Evaluation of cross-platform frameworks for mobile applications

Andreas Sommer, Stephan Krusche

andreas.sommer@in.tum.de, krusche@in.tum.de

Abstract: With today's ubiquity of smartphones, tablets and other mobile devices, more and more businesses develop and use mobile applications. Multiple platforms including Android and iOS form the market of mobile devices, so it can be important to be able to deliver software for more than one platform. Vendor-supported SDKs are feature-rich but incompatible with other platforms. We compared a number of cross-platform frameworks for mostly platform-independent development on mobile platforms, by evaluating them in several categories and weighing them against native SDKs. We found out that cross-platform solutions can be recommended in general, but they are still limited if high requirements apply regarding performance, usability or native user experience.

1 Introduction

The increasing demand for mobile applications in many areas [Syb11] presents developers and companies with several problems: implementing an application for multiple mobile platforms, like Android and iOS, may require high effort in terms of development time, resources, maintenance and possibly licenses, tools and deployment. One influencing factor is the fragmentation of the market of mobile platforms, i.e. smartphones and tablets. Android and iOS lead the sales with a high market share [Int12], while other platforms are less prevalent but might become more popular in the future (e.g. BlackBerry 10).

Using vendor-provided SDKs in order to implement an application for each desired platform results in highly responsive applications with a native look and feel, but has the downside of different, often incompatible programming languages, libraries, and user interface guidelines or designs. If instead an application is implemented with a common code base and possibly a single programming language or a small set of technologies, the compatibility issues are reduced, and also potentially the time required to implement and deploy the application on all desired platforms. Previous research regarding cost savings and other advantages exist in other areas of software reuse [A+09] [MC08], but not for all types of cross-platform solutions such as application frameworks. There seems to be no in-depth research available on the quantification of savings with existing cross-platform solutions, but *VisionMobile*'s developer survey of 2012 [Vis12] states that typically, more than 55% of production costs pertain to development and debugging, hinting that great savings could be achieved regarding invested time and resources for development.

Different cross-platform approaches emerged within the last years. In this paper, we describe which types of cross-platform solutions exist and evaluate three frameworks versus two native SDKs in criteria pertaining to functionality, usability features, performance and other categories. We focus on criteria and frameworks suited for developing business applications, i.e. excluding games in particular. Previous research in this area is sparse and often focuses on comparing a set of existing frameworks. We would like to conclude whether these solutions are useful at all and in which cases they can be recommended. The problem of supporting multiple platforms is not new but rather the reappearance of the same, 20-year old problem for PC platforms[1]. With the ubiquity of mobile devices and fragmentation of platforms, it makes sense for developers to target multiple platforms and thus for framework vendors to provide appropriate solutions [A+10].

In the following, we present existing types of cross-platform frameworks and the solutions selected for comparison (ch. 2). Chapter 3 details the methodology of the evaluation. We then list and explain the results for each category of criteria (ch. 4) and conclude about the advantages and problems of the solutions as compared with a native approach, and state in which cases the cross-platform approach is useful and ready for production (ch. 5).

2 Cross-platform frameworks

2.1 Definitions

Platforms are a combination of hardware (system plus any possibly built-in sensors or actuators), operating system, vendor-provided software SDKs and standard libraries [BH06], which together offer the base for building software for that platform.

A *framework* allows the reuse of a usually predefined application architecture and provides components that help in quickly setting up an application [Pas02]. Thus, a cross-platform framework allows to reuse parts of the application source code for multiple platforms and may additionally offer building blocks such as an architecture style (e.g. MVC[2]), user interface API or other functionality that is not specific to a single platform. Such frameworks can also expose APIs for platform-specific features.

Mobile applications exploit features specific to mobile devices and platforms, for instance a camera, accelerometer or fine-grained location retrieval. Pure web applications running in a mobile browser do not yet have access to all device features through a JavaScript interface [HHM12] and hence do not fall into the category of mobile applications.

