KATHOLIEKE UNIVERSITEIT
LEUVEN

Faculteit
Ingenieurswetenschappen

A comparative study of tools for cross-platform mobile application development

Michiel Staessen

Thesis voorgedragen tot het behalen
van de graad van Master of Science
in de ingenieurswetenschappen:
computerwetenschappen,
hoofdspecialisatie Software
engineering

Promotor:
prof. dr. ir. Erik Duval

Begeleider:
ir. Gonzalo Parra

Academiejaar 2012 – 2013

Preface

Michiel Staessen

Contents

Abstract

The `abstract` environment contains a more extensive overview of the work. But it should be limited to one page.

Chapter

1 Introduction

The mobile industry is without a doubt one of the most vibrant industries at the moment. It is characterized by rapid growth and intense competition which has led to fragmentation.

This chapter give an overview of the evolution in the mobile device landscape, explain the problem of fragmentation and how cross-platform tools (CPTs) can solve this problem.

1.1 The mobile device landscape

Mobile phones have been around since the nineties and before but the smartphone as we know it now has only been around since the (nearly simultaneous) introduction of the iPhone 3G and the HTC Dream in 2008. In the last five years, smartphone sales have grown tremendously. According to quarterly studies by Gartner[1] [5, 6, 7, 8, 9, 10, 11, 14, 15, 12, 16, 17, 18**? ?**], smartphone sales have grown 544% since the second quarter of 2008 (see Figure 1.1). Nowadays, smartphones are becoming ubiquitous and, in some regions like the United States, smartphone penetration has already reached more than 50% [23].

Figure 1.1 and Figure 1.2 also show that there is not a single major platform. The IDC[2] predicts that there will be at least three major platforms covering 90% of the worldwide smartphone market by 2016 [20].

A similar scenario is playing in the tablet industry. According to other studies by both Gartner [13, 19] and the IDC [21], tablets will continue to gain popularity and sales will be mainly driven by iPads and Android tablets (see Figure 1.3).

Even though both companies do not agree on which platform will be the biggest by 2016, they both predict there will be at least three major platforms; iOS, Android and Windows.

[1] Gartner, Inc. is the world's leading information technology research and advisory company [**?**].

[2] International Data Corporation is an American market research, analysis and advisory firm specializing in information technology, telecommunications and consumer technology.

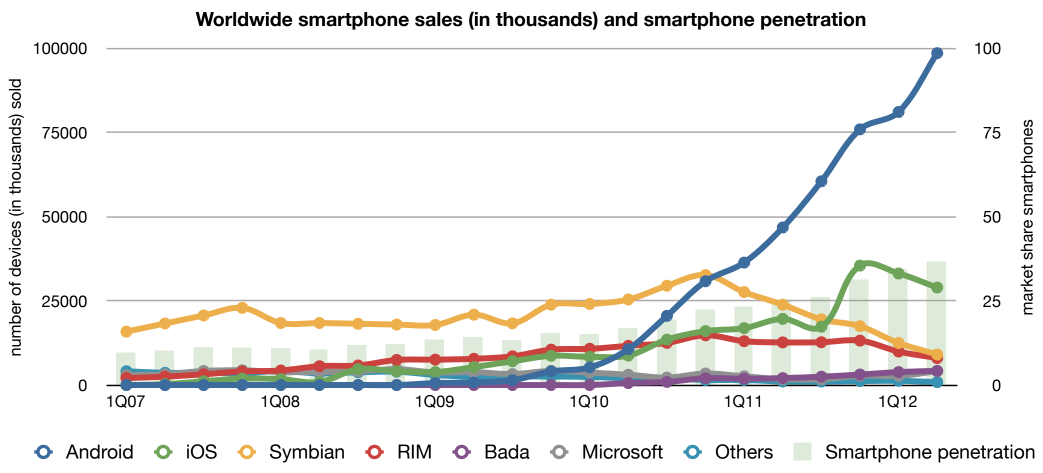


FIGURE 1.1: Growth of worldwide smartphone sales and smartphone penetration. Source: Gartner [5, 6, 7, 8, 9, 10, 11, 14, 15, 12, 16, 17, 18**?** **?**]

