Survey, Comparison and Evaluation of Cross Platform Mobile Application Development Tools

Isabelle Dalmasso, Soumya Kanti Datta, Christian Bonnet, Navid Nikaein
Mobile Communication Department, EURECOM
Sophia Antipolis, France
{dalmasso, dattas, bonnet, nikaeinn}@eurecom.fr

Abstract—**Mobile application development is becoming more challenging with diverse platforms and their software development kits. In order to reduce the cost of development and reach out to maximum users across several platforms, developers are migrating to cross platform application development tools. In this paper, we provide several decision criteria beyond the portability concerns for choosing suitable cross platform tool for application development. The desirable requirements in a cross platform framework are identified. A general architecture for cross platform application development is discussed. Then a survey of several write once run anywhere tools (PhoneGap, Titanium, Sencha Touch) are provided along with a classification and comparison among the tools. To examine the performance in terms of CPU, memory usage, power consumption, Android test applications are developed using such tools. It is found that PhoneGap consumes less memory, CPU and power since it does not included dedicated UI components. Finally the paper summarizes the contributions and concludes with some future directions.**

Keywords-mobile development; cross platform tools; PhoneGap; Titanium; Android.

I. INTRODUCTION

The landscape of mobile platforms has seen major evolution in recent past. While BlackBerry, Bada and Symbian failed to reach out to the masses, iOS and Android have won the war of mobile platforms. In the era of smartphones and tablets, mobile applications are providing added value to several industries including transport, ecommerce, net banking, travel, retail and enterprise services. Developers are exploiting the state-of-the-art functionalities of the smart devices to offer revolutionizing user experience. In-turn they are becoming the engine for innovation. Thus it is of prime importance for a mobile platform provider to attract more and more developers in order to boast external investment and revenue via them. Not only the mobile platform owners and handset manufacturers but also network service providers and chipset makers are investing heavily to develop and release software development kits to reach out to the developers. Also there are several tool vendors like cross-platform tool vendors (PhoneGap, Titanium), app diagnostic tool vendors (BugSense) and more who are also trying to catch the attention of mobile application developers.

But the diversity of mobile platforms and the variety SDKs and other tools pose unique challenges. They include choice of SDK, user experience, stability of framework, ease of updating, cost of development for multiple platform and time to market an app. Most of the developers would like to release apps for major mobile platforms (iOS, Android) and provide a consistent user experience (UX) across the platforms. Developing an app for separate mobile platforms require in-depth knowledge of them and their SDKs. This increases the cost of development, ease of updating and time to market an app. This is where the cross platform development tools come into picture. Table I provides an in-depth comparison among native, mobile web and cross platform app development approaches.

Cross platform tools (e.g. PhoneGap, Titanium, Rhomobile) allow implementing an app and its user interface (UI) using web technologies like HTML, CSS. Then the app can be built for several mobile platforms (e.g. iOS, Android, Windows Phone 7, BlackBerry). This process is helpful only when a developer is willing to compromise user experience and more importance is given to launching of the app in several platforms to reach to maximum users. This approach allows developing app for multiple mobile platforms at the same time. Thus the cost of development and time to market the app is reduced.

TABLE I. DECISION FACTORS

Decision criterion	**Native approach**	**Mobile web approach**	**Cross platform approach**
Quality of UX	Excellent	Very good	Not as good as native apps
Quality of apps	High	Medium	Medium to low
Potential users	Limited to a particular mobile platform	Maximum including smartphones, tablets and other feature phones	Large - as it reaches to users of different platforms
App development cost	High	Low	Medium to low
Security of app	Excellent	Depends on browser security	Not good
Supportability	Complex	Simple	Medium to complex
Ease of updating	Complex	Simple	Medium to complex
Time-to-market	High	Medium	Short
App extension	Yes	Yes	Yes

The research is sponsored by the French project WL-Box 4G Pole SCS.