[1] Windows, Unix, Mac OS, etc.
[2] Model, View, Controller

2.2 Types of cross-platform solutions

Pure web application frameworks Even though web applications without extensions were not considered as mentioned above, it should be noted that several frameworks exist which primarily rely on HTML as user interface technology and thus can be combined with pure web application frameworks. For example, *jQuery Mobile* [jF12] provides an architecture for web applications, including predefined user interface components[3], page loading and transitions. Other frameworks even contain custom styles for mobile platforms, e.g. components that look similar to iOS native user interface components.

Partially based on web technology Frameworks that primarily use HTML and related technologies to display the user interface, but additionally offer an API for calling device-specific functionality, belong to this category. Existing frameworks employ JavaScript to allow access to native functions, bridging between JavaScript function calls and native APIs (e.g. functions of the vendor-provided SDK for the underlying platform). However, there are frameworks that support calling native functionality but do not provide the means to implement user interfaces. For instance, *PhoneGap* [Ado12] offers many native functions but only supports using the platform's web view component[4] in order to display user interfaces – no UI functionality is included. Such frameworks can be combined with UI-focused frameworks, for example the web application frameworks mentioned above.

Compiled or interpreted code Unlike the previous category, frameworks of this type do not mainly use web technologies, but usually incorporate a different kind of user interface API and a single programming language which is then used across all supported platforms. For instance, the *Titanium* framework [App12] packages applications with a standalone JavaScript interpreter. Web-related APIs like DOM (Document Object Model) manipulation are not included and instead, Titanium offers its own API for generating a user interface from code, and for other purposes like network, device and file access.

Since these frameworks often support native user interface components, differences between the platforms' typical UI design must be considered. Therefore, different framework architectures can be found. Titanium offers an abstract interface for common components like buttons, but specialized APIs for platform-specific ones (e.g. iOS toolbar). Other frameworks only offer a programming language for platform-independent code, while requiring user interfaces to be developed separately for each mobile platform.

Other types A number of existing cross-platform solutions use approaches that differ from the above categories. For example, the *XMLVM* project has the goal of translating programming languages into each other. One example utilization is the cross-compilation of an Android application to a native iPhone application by translating its source code and adding a compatibility library that mimics the Android API on the iOS platform [XML12].

[3] Buttons, lists, text labels and fields, etc.

[4] Component supporting web browsing functionality, including rendering of HTML, JavaScript execution etc.

2.3 Compared frameworks

We conducted a case study comparing three cross-platform solutions with two native SDKs. The evaluation methodology is explained in the next chapter. In the following, the tested cross-platform frameworks and their architecture are outlined.

Titanium *Titanium* [App12] is a framework by *Appcelerator Inc.*, supporting Android and iOS with the latest version 2.0 at the time of evaluation. It leverages JavaScript as main development language, but does not primarily use HTML or other web technologies. HTML display using an embeddable web view component is supported, however. The architecture consists of two major parts: an application is bundled together with a standalone JavaScript interpreter to execute the application code, and the Titanium library that provides APIs for device functionality like sensors, file system access, native UI components and others, abstracting away many differences between the platforms. An application can use the Titanium library from anywhere in the source code. Calls to methods or properties of the global object *Ti* are passed to the native implementation for the respective platform. This bridging facility also allows own native extensions (platform-specific native code) to be added and used in a project from JavaScript code.

Rhodes The framework *Rhodes* [Mot12] was developed by the company *Rhomobile* which was acquired by *Motorola Solutions* in October 2011. The latest version at the time of evaluation was 3.3.2, supporting the 5 platforms Android, iOS, BlackBerry, Windows Mobile and Windows Phone 7.

The client-server model is used as basic architecture. The user interface of an application is shown entirely by a web view component that renders HTML content and evaluates JavaScript code. The application's backend contains models and controllers (based on the MVC architecture) written in the *Ruby* programming language. Views are written using HTML and a template language[5]. Any controller action can be accessed from the HTML-based views by accessing the respective URL. The framework includes a simple web server that is embedded in applications and runs locally on the device. This web server handles the translation of URL accesses to method calls in a controller and returns the view that is created as result. Controllers do not have to return a new view, but can also just run a task in the background without rendering an HTML page. They can directly interact with the user interface by sending JavaScript code which is executed by the web view.