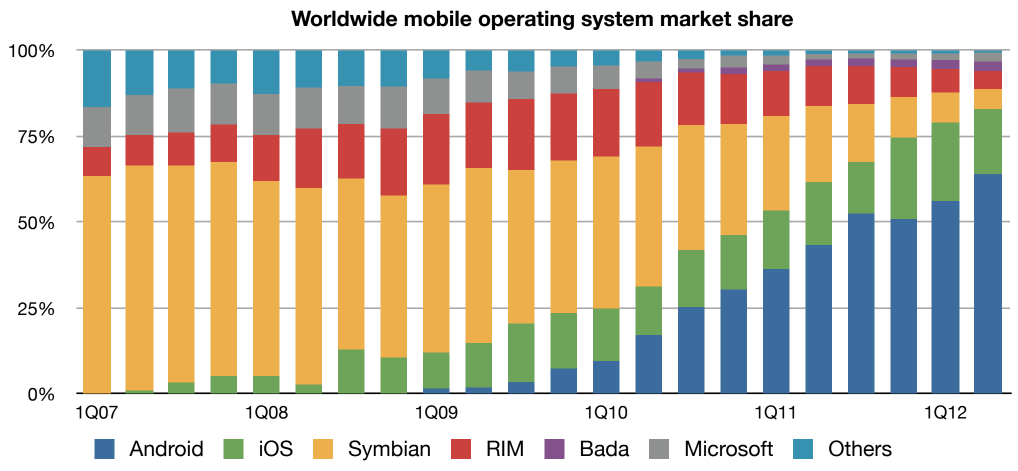
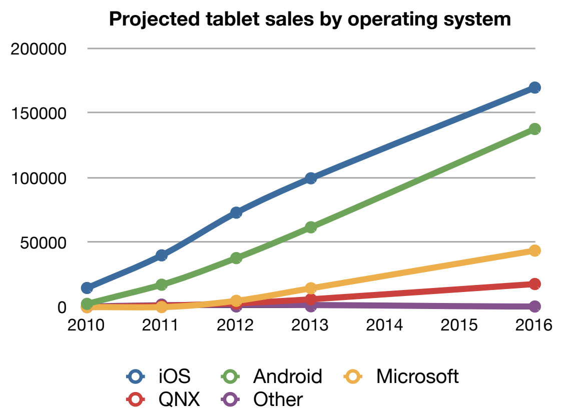


FIGURE 1.2: Growth of worldwide smartphone operating system market share. Source: Gartner [5, 6, 7, 8, 9, 10, 11, 14, 15, 12, 16, 17, 18**?** **?**]

1.2 The problem of fragmentation

The competition among mobile device manufacturers has led to fragmentation on many levels. For consumers, fragmentation is usually a good thing. The more different devices there are, the easier it is for a consumer to pick one that fits his needs.

For developers on the other hand, fragmentation is usually a bad thing because they will have to develop and test their applications on multiple devices to be able to guarantee



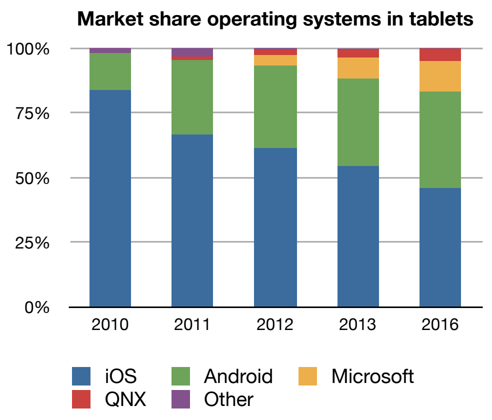


FIGURE 1.3: Growth of worldwide smartphone sales and smartphone penetration. Source: Gartner [13, 19]

the desired experience. This is expensive and time consuming.

From Figure 1.2 and Figure 1.3 it is already clear that the market is divided by operating system or platform. This is platform fragmentation. But even within a single platform, there is fragmentation and it is multi-dimensional [22].

1.2.1 Fragmentation on iOS

iDevices (Apple devices running iOS) are available in different shapes and sizes but there are only a limited number of device configurations. This is why Apple can manage fragmentation quite well.