This paper presents several criteria beyond portability concerns to choose an appropriate cross platform tool for development. Several such tools are present at the moment. We have put forward the requirements of cross platform framework and the high level architecture of the same. In-depth survey of several such Write Once Run Anywhere (WORA) enlightens about the API & documentation, development environment, deployment, advantages and weakness. To evaluate the performance of such tools, we have developed Android apps with four such tools and measured the CPU usage, memory usage and power consumption. During the test, it is found that PhoneGap consumes less CPU, memory and power than other tools. But the app developed using PhoneGap does not have a very good UI. To create a better UI, additional tools are required. In that case, Sencha 2.0 stands out among others.

The rest of the paper is as follows. Section II outlines the basic requirements of a cross platform development framework. Section III portrays a general architecture of cross platform mobile application development. Section IV presents a detail survey of WORA tools. Section V classifies WORA tools and provides a comparison among the tools. Section VI demonstrates the performance testing results. Finally the paper concludes with some future directions.

II. REQUIREMENTS OF A CROSS-PLATFORM FRAMEWORK

We have identified the desirable requirements of any cross-platform framework as stated below:

- **Multiple mobile platform support**: The framework must support several mobile platforms. Support for Android and iOS are very essential since they have the largest share in the application markets.
- **Rich user interface:** Currently the cross platform tools can not provide rich user interface (UI) as native apps. Since the success of an application highly depends on user experience of the interface, rich UI development should be incorporated. Support for sophisticated graphics (2D, 3D), animation, multimedia are necessary.
- **Back-end communication:** Mobile devices promote an "always-connected" model in which the users are sharing material in social networking sites, watching videos, communicating via live chat, gathering information 24X7. There smooth support for backend communication protocols and data formats are absolute mandatory.
- **Security:** Applications developed by cross platform tools are not highly secure. Proper research needs to be carried out to secure the tools and applications.
- **Support for app-extensions**: It is required to install app extensions on top of existing applications like in-app purchase/billing capability.
- **Power consumption:** It is an important issue now-a-days with thousands of smartphones and tablets are being activated daily. The generated applications must be optimized for power.
- **Accessing built-in features:** The tools must be able to access the built-in features of a smart device. Use of camera, sensors, geo-localization and more features helps to provide better user experience.
- **Open source**: It attracts more of application developers and the developer community can participate in bug-fixes and further development. It is to be noted that this is not a technical requirement.

III. GENERAL ARCHITECTURE OF CROSS PLATFORM APPLICATION DEVELOPMENT

This section provides a general architecture of cross platform application development portrayed in Figure 1.

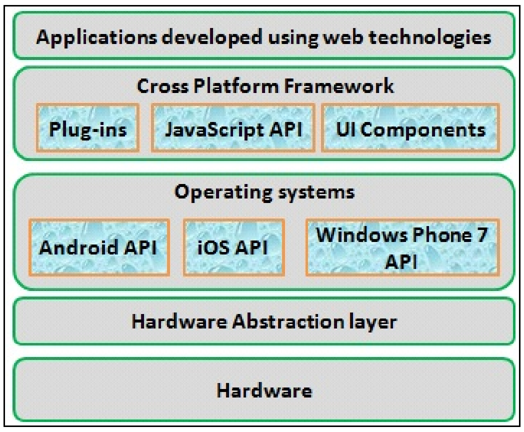


Figure 1: General architecture of cross platform mobile application development.

The application developer implements the business logic or the application functionalities using web technologies. The cross platform framework allows implementing user interface and access storage facility and device features (sensors, camera and contacts) which interacts with a JavaScript API. The API will in turn interact with the native API of a mobile platform. The application is then built separately to generate the executables for different platforms. The APIs for the mobile platforms actually allow generating the respective application. Thus the generated application can be run in corresponding mobile device.

IV. SURVEY OF WRITE ONCE RUN ANYWHERE TOOLS

There are several WORA tools available in the market e.g. PhoneGap, Titanium, Rhomobile, JQuery Mobile and more. This paper presents a survey of these tools to provide a quick overview to application developers.

A. PhoneGap

The PhoneGap open source framework provides a decent toolbox for building native mobile applications using only HTML, JavaScript and CSS [1], [11], [12], [13]. It’s quite popular among users mainly because of its flexibility, straightforward architecture and ease of use.

- **API & Documentation:** It offers a pure JavaScript / HTML / CSS API (Webkit framework) together with a library wrapping material's resources. PhoneGap also provides documentation for the proprietary API.