Device capabilities can be accessed through the Rhodes library, and thus only from backend code (controllers written in Ruby, not from views[6]). A reduced Ruby standard library is included with the Ruby virtual machine that is bundled with an application. As a result, common functionality such as file access does not require the use of a separate API. Additionally, Rhodes provides modules for accessing, storing and synchronizing data.

[5] *ERB* (*Embedded Ruby*), a templating system that is also used in web frameworks such as *Ruby on Rails*

[6] In a subsequent version, it seems that some functionality can also be accessed in HTML and JavaScript.

PhoneGap and Sencha Touch *PhoneGap* [Ado12] is a framework that does not provide any UI generation functionality, but only the capability for displaying web technology based content. Therefore, we decided to combine PhoneGap with the web application and UI framework *Sencha Touch* [Sen12] because usability features are part of the evaluation criteria. The evaluation results are valid for this combination of frameworks and may vary if another web framework is chosen.

PhoneGap was initially developed by the company *Nitobi* which was acquired by *Adobe Systems* in 2011. As of the evaluated version 1.9.0, PhoneGap supports 7 mobile platforms: Android, iOS, BlackBerry, webOS, Windows Phone 7, Symbian and Bada. It allows development of mobile applications using web technologies by providing an interface to a web view component and tools to create platform-specific project files and initial source code that shows the web view. Additionally, PhoneGap provides APIs for device and platform functionality through a JavaScript-to-native bridge and native code can call back JavaScript functions inside the web view as well. This bridge is implemented differently on each platform. Developers can extend PhoneGap with native plugins by implementing a simple interface which also differs across the platforms. No user interface functionality of any sort is included, so PhoneGap can be seen as a wrapper for mobile applications that use HTML and other web technologies for displaying their user interface.

Features that are not natively supported on a platform are mimicked by PhoneGap's own implementation. For instance, if the *W3C Web SQL Database* specification is not implemented by a platform, PhoneGap's own implementation is used. Close adherence to such standards enables developers to use the same interface on all platforms.

3 Comparison

3.1 Methodology

The decision for the three selected cross-platform solutions was mostly based on active development, stability and popularity of the frameworks in order to gain an insight on solutions that are mostly ready for production use. We evaluated all solutions in five categories which are described below. Each category has a weight between 1 and 5 points (1=least important, 5=most important) which is multiplied with the score in a category. The values are summed up in order to gain the final result. Single scores are also in the range 1 to 5 (1=not fulfilled, 2=poorly fulfilled, 3=basic expectations met, 4=more than fulfilled, 5=all expectations completely fulfilled or exceeded). Since many criteria require evaluation of a framework's features from a developer's point of view, we based the comparison on the implementation of a well-defined sample business application with each solution. Criteria that are not related to development, such as license costs, were also considered to a certain extent. The main focus, however, was the usefulness of cross-platform solutions for developers and advantages for small to medium sized companies that develop mobile applications. The same methodology was applied to both

the Android and iOS native SDKs[7] so that comparable results could be achieved and potential advantages and pitfalls of both the cross-platform and native approaches would become clear during evaluation.

3.2 Sample application

The sample business application "MobiPrint" was invented as mobile app to support customers of a contrived chain of photo printing stores. Its main purpose is to order printed photographs over the Internet using a mobile device. Customers gain the convenience of sending pictures from their mobile device, without having to bring a physical medium to the store. The store itself saves time because pictures come over a common channel, and it is not necessary anymore to extract files from different media types (USB sticks, CDs, SD cards). Amongst other points, development of this application mainly tests

- **Integration with web services:** A publicly accessible, HTTP-based web service and its exposed actions are predefined and must be used by the sample application.
- **Usage of device features:** For example, the current location is used to retrieve information about nearby stores.
- **Additional data sources on the device:** In order to read and select pictures for upload to the web service (e.g. access to photos or file system).
- **Usability features:** Mockup designs for different screens were given and the UI had to be implemented from scratch with each of the five competitors. Thus, features and problems with e.g. layouting could be evaluated in each implementation.