For instance, iOS developers only have to support three *logical* display resolutions: 480 by 320 (for all iPhones and iPod Touches until the iPhone 5 and iPod Touch 5G), 568 by 320 (for the iPhone 5 and iPod Touch 5G) and 1024 by 768 (for all iPads). Note that even though retina displays have four times the number of physical pixels, the logical resolution remains unchanged but the pixel density is doubled.

Apple also provides good fitting mechanisms for applications that run on screens they are not designed for: on the iPad a “2x” button is available to make use of all the screen real estate when running iPhone apps and on the iPhone 5 old applications are centered on the screen.

Overall, fragmentation is rather low on iOS.

1.2.2 Fragmentation on Android

Android is an open source platform and vendors are allowed to tailor it for their devices. As a result, there are hundreds of Android devices but also hundreds of Android flavours with different user interfaces as well. This leads to fragmentation in all dimensions.

Maintenance of such Android flavours is expensive and for this reason, manufacturers do not often provide updates for their devices. This has led to the notorious runtime fragmentation among Android devices (see Figure 1.2.2).

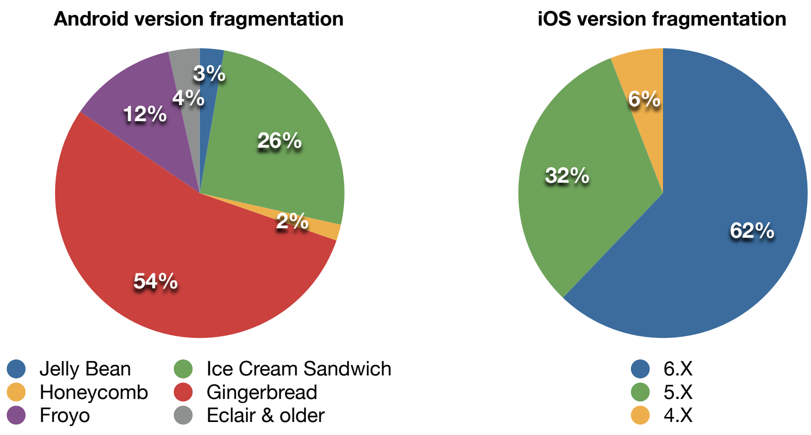


FIGURE 1.4: Runtime fragmentation for Android (data collected by Google during a 14-day period ending on November 1, 2012) [1] and iOS (based on the statistics of developer David Smith) [24].

These flavours also contribute to user interface fragmentation. Large vendors often ship their devices with a custom user interface to make their product more unique. This results in a non-uniform user interface across Android devices.

Because of the sheer number of Android devices, there is also a sheer number of display resolutions and pixel density configurations which leads to a high display fragmentation.

Overall, the Android platform is characterized by fragmentation in all dimensions.

1.3 Cross-Platform Tools

In the current economy, information is a company's most valuable asset and the rate at which information exchange takes place increases every day. Mobile Internet-enabled devices are a valuable resource for this purpose and, as a consequence, many companies want mobile applications for their businesses.

However, in an ever-changing and unpredictable industry like the mobile industry, it is very unwise to target a single platform. This could eventually lead to lock-in situations which companies try to avoid at all costs. Consequently, they will ask for a cross-platform solution.

Cross-Platform Tools (CPTs) can help solving this problem. They reduce entry barriers (access to new platforms) and exit barriers (lock-in) by allowing developers to create cross-platform applications from a single codebase [**?**].

CPTs try to solve three major problems [**?**]:

1. **Fragmentation** The fragmentation issues described above are a pain for every developer. They need to test their applications on a large number of devices in order to be able to guarantee the desired user experience. A CPT can help to identify platform quirks and can provide workarounds.
2. **Access to new platforms and screens** Targeting a new platform is often hard. A developer needs to learn yet another SDK and/or programming language in order to deliver a working application. Using a CPT will drastically decrease learning time for new platforms.
3. **Development inefficiency** Maintaining codebases for multiple platforms is a difficult task. When using a CPT, all code is contained within a single codebase and no time is lost while synchronizing features and other maintenance tasks across codebases.

