III, the licenses available for these tools are MIT, Apache 2.0, GNU General Public License 2 (GPL2) and Commercial. The first two licenses are free and moreover they offer an open-source support. This kind of approach is much important for all developers that would develop applications and want to provide a support to the development platform without having commercial restriction. On the other hand, commercial licenses could be useful for companies that want to receive the support directly from the manufacturer.

Unlike from other frameworks, Rhodes is the only framework with MVC support. One of the most important benefits of the MVC design pattern is the possibility to develop applications in a distributed way by means of the model, view and controller with the aim to separate one part from each other; so the advantage is to modify each part independently. Another benefit that MVC offers is the easy migration of legacy programs, because the view is separated from the model and controller. Furthermore it provides an environment that embeds different technologies across different locations and architecture that better support the scalability [10]. Rhodes MVC framework provides support to native mobile applications, the ability to write real business logic on local native applications. This explains the reason of so many robust enterprise applications written with it [18], [21].

Name	License	Open source	Architecture	MVC
Rhodes	MIT, Commercial	Yes	Local, Web	Yes
PhoneGap	Apache 2.0	Yes	Local, Web	No
DragonRad	Commercial	No	Translate	No
MoSync	GPL2, Commercial	Yes	Local, Web	No

TABLE III
COMPARISON OF GENERAL FEATURES.

Rhodes, PhoneGap and MoSync are the tools that provide architecture of interfacing between JavaScript API's Layer and native API's Layer. These platforms use a native language of mobile OS to access the hardware and software resources with the purpose to add basic functionalities to the JavaScript Engine and make it easy to use for the application as traditional library methods. Based on mobile OS in which the tool is interfacing, the user-code will be converted in native-code, such as Objective-C for iOS, Java for Android, etc. Some important API's that represent hardware and software functions are listed in Table IV. But in MoSync, the method for interfacing is available both though C/C++ and JavaScript API's. The Mosync has different library named as "WormHole" when the application is build in by using web technologies, wormHole is responsible for interacting with the native applications of the selected mobile OS [13]. In case of DragonRad, it provides hardware integration in its own method which is either by using different dependent libraries or by following some sort of defined wizards. The list [18] provides more details to understand it well according to the selected mobile OS application.

IX. SUMMARY AND CONCLUSIONS

This paper provides the comparison between four different cross-platform tools to develop applications on different mobile OS's. Currently, these tools are mainly used in companies with the aim to create applications designated to be sold on different market places, such as Apple Store, Play Store, etc.

The tools explained in this paper have been categorized on the basis of parameters explained in Section III. The development in DragonRad is based on D&D approach to facilitate the application development and to make the tool environment more user friendly, whereas in Rhodes, PhoneGab and Mosync the development is based on web languages. DragonRad uses API's written with their language to interface the application with native API's, instead the other tools

API Name	Rhodes JavaScript	PhoneGap JavaScript	MoSync JavaScript	MoSync C, C++	DragonRad
Accelerometer		√	√		
Barcode	√	√			√
Bluetooth	√	√		√	
Calender	√	√	√	√	√
Camera	√	√		√	
Capture		√	√	√	√
Compass		√	√		
Connection		√	√	√	
Contacts	√	√			√
Device	√	√	√	√	√
File	√	√	√	√	
Geolocation	√	√	√	√	√
Menu	√				√
NFC	√	√	√	√	√
Notification	√	√	√	√	
Screen Rotation	√	√		√	
Storage	√	√	√	√	√

TABLE IV
COMPARISON ON MAIN SUPPORTED API'S.

use JavaScript API's to interface the application with native API's. As mentioned in Section II, these tools offer many benefits including the high-level development, but on the other hand one of the most relevant bottlenecks of the cross-platform development approach is the performance in executing applications that use JavaScript API's. Lately, although the JavaScript performance has been increased significantly is not recommended to develop applications that implement very complex business functionality or background services, such as application DropBox. Other limitations of this approach are generally the poor support of high-end graphics and 3D technology, and lack of last features introduced by OS's, because for each update done in OS's supported by the tool the vendor should update its own tool as well. The development of complex applications such as games, benchmarks, etc. is dependent more on the access to built-in API's, which is different in all defined tools [19].