3.3 Categories

The three cross-platform solutions and two native SDKs were evaluated in five categories which are roughly based on the *FURPS+* model [BD09][8], but adapted to describe the requirements of frameworks instead of finished software. The adapted model shall describe how well a framework supports the various categories when using it to develop software products, e.g. "features to improve usability" instead of "actual usability". FURPS+ was chosen because it concisely categorizes different types of software requirements.

Functionality This category typically comprises all functional requirements of a software product. For cross-platform frameworks, the available variety and quality of features

[7] Android SDK v10 (for Android 2.3), iOS 5.1 SDK

[8] Describes software requirements in functionality, usability, reliability, performance, supportability and other categories

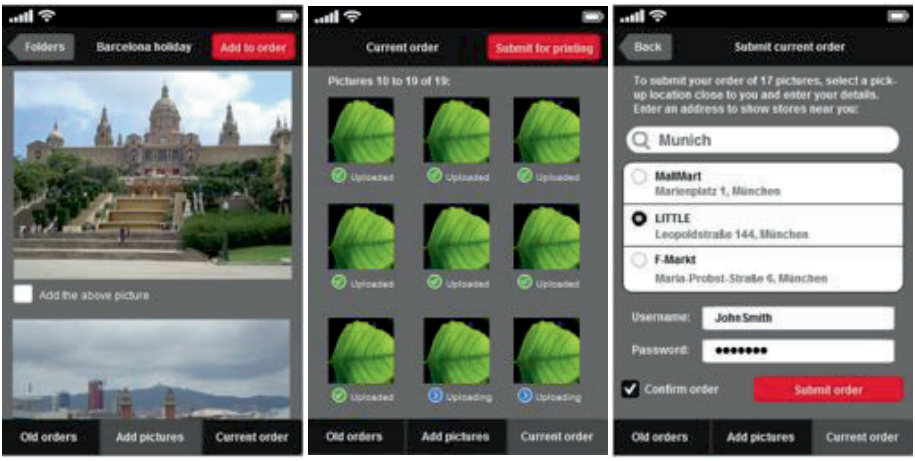

Figure 1: Mockups for the sample application screens

are considered, which includes: geolocation and network access, a persistence layer or database access, hardware sensors and actuators, buttons (hardware buttons, or software buttons provided as a persistent element of the operating system's user interface[9]), application lifecycle (events such as start, pause, resume; ability to run a background service or similar) and access to data provided by other applications or the operating system (e.g. contacts). Extensibility positively affects the score, for example if natively running code or access to features of a platform's native SDK is supported.

Usability features This category answers the question: how well does a framework support features and concepts that can help improve usability of the developed application? This includes features offered for creating a user interface: typical elements like buttons, lists, a tab or navigation bar, but also more mobile-specific elements like a paging component for switching between several pages by touch gestures.

The support for touch gestures is important because gestures (potentially platform-specific) improve learnability and satisfaction by allowing users to achieve certain actions quicker and in a familiar way (e.g. "pull to refresh"[10]). Another criterion is customizability, meaning which attributes of UI components can be changed and to what extent. This includes colors, transparency, font type and size or shapes. Corporate identity, for example, may prescribe a certain color scheme or font type. The better an interface can be accommodated to the intended design, the better its usability can potentially be – or in other words, if the necessary customizations and features are not available, some usability features (like gestures and UI patterns based on them) cannot be implemented, or only with high effort.

[9]Fixed button bar as introduced in Android 3.0, for example.

[10]If a list component is scrolled to the very top, and the user shortly swipes down with the finger, an application can recognize this gesture and e.g. refresh the list.