- **Environment Dev/Build:** It provides a Eclipse plug-in but can also be used from other IDEs to build mobile applications.
- **Deployment**: The binary file generated at the build cannot be published on any mobile application market. The developer obtains the final release through the pricing PhoneGap Build service.
- **Framework stability**: It is a mature framework.
- **Advantages**: All native wrapper source code is provided so it can be customized further. Broad ranges of platforms are supported by PhoneGap. Apps are built purely in HTML, JavaScript and CSS allowing web developers to adopt the tool easily.
- **Weakness**: There is lack of support for native UI components, design patterns and dev tools. However, the developers are free to combine PhoneGap with another tool like JQuery or Sencha to produce a better UI for the apps.

B. *Sencha Touch 2.0*

- **API & Documentation:** Sencha Touch 2.0 is a powerful yet a complex framework. The HTML code is generated by JavaScript APIs. In theory Sencha SDK Tools are self-contained development platform providing an access to a subset of phone native API (e.g. camera, notification, connection, orientation). It also offers to build native packaging deployable on iOS and Android application markets [4].
- **Framework stability**: It is a mature framework.
- Alternatively Sencha may be installed as JavaScript/ CSS resources of a PhoneGap project. This is a workaround but permits to develop the UI using Sencha API.

C. *PhoneGap + Sencha Touch 2.0*

- **API & Documentation: :** A complete JavaScript framework with MVC concept and complete API documentation are available for developers.
- **Environment Dev/Build:** Uses imported main CSS/ JavaScript files.
- **Framework stability**: It is a mature framework.

D. *PhoneGap + JQuery mobile*

JQuery mobile is a light-weight API and development of any application querying web services is quite easy [3].

- **API & Documentation:** JQuery Mobile is built on top of JQuery API and offers a JavaScript library of CSS and other components. Good documentation of the mobile API allows developers to familiarize with the tools quickly.
- **Environment Dev/Build:** The environment uses imported CSS and JavaScript file for a project and builds as any regular Android application.
- **Framework stability**: It is a mature framework.

E. *Application Craft*

It is a cloud based development environment for mobile, tablet and desktop. Application Craft permits to design a web application by connecting the platform through a browser,. The code generated is stored to the servers, with private/public address to run the application. The code generated is downloadable.

F. *Appcelerator Titanium Studio 2.0*

- **API & Documentation:** It provides a rich API and low level objects like TCP Sockets. UI objects are customizable through a JavaScript API. There is no HTML and CSS coding here. The tools presents good API documentation but the SDK tools usage (compiler etc) are not exposed [2].
- **Environment Dev/Build:** The IDE Titanium Studio (based on Eclipse IDE) is mandatory. It embeds the SDK and tools. The generation to native code then the compilation and packaging to a native application is made thanks a python script (for Android).
- **Framework stability**: It is a mature framework.
- **Advantages**: The native code output is very quick and smoothly executes on mobile devices. The setup is easy for new developers. The documentation is excellent and potentially supports tablet development.
- **Weakness**: It has restrictive APIs and small set of phones are supported.

V. Classification of Write Once Run Anywhere Tools

In this paper we concentrate on WORA tools as they provide the opportunity to implement an application for several mobile platforms. Efforts have been made to broadly categorize the tools based on the functionalities they provide. Five high level factors are listed that can guide developers to select a possible category.

A. *Application development tools*

- **Hybrid App:** This category of tools provides a platform-specific shell application which has the capability of rendering pre-packaged HTML pages. It extends the HTML capabilities through APIs which allow access to device specific features. Some Tools include libraries to render platform specific UI. AppMobi and PhoneGap are such category tools.

- **Mobile Web:** These tools are primarily JavaScript libraries which in combination with suitable HTML5 and associated CSS are rendering your mobile website on different types of device. Some of these tools can work in conjunction with Hybrid App tool and the result can be packaged into a native application. For example, JQTouch, JQurey Mobile and Sencha Touch.