Chapter

2 Literature Study

2.1 Multi-Criteria Decision Making (MCDM)

Finding the right software package is often a daunting task. In order to suit the end-user's needs, the software should meet a large number of – sometimes conflicting – requirements and will result in making important trade-offs. Because of the these characteristics, software selection can be modeled as a Multiple-Criteria Decision Making (MCDM) problem [**?** **?**].

There are two categories of MCDM problems: Multiple-Attribute Decision Making (MADM) problems and Multi-Objective Decision Making (MODM) problems. The first category involves sorting and ranking of a limited number of available alternatives, based on a number of decision criteria. In the latter category, there are no alternatives specified beforehand and the number of alternatives is effectively infinite [**?**].

The software selection process belongs to the category of MADM problems. Their goal is to find the best alternative in a set of alternatives and at the same to time create a ranking of all these available alternatives.

There are a plethora of solution methods for the MADM problem. The following subsections will describe the most frequently used methods in literature, together with their advantages and disadvantages.

2.1.1 Arbitrary scoring models

There are a number of arbitrary scoring models but they all have one thing in common: for each criterion, the values are translated to a numerical score. In some cases, this score can be derived from the value of the criterion itself (e.g. speed, age, cost, ...). In other cases, a mapping is provided (e.g. in order to obtain a score s, at least features f_1, f_2 and f_3 should be supported).

Different methods are available to select potential candidates using these scores [**?**]:

FIGURE 2.1:

- When the “dominance” method is used, one alternative should clearly outperforms the other alternatives for at least one criterion.
- When the “maximin” method is used, the final score of each alternative is equal to the lowest score of all criteria for this alternative.
- When the “maximax” method is used, the final score of each alternative is equal to the highest score of all criteria for this alternative.
- When the “conjunctive” method is used, an alternative should exceed certain thresholds for *all* criteria.
- When the “disjunctive” method is used, an alternative should exceed certain thresholds for *at least one* criterion.

Additionally, the importance of the criteria can be accounted for by assigning weights. The final score can be calculated as the weighted sum, weighted product or weighted average.

The strength of these methods is that they are the most easy to use. However, scores and weights are assigned arbitrarily and might get tough when there are a lot of criteria. Also, not all criteria are suitable for conversion into a numerical scale [**?**].

2.1.2 Analytic Hierarchy Process (AHP)

The Analytic Hierarchy Process (AHP) was originally developed by Thomas L. Saaty and offers a mathematical solution for MADM problems. The method can even compensate for inconsistency and deal with non-numerical scales by using pairwise comparisons.

In the first stage, the MADM problem is decomposed into a tree of criteria. A criterion can consist of a number of sub-criteria or it can contain the values of all alternatives for this criterion. This results in a criterion tree, similar to figure 2.1.

In the next stage, the weights of each subtree (including the values of a criterion) will be calculated using a pair-wise comparison. For every combination of criteria (c_i, c_j), define the importance of c_i in terms of c_j and aggregate the preference scores in a matrix W. For example, if c_i is twice as important as c_j, then $c_i = 2 \times c_j$, and write $w_{i,j} = 2$ in the resulting matrix W. Conversely, $c_j = \frac{1}{2} \times c_i$ and $w_{j,i} = \frac{1}{2}$. The values on the diagonal of W are all equal to 1. The weights of c_i can now be calculated as the eigenvalues of W. This stage results in a weight distribution for the criteria tree.

In the last stage, every alternative can be scored and ranked using this the weights obtained from the previous stage.

The strengths of the are that it (1) enables decision makers to structure a problem into a hierarchy, (2) that is provides a powerful tool for handling both quantitative and qualitative multi-criteria decision making problems and (3) that this system can deal with inconsistency (on certain levels) [**?**], [].

The weakness of AHP is that it is a time consuming method due to the large number of pair-wise comparisons. Also, the ordering may change entirely when other alternatives are taken into account.

2.1.3 Fuzzy MCDM

2.2 Software evaluation methodology

Based on their literature review [**?**], the authors have proposed a generic, six-stage methodology for the selection of software