The incorporated points of comparison include UI functionality, i.e. basic and advanced features and components, customizability, gesture support and the ability to use native UI components, e.g. the iOS navigation bar. Moreover, the easiness of applying platform-specific looks and the overall subjective UI impression are taken into consideration.

Developer support This category aims at the ease of use that a framework offers to a developer. Documentation completeness and quality plays an important role, but also the development tools provided. Frameworks based on web technology may allow testing a mobile application in a browser or simulator, which would give developers the advantage of modern debugging tools built into browsers[11]. Other frameworks may provide their own simulator or support for using an existing simulator (e.g. iPhone simulator) to test applications. Some frameworks may restrict developer tools for common tasks (build, framework update, etc.), such as a certain IDE. Restrictions of this type may hinder the workflow of developers or existing automatic build setups. Overall, the following points are considered for the developer support category: documentation, debuggability and testability, tool restrictions and learning effort (e.g. number of technologies involved).

Reliability & Performance These categories are defined separately in the FURPS+ model – for simplicity, they are combined here. A reliable application is able to "perform its required functions under stated conditions for a specified period of time" [IEE90]. This is particularly important for mobile applications: since they typically run with restricted resources (e.g. available memory, process paused while application is in background), applications should run without crashes even if certain operations need a high amount of memory for a short time. Performance – specifically in the area of mobile applications – is mainly concerned with short response times for user interactions. Thus, an example of a poorly performing application feature could be a user interface that takes several seconds to switch between screens. Of course, general performance criteria for software apply, too, i.e. any measurable attribute that affects speed, waiting times or accuracy.

Performance was evaluated on two older test devices, the *Huawei Ideos X3*[12] and *iPhone 3GS*[13], in order to see how well the sample application implementations perform on low-end devices. For this category, the sample application's stability and size of the packaged/installed application are considered, as well as UI and application performance.

Deployment, Supportability, Costs Supportability is determined by the "ease of changes to the system after deployment" [BD09]. App stores are an important point to consider, in particular regarding the framework's support for packaging and its compatibility with restrictions and submission guidelines of the stores. Supportability also includes the portability to other systems – in this case, other mobile platforms. Experiences porting the sample business application from Android to iOS will be documented

[11] For instance the WebKit development tools that can display the HTML document tree, CSS (Cascading Style Sheets) rules, loading times, a JavaScript console and other helpful utilities

[12] Android 2.3.3, 600 MHz, 256 MB RAM

[13] Tested with iOS 5.1.1, 600 MHz, 256 MB RAM

in the evaluation. Portability for the native SDKs is rated low because native applications cannot be ported easily. This criterion is not weighted higher or regarded as separate category because there is little research available on quantitative differences in development time and costs when porting applications. Furthermore, the activity of framework development and the viability of vendors are important, especially if a business wants to use a certain framework in their long-term mobile strategy.

This category comprises a variety of criteria: a framework's supported mobile platforms, compatibility with app store guidelines, additional dependencies (e.g. a runtime environment has to be installed separately to run an application) and built-in support for internationalization, i.e. translations and localization (formatting of dates, currency values and other locale-dependent settings). We also considered support for simplified or automatic builds, for example through external build services or included tools, and checked how actively the frameworks are being developed. Regarding costs, we evaluated the direct costs for using a framework and additional (professional) support options and costs.

3.4 Category weights

Weights describing the importance of a category from the point of view of a small development company are assigned as follows (1=least important, 5=most important):

Functionality: 4 Certain built-in features and support for hardware sensors and actuators can be very important depending on the type of application. Extensibility, e.g. being able to add code that uses native functionality or to use third-party libraries, is even more important.

Usability features: 3 Usability greatly depends on a good application design. However, in case a framework does not provide extended features such as gesture recognition, some features of the finished application are harder to implement.

Developer support: 2 Tool restrictions and debuggability play a smaller role with mobile applications because they are typically less complex or require less development time than comparable desktop applications, and as such the need for helpful developer tools is not as important.