Rhodes, differently to other tools offers the support of the MVC framework, which is a very popular and convenient way to develop applications. By considering all these things, we can figure out that Rhodes tool has the edge over other three tools because it has the support for both web-based services and MVC framework. PhoneGap offers a wrapper that works with several IDE's and it is very versatile with respect to the extendibility with plug-ins and extensions. Mosync is the tool that supports the greater number of API's across JavaScript and C/C++, and the greatest number of programming languages supported. Finally, DrangonRad is the only tool which present many differences in almost all treated aspects, such as language, architecture, access to native API's and non-open source redistribution.

Future works will extend our comparison providing more details on architectures and how each layer interacts with each other. A more accurate analysis of the tool extendibility with plug-ins and extensions will be provided as well as a deeply analysis on cross-platform application performances. Besides, another possible extension of this work could be the comparison between development platforms.

REFERENCES

[1] M. AB. What is MoSync. http://www.mosync.com/content/mosync-cross-platform-mobile-development-made-easy, 2012.
[2] R. T. Allan Hammershoj, Antonio Sapuppo. Challenges for Mobile Application Development. 2010.
[3] C. Best. Seregon solutions announces support for blackberry playbook. http://www.prweb.com/releases/DragonRAD/PlayBook/prweb4960674.htm, January 2011.
[4] P. Broman. The runtime architecture. http://www.mosync.com/documentation/manualpages/runtime-architecture, June 2010.
[5] B. Curtis. PhoneGap and the Enterprice. http://www.slideshare.net/drbac/phonegap-day-ibm-phonegap-and-the-enterprise, July 2011.
[6] M. Dalu. Mobile Apps cross-platform development challenge: PhoneGap vs. Titanium vs. Rhodes. http://surgeworks.com/blog/lab-mobile/iphone/mobile-apps-cross-platform-development-challenge-phonegap-vs-titanium-vs-rhodes, January 2010.
[7] D. Dern. Tools and Toys: IEEE Spectrum. June 2010.
[8] DragonRad. http://dragonrad.com, 2012.
[9] J. Fermoso. Seeks to Bridge the Gap Between Mobile App Platforms. http://gigaom.com/2009/04/05/phonegap-seeks-to-bridge-the-gap-between-mobile-app-platforms, April 2009.
[10] IBM. Benefits of the mvc design pattern. http://publib.boulder.ibm.com/infocenter/adiehelp/v5r1m1/index.jsp?topic=2003.
[11] D. Infoway. White paper on mobile os and efforts towards open standards. http://www.dotcominfoway.com/attachments/268_white-paper-on-Mobile-OS-and-efforts-on-Open-standards.pdf, 2012.
[12] A. Jakl. Mobile Operating System: is it PC? April 2009.
[13] M. Kindborg. The wormhole javascript library. http://www.mosync.com/content/html5-javascript-wormhole, February 2012.
[14] Leckylao. Rhodes framework: Agile mobile web development. http://leckylao.com/2010/06/12/rhodes-framework-agile-mobile-web-development, June 2010.
[15] MokaByte. Applicazioni mobili negli scenari Enterprise. http://www2.mokabyte.it/cms/article.run?articleId=O5R-R6L-HN8-8R8_7f000001_18359738_2ff5bd55, March 2010.
[16] Motorola Solutions. http://www.motorola.com/Business/US-EN/Business+Product+and+Services/Software+and+Applications/RhoMobile+Suite, 2012.
[17] T. Myer. *Beginning PhoneGap*. Wrox, November 2011.
[18] T. Paananen. Smartphone Cross-Platform Frameworks, Bachelor's Thesis. April 2011.
[19] PhoneGap. http://phonegap.com, 2012.
[20] RhoMobile. What is so special about the rhodes smartphone app framework? http://rhomobile.com/blog/whats-so-special-about-the-rhodes-smartphone-app-framework, March 2010.
[21] J. Rowberg. Comparison: App Inventor, DroidDraw, RhoMobile, PhoneGap, Appcelerator, WebView, and AML. http://www.amlcode.com/2010/07/16/comparison-appinventor-rhomobile-phonegap-appcelerator-webview-and-aml, July 2010.
[22] V. G. Sarah Allen and L. Lundrigan. *ProSmartphone Cross-PlatformDevelopment*. Apress, 2010.