- **Generator:** These are tools where a developer writes the application in a specific language (JavaScript or PHP) and the tool translates it into a deployable native application for different specific platforms. The deployable application may include a runtime engine or a virtual machine. The main difference between a Generator and a Hybrid App is the potential performance gain using the generated native code instead of translating HTML for graphic objects access. Appcelerator and Rhomobile[9] are typical examples of such tools.

B. *User Interface Tools*

- **Visual Tools:** They provide a visual interface where elements / widgets are dropped into the screen and the internal application plumbing is taken care by the tool. The result, depending on the tool is either a native application or a mobile website. Some visualization tools may be used in conjunction with application development tools (mainly the Hybrid App tools). Dragon rad [5], Application Craft [6], July Systems [7], NetBiscuit [8] are some examples of visual tools.

C. *Guildeline for developers*

We provide a guideline for developers to select an above mentioned category.

- **Native user experience:** If the resulting application is desired to have user experience similar to native applications, then "Generator" must be chosen. When user experience can be compromised, any other category is sufficient.
- **Offline/online usage:** Mobile web applications only allow online usage through a browser. Generator and hybrid apps create installable applications which permit offline usage.
- **Compatibility:** The mobile devices are evolving very fast in terms of hardware, mobile platforms and SDKs. If an cross platform application is desired for a long term, mobile web is the least risky as mobile devices are all supporting HTML5 and newer versions.
- **Limited access of built-in features:** In this case, hybrid apps can be preferred. But if applications require frequent access of built-in features, generator should be chosen although native approach will more suitable.
- **Security:** Hybrid and generator are likely to be more secure than pure mobile web applications. Better security can be offered by native approach.

D. *Types of applications developed using WORA Tools*

Several types of mobile applications are generated using the WORA tools: business, games, multimedia, tools, entertainment, ecommerce and social.

Some applications developed by PhoneGap are: Wikipedia (multimedia), BBC Olympics (multimedia), UnTapp (social), Facebook (social) and Zinga Game - Mafia Wars Mobile Game (game).

Some applications developed by Titanium are: Legoland Parc California (multimedia /entertainment / ecommerce), Ebay Corporate (business). New York Senate (multimedia) and ZipCar (ecommerce).

VI. PERFORMANCE EVALUATION

To examine the performance, Android test applications have been developed using cross platform method. They have the following features and can be categorized as a part of a 'business' type application.

- The UI presents buttons, each of which has request type. When clicking on a button the application queries free web services (available on the internet) over AJAX, REST and SOAP technologies.
- Parse and display the different formats of responses: Text, XML, JSON.

The applications are installed and tested on an Archos tablet running Ice Cream Sandwich (Android 4.0.4). The testing environment consists of Linux Fedora, Eclipse IDE Indigo Classic, web development Plug-ins and Android ADT 16. Figure 2 depicts the user interfaces of the test applications. It is to be noted that no measure has been taken to make a sophisticated user interface and the default version is adopted.

The above applications are representative of mobile machine-to-machine applications. Within the framework of the project WL-Box 4G [19], we are developing mobile applications that receive data from various sensors via a device gateway. The data is processed on-the-fly and the UI of the applications are updated.

A. *Memory usage*

It is found that such type of applications are developed in Titanium. But when the 'Guidelines for developers' are considered, the choice points to PhoneGap because of the application requirement. So the application is developed using the following tools and then the memory & CPU usage and power consumption are noted.

- Only PhoneGap.
- PhoneGap & JQuery mobile
- PhoneGap & Sencha Touch 2.0
- Titanium

The memory usage information is obtained from DDMS tool of ADT.

- **Proportional set size (PSS):** PSS is the amount of memory shared with other processes, account in a way that the amount is divided evenly between the processes that share it. This is memory that would not be released if the process was terminated, but is indicative of the amount that this process is “contributing” to overall memory load.
- **Unique set size (USS)**: USS is the set of pages that are unique to a process. This is the amount of memory that would be freed if the application gets terminated.

Table II provides the memory usage metrics.

TABLE II. MEMORY USAGE METRICS

Developed app	PSS	USS
PhoneGap only	12091	6036
PhoneGap + Jquery Mobile	14730	9424
PhoneGap + Sencha Touch 2.0	24526	20164
Titanium	17500	8676

From the above table it is clear that the application written in PhoneGap only has smallest PSS and USS value. This is due to the fact that PhoneGap by design does not use any styling element or tools for betterment of UI. So we can conclude that the memory usage will increase with the addition of features to generate better UI.

