Cross-Platform Mobile Application Development

ASQT 2013
Extended Abstract

André Nitze
Department of Cooperative Studies
Berlin School of Economics and Law
Berlin, Germany
andre.nitze@hwr-berlin.de

Andreas Schmietendorf
Department of Cooperative Studies
Berlin School of Economics and Law
Berlin, Germany
andreas.schmietendorf@hwr-berlin.de
I. INTRODUCTION

Mobile applications have become an important part of our everyday life. Many organizations are offering apps to customers and employees to increase retention and productivity [1]. But investment decisions are still a challenge regarding the unprecedented diversity of the mobile market. Smartphone vendors steadily improve the hardware and software of their products resulting in a plethora of device configurations – even within a specific vendor’s ecosystem. This causes severe problems for the software engineering discipline and general IT management. Multiple development teams (one for each platform) have to produce and maintain different codebases for one product.

Cross-platform mobile applications could be a viable solution to this. The idea of one application running on all platforms (e.g. iOS, Android, Windows Phone, Symbian, Bada, webOS) is compelling since development and maintenance efforts would decrease considerably. Analysts predict that more than half of the mobile apps will be hybrid by 2016 [2].

In this work we describe our practical experiences with the current state-of-the-art of cross-platform app development and explain why enterprises should now begin to think about shifting their own or suppliers’ mobile software development approaches.

II. BACKGROUND

Cross-platform applications hinge on a single consistent codebase promising less development and maintenance effort, more focused development expertise and thus a more flexible approach for creating software products.

The available technological options for developing cross-platform mobile applications are web-based apps and hybrid apps.

Web-based apps (or ‘web apps’) are basically web sites, which are optimized for mobile devices. Because of this, they are said to lack device-specific functionality and performance [3, 4]. Whereas performance issues are slowly decreasing due to optimizations, missing functionality is not an option for today’s sensor-packed mobile devices.

As opposed to this, hybrid apps, though also relying on standardized web technologies, are bundled within a native app “container”, which serves as a bridge to access device hardware and functions like ringing, vibrating, notifications, making calls, using the camera, GPS sensor and accelerometer. The web standards mainly used are JavaScript for business logic and HTML5 and CSS3 for presentation (forms, styling, transitions etc.) and offline storage. These technologies are used by most modern web browsers and therefore they are supported more and more completely by all vendors [5]. As a result of the web technology-based approach, cross-platform apps also circumvent the vendor-specific development process (special programming languages and IDEs).

A recent tech-demo covering core features of Facebook impressively showed that web standards can be used to create a more than competitive user experience, outperforming even their native counterpart [6] while there also have been less dynamic approaches [7].

This, of course, poses challenges on the programming language used, as it has to provide a high-level API to access all or at least the most commonly used functions of all target platforms (e.g. I/O operations, geolocation and taking pictures).

Another important issue concerning cross-platform apps seems to be a divergence in terms of user interface and its challenges [8, 9]. The vendors provide app style guides to the developers to enhance the user experience on their platform through common icon sets, navigation elements, app behavior et cetera. Figure 1 illustrates these differences regarding the navigation approaches in Android and iOS. But even in popular productive apps these style guides are not always applied correctly (e.g. “Deutsche Bahn Navigator”).

III. METHOD

To assess the maturity of the cross-platform app development process we developed a hybrid app prototype. There are several frameworks to leverage the interoperability of web standards. We decided to use the “Sencha Touch” framework because it combines the popular JavaScript framework “ExtJS” and vendor presets (icons etc.) to provide a native look and feel on all devices. The framework and its tools are rapidly evolving. There are “fallbacks” to provide older devices
with full functionality and it utilizes the model-view-controller (MVC) design pattern to foster separation of concerns, which is considered an important asset in enterprise environments.

The app should provide basic information on a conference: a list of speakers, a time schedule, a floor plan and an activity feed showing recent conference-related tweets (e.g. “#softwareresting”). These functions require typical external and internal data access and also a certain degree of user interaction. The objective is to have one codebase to deploy mobile applications to every major mobile operating system from.

The “packaging”, i.e. creating a container that works as native app on the target platform, is done using the framework’s command line interface (CLI). Other ways to wrap the app for multiple platforms include manual packaging via “PhoneGap” (based on Apache Cordova [11]) and the cloud-based service “PhoneGap Build” [12].

IV. RESULTS

The app has successfully been developed with a common text editor, the CLI and a web browser for testing. The application code has been packaged for Android using the CLI and the most recent Android software development kit (SDK). The cloud-based build was still prone to (undebuggable) errors.

The resulting app has successfully been tested on two Android-based smartphones (Galaxy Nexus and Samsung S3 Mini) and several desktop web browsers (Windows 7: Chrome 26, Internet Explorer 9/10, Firefox 11). It started up fast, ran smoothly (including remote data fetching) and the interaction felt “native” (see fig. 2).

V. DISCUSSION

While the framework of course could not decrease the effort of devising appropriate user interfaces for the different device types and orientations, it helped to create a codebase which is usable across multiple platforms.

We assume, that if a corresponding user interface is provided for every platform, end-users will not even notice the difference between native and hybrid apps. The MVC pattern makes this a solvable problem [13].

Though our prototype offers great performance, it does not look “native” yet. Without further customization it imitates Apple’s user interface style. Although the app can be themed according to the target platform to further incorporate the different OS’s looks and approaches to navigation and user guidance, there is still effort needed to make a hybrid app look and feel like a native app on all platforms. Although there are design commonalities, a common cross-platform style guide for mobile applications is as unlikely to occur as it would be useful.

VI. CONCLUSIONS

We learned that hybrid apps indeed are competitive to native apps. We support the opinion that hybrid apps will be pivotal in cross-platform development. That is why organizations should now begin to think about the benefits of cross-platform app development to safeguard their software development investments through a single codebase, more focused professionals and the ability to cover a significant share of the mobile market with their apps.
REFERENCES