Reliability & Performance: 3 Performance is a critical point for customer satisfaction because slow loading times may prevent users from ongoing use of an application. Nevertheless, new technologies such as multi-core processors are already emerging and will help mitigate performance issues. Also, cross-platform frameworks are relatively new and still have the potential to progress in terms of performance. Together with reliability, a common and surely important aspect of software, this category is considered of medium importance overall.

Deployment, Supportability, Costs: 5 Costs and time-to-market are key factors for the success of an application. The typically short development cycle of mobile applications requires fast delivery of new versions, which can be simplified by automatic

build services or built-in packaging or deployment functionality. This category also comprises the fulfillment of app store rules, support for different platforms and how actively a framework is developed. Costs of a framework and available support plans are also evaluated. Therefore this category is most important because it includes factors that influence the cost, ease and speed of deployment in the long term.

4 Results

Table 1 summarizes the results of the evaluation. The scores show that the native SDKs win most categories. However, the actual differences in each category need to be considered.

Category	*Weight*	*Titanium*	*Rhodes*	*PhoneGap and Sencha Touch*	*Android SDK*	*iOS SDK*
Functionality	4	3	2	3	**4**	**4**
Usability features	3	4	3	3	**5**	**5**
Developer support	2	**4**	3	**4**	**4**	**4**
Reliability & Performance	3	3	2	2	4	**5**
Deployment, Supportability, Costs	5	**4**	**4**	**4**	**4**	3
Rounded final score:		**3.6**	**2.9**	**3.2**	**4.2**	**4.1**

Table 1: Overview of evaluation scores for all frameworks

4.1 Functionality

All three cross-platform frameworks support basic functionality like lifecycle events (except Rhodes), access to built-in hardware, intercepting button presses, or network, file and data access. In order to support all functions that the native SDKs offer, a framework would need many high-level abstractions or has to make compromises – Titanium supports some extra features of both Android and iOS without exposing them as platform-independent interface. The number of developers actively involved in the cross-platform frameworks is supposedly small in comparison to the native SDKs that are supported by large companies. Therefore, functionality of the frameworks is first of all restricted to what their developers can implement and support. PhoneGap takes a good approach here, in that only portable functionality is supported in the core API, whereas other features are available in separate plugins that are written in native code and thus can utilize all functions of the respective platform.

4.2 Usability features

The score difference in this category is mostly due to the support for native components or native-like styling capabilities. Titanium uses native components, but both Rhodes and PhoneGap are based on HTML, so achieving a fully native experience is cumbersome. Also, the native SDKs provide high customizability and well-designed UI components. Regarding quality and availability of features for usability, the cross-platform frameworks are rated good throughout, supporting the basic features as listed before (e.g. transitions).

The way how layouting works in the five solutions varies greatly, but with all of them, a customizable frontend can be created. Titanium additionally requires testing on both supported platforms because missing layouting attributes and different default behaviors can cause problems. The other two cross-platform solutions make porting very easy by using web technology. The subjective impression varies and the look of UI components differs greatly, which can be an important point when deciding between the solutions.

4.3 Developer support

Except for PhoneGap, which is used with separate native project files per platform and thus can be used with the respective IDEs, all other frameworks provide an IDE for development, debugging and deployment of applications. Rhodes is the only framework with a score of only 3 points because the learning effort for two programming languages (Ruby and JavaScript) is considered higher and Ruby code debugging is only supported on *RhoSimulator*, not on devices or other simulators. Typical requirements, such as debugging with breakpoints, are fulfilled by both the cross-platform frameworks and the native SDKs, rendering them pretty much equivalent for this category. Documentation quality and completeness is also adequate for serious development – solutions for Android and iOS SDK problems can be found more easily because of the large developer community.

4.4 Reliability & Performance

During implementation and testing of the sample application, no relevant crashes were found with any of the cross-platform frameworks and native SDKs, apart from memory management of loaded thumbnails from large pictures. With Titanium and the Android/iOS SDKs, references are managed manually, which led to crashes on the Android device because of the limited application memory. The native Android SDK solves this with functions to efficiently handle thumbnail loading which are however not available in Titanium. In these cases, the frameworks are still missing functionality. With the web technology based frameworks, memory management is performed automatically by the web view component and thus did not lead to similar issues.