PhoneGap integrated with JQuery Mobile or Sencha Touch 2.0 is a complete environment for better UI development for application. In this case the memory usage increases with the use of HTML and JavaScript files.

Titanium comprises of a full SDK and the memory usage is thus on the higher side.

B. *CPU usage*

To measure the CPU usage,, the apps are developed using PhoneGap and other tools. The code segments to measure the CPU usage are added to the Android activity through PhoneGap. But no app could be developed using Titanium as it does not allow to add the activity. A plug-in could be developed for this purpose but that is time consuming. We have followed the two different avenues to record CPU usage as follows:

- The first approach takes a CPU snapshot at each state of the Activity life cycle (i.e. onCreate, onStart, onPause, onStop and onDestroy) of the apps.
- The other approach is to read a 'top' result every second during the whole life cycle of the apps. Then an average for each state of the Activity life cycle is computed.

The developed apps perform the same functionalities as the previous test. Table III portrays the results of the test.

The values obtained from CPU snapshot approach is computed when the app is doing much computation and is thus represented using very high values. Also it is to be noted that these values may vary a lot from a millisecond to another since they are snapshot at CPU usage for very short amount of time.

The values obtained from 'top' result shows varying CPU usages. The min value is always 0 as - once the app fetches the requested page and shows it, the app does not use any CPU waiting for the next system input. The average value is computed using the total elapsed time. From this approach, it is clear that the first app utilizes very less CPU but the user experience is not very sophisticated. When Sencha Touch 2.0 is used along with PhoneGap, the CPU usage is more but the user experience is significantly better.

TABLE III. CPU USAGE METRIC

Developed app	CPU usage from snapshot approach	CPU usage from 'top' command approach
PhoneGap + HTML + CSS tools	81.92771%	Max: 10% Min: 0% Average: 2%
PhoneGap + JQuery + HTML	80.26316%	Max: 42 Min: 0% Average: 10%
PhoneGap + Sencha Touch 2.0	44.0%	Max: 32% Min: 0% Average: 8%

C. *Power consumption*

Power consumption of mobile applications has received much attention of the researchers recently [14], [15], [16]. To effectively use the battery of mobile devices, the apps developed using the cross platform tools should be power efficient. We have measured the power consumption of the same apps using "Power Tutor' [17], [18]. It is a very popular Android app that reports power consumption of individual apps installed in a mobile device. Table IV tabulates the results. It is to be noted that the reported values are average power consumption of the apps.

TABLE IV. POWER CONSUMPTION METRIC

Developed app	Power consumption (mW)
PhoneGap + HTML + CSS tools	107
PhoneGap + JQuery + HTML	168
PhoneGap + Sencha Touch 2.0	120

Again the result points out that the first app consumes least power among the three apps. The reason is attributed to the fact that the UI is very simple. The app developed with PhoneGap and Sencha Touch 2.0 also works efficiently since it consumes 120mW power.

From the above evaluations, it is quite clear that PhoneGap and Sencha Touch 2.0 work efficiently in terms of CPU usage and power consumption. But currently there is no leader cross platform development tool in the market that fits all the general requirements. However the most popular tool among the developers are PhoneGap and Titanium. We identify the main differences between them in the Table V.

The above results are also analyzed with respect to the requirements of cross platform tools mentioned in Section II.

- The chosen tools are open source and can build applications for multiple operating systems. This paper describes the performance of the Android applications and iPhone applications will be developed and evaluated in future works.

- PhoneGap does not contain sophisticated UI building tools. Thus the memory, CPU and power requirements are less than PhoneGap combined with JQuery or Sencha Touch 2.0.
- Back-end communications are supported by all the tools.
- Rest of the requirements like security, app-extension will be evaluated in future work.

TABLE V. DIFFERENCE BETWEEN PHONEGAP AND TITANIUM

PhoneGap	Titanium
• A PhoneGap application is written in HTML5 and runs in an native container. • Developers can utilize HTML5, CSS and JavaScript also. • The UI elements may be rendered using JQuery Mobile, Snecha Touch or some other JavaScript web mobile development framework.	• A Titanium application is written in JavaScript and complies into a native application and utilizes native controls. • It is not an HTML5 application running in a web container. • By design, this tool can provide better performance and better user experience.

VII. CONCLUSION

In a nutshell, we have concentrated on cross platform development tools since they build apps for several platforms and the development cost and time to market are less. Although the user experience is not as good as native apps, but the apps can be released in several platforms at once to reach out to most of the potential users. We have described the general requirements of cross platform framework and its general architecture. Then a detail survey is presented that covers several aspects of the tools allowing developers to gain insight about the tools. Some high level guidelines to choose among the categories are provided for the developers. Example of applications developed by PhoneGap and Titanium are given. The performance of the Android test apps are measured in terms of memory, CPU usage and power consumption. The app written in PhoneGap is found to use minimum memory, CPU and power but provides a very simple user experience. It is also reported that PhoneGap with Sencha Touch 2.0 work significantly well when available memory is not an issue and better UI is desired.

In future we are targeting to identify several other performance test cases. At the same time, the other cross platform tools need to be examined using the memory, CPU and power consumption metrics described in this paper. Applications will be developed for iOS platform. Further experiments will be carried out to examine the security and app extension capabilities of the cross platform tools. One important issue that has not been addressed yet is access to context information from sensors as current mobile computing heavily depends on it. Such information mining using cross platform approach has to be researched further.

REFERENCES

[1] http://phonegap.com

[2] http://www.appcelerator.com

[3] http://jquerymobile.com

[4] http://www.sencha.com

[5] http://dragonrad.com

[6] http://www.applicationcraft.com

[7] http://julysystems.com

[8] http://www.netbiscuits.com

[9] http://www.rhomobile.com

[10] http://www.appmobi.com

[11] https://github.com/remy/PhoneGap-Plugin-WebSocket

[12] https://github.com/phonegap/phonegap-plugins/tree/master/iPhone/InAppPurchaseManager

[13] https://github.com/anismiles/websocket-android-phonegap

[14] S. K. Datta, "Android stack integration in embedded systems," in International Conference on Emerging Trends in Computer & Information Technology, Coimbatore, India, 2012.

[15] Datta, S.K.; Bonnet, C.; Nikaein, N., "Android power management: Current and future trends," First IEEE Workshop on Enabling Technologies for Smartphone and Internet of Things (ETSIoT), pp.48,53, 18 June 2012.

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

[17] L. Zhang, et al. "Accurate online power estimation and automatic battery behavior based power model generation for smartphones." In Proc. Of ACM CODES+ISSS'10, Arizona, USA, 2010, pp. 105-114.

[18] https://play.google.com/store/apps/details?id=edu.umich.PowerTutor

[19] http://www.pole-scs.org/projet/wl-box4g

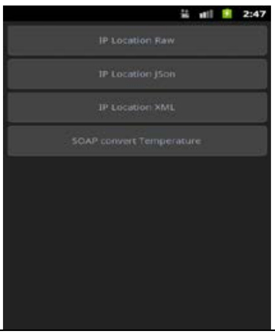
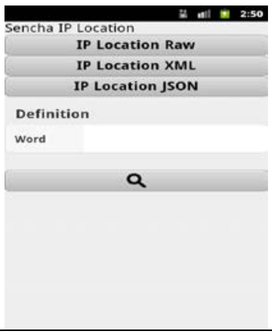
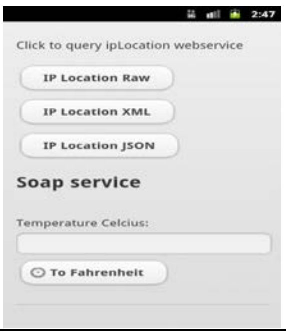
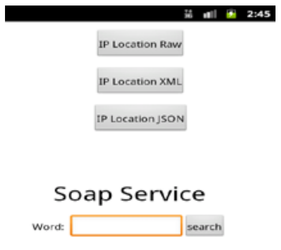
PhoneGap	Sencha	JQueryMobile	Titanium
			

Figure 2: User interfaces of the test Android applications.