Performance-wise, the cross-platform frameworks vary greatly from its native competitor. Both in observed reaction times and subjective regard, Titanium was the fastest of the

three cross-platform solutions, which, together with the native UI components, brings it close to the native SDKs. The choice of JavaScript, however, has certain implications: multithreading and asynchronous operations are not supported directly, apart from those APIs which are already implemented to run in the background (e.g. HTTP requests). PhoneGap as framework for use of web technology has this problem, while Rhodes and Sencha Touch have additional causes for their very poor performance[14]. PhoneGap's performance can be fast enough for many types of applications if the right web framework is chosen. Combined with Sencha Touch, it was very slow on the test devices, with screen switch times of up to multiple seconds. Rhodes has equally poor performance.

4.5 Deployment, Supportability, Costs

Each of the vendors of the tested cross-platform frameworks offers professional support with different options or payment plans. Google and the Open Handset Alliance do not seem to offer such plans for Android, and Apple only seems to offer e-mail support through a ticket system. Free support options such as online forums exist for all solutions, and a point that speaks for Android and iOS is that their developer community is so large that solutions to typical problems can almost always be found on the Internet and not only in the official forums. Even though support options differ, all five solutions can be considered very good in this category. One point was deducted from the iOS SDK score because of its only basic translation support[15] and the need to pay the membership fee to gain access to technical support and the bug tracker. Regarding internationalization capabilities, all frameworks are extensible so that external solutions can be used for internationalization and other purposes. Android already offers advanced support for translations.

5 Conclusion

A major advantage of the cross-platform frameworks is code reuse. In the evaluation, the sample application was always first implemented and tested on Android and only later we checked whether it runs unchanged on the iOS platform. Both HTML-based solutions, Rhodes and PhoneGap with Sencha Touch, did not require any changes to the code itself. Titanium, on the other hand, required small changes to the layouting attributes.

Regarding developer support, documentation and additional learning sources (screencasts, tutorials) are available for all evaluated solutions, and mostly of good quality even for the fairly young cross-platform frameworks. As the frameworks are under very active development, the quality can be expected to improve over time.

HTML-based framework types have specific advantages. For instance, PhoneGap requires close to no extra knowledge for an experienced web developer: project setup is simple

[14] Possible reasons are a complex and deeply nested HTML DOM tree with Sencha Touch and the client-server architecture with embedded web server of the Rhodes framework.

[15] Translation support was improved with the iOS SDK 6 which was not tested in this evaluation.

and applications can be developed with web technology only (unless PhoneGap plugins or platform-specific functionality are needed). In addition, an application can be deployed as normal web site if it can run without device features that are not available in a browser. Hence, PhoneGap is a good option if an application should also be the base for a mobile web site. Rhodes does not support web deployment in the evaluated version.

The main disadvantage of the cross-platform approach with the evaluated frameworks is the availability of usability features and the partially very poor performance as compared to the native implementations of our sample application. Rhodes and Sencha Touch do not support the native platform design which might be desired to present users with a more familiar interface, or to make sure an application is not rejected on app stores for disregarding UI guidelines. PhoneGap uses a better approach by allowing plugins to run native code and thus add platform-specific elements like the iOS tab bar. Regarding performance, it is notable that Facebook switched from mostly HTML5-based mobile applications (for Android and iOS) to native implementations and thereby achieved large improvements in UI responsiveness and overall performance [Dan12] [Du12].

In general, cross-platform solutions offer many advantages, in particular code reuse and quick setup for multiple platforms. If the mentioned usability and performance problems can be considered minor for an application, we can recommend to employ *Titanium* or *PhoneGap* for production use. Both frameworks can offer high performance[16]. Depending on the use case, considerations of desired user interface design (customized vs. native) and deployment as mobile web site, one framework has advantages over the other as described above. However, native development is still the better choice for usability-centered applications with a high focus on user experience. Also performance-critical applications, e.g. software performing heavy background processing, should be implemented natively. Nevertheless with modern devices, performance issues become less prevalent.

We expect the field of cross-platform mobile application frameworks to become appealing to developers through their ongoing development. As only little research was conducted in this area, many open questions exist: Will HTML5 become widely used when standards for accessing device features are implemented? How much quantitative influence on development/maintenance time and cost does the cross-platform approach have? Are other types of frameworks more suited to port an application to another platform? For instance, *XMLVM* provides translation of source code to a different target, and various other concepts exist. Also, our evaluation criteria were mostly based on aspects important for developers. Other viewpoints could be investigated as well. We could only test a small set of available solutions, and more frameworks may become ready for production use.

References

[A+09] Rama Akkiraju et al. Toward the Development of Cross-Platform Business Applications via Model-Driven Transformations. In *SERVICES I*, pages 585–592. IEEE Computer Society, 2009.

[16] PhoneGap's performance depends greatly on the HTML markup and related resources like CSS rules and JavaScript code. A fast web application framework, if at all required, is recommended for use with PhoneGap.

[A+10] Sarah Allen et al. *Pro Smartphone Cross-Platform Development: iPhone, BlackBerry, Windows Mobile, and Android Development and Distribution*. Apress, September 2010.

[Ado12] Adobe Systems Inc. PhoneGap. http://phonegap.com/, December 2012.

[App12] Appcelerator Inc. Appcelerator Titanium. https://www.appcelerator.com/platform, December 2012.

[BD09] Bernd Brügge and Allen H. Dutoit. *Object Oriented Software Engineering Using UML, Patterns, and Java (Third Edition)*. Prentice Hall International, 2009.

[BH06] Judith Bishop and Nigel Horspool. Cross-Platform Development: Software that Lasts. In *Software Engineering Workshop, 30th Annual IEEE/NASA*, pages 119–122, April 2006.

[Dan12] Jonathan Dann. Under the hood: Rebuilding Facebook for iOS. https://www.facebook.com/notes/facebook-engineering/under-the-hood-rebuilding-facebook-for-ios/10151036091753920, August 2012.

[Du12] Frank Qixing Du. Under the Hood: Rebuilding Facebook for Android. https://www.facebook.com/notes/facebook-engineering/under-the-hood-rebuilding-facebook-for-android/10151189598933920, December 2012.

[HHM12] Frederick Hirsch and Dominique Hazaël-Massieux. Device APIs Working Group. http://www.w3.org/2009/dap/, December 2012.

[IEE90] IEEE. IEEE Standard Glossary of Software Engineering Terminology. *IEEE Std 610.12*, 1990.

[Int12] International Data Corporation (IDC). Android- and iOS-Powered Smartphones Expand Their Share of the Market in the First Quarter, According to IDC. http://www.idc.com/getdoc.jsp?containerId=prUS23503312, May 2012.

[jF12] The jQuery Foundation. jQuery Mobile. http://jquerymobile.com/, December 2012.

[MC08] Parastoo Mohagheghi and Reidar Conradi. An empirical investigation of software reuse benefits in a large telecom product. *ACM Trans. Softw. Eng. Methodol.*, 17(3), June 2008.

[Mot12] Motorola Solutions Inc. RhoMobile Suite. http://rhomobile.com, December 2012.

[Pas02] Alessandro Pasetti. *Software frameworks and embedded control systems*. Springer-Verlag, Berlin, Heidelberg, March 2002.

[Sen12] Sencha Inc. Sencha Touch. https://www.sencha.com/products/touch, December 2012.

[Syb11] Sybase Inc. Enterprise Mobility Guide 2011. http://www.sybase.de/mobilityguide, 2011.

[Vis12] VisionMobile. Developer Economics 2012. http://www.visionmobile.com/product/developer-economics-2012, June 2012.

[XML12] XMLVM. Overview: Android to iPhone. http://www.xmlvm.org/android, December 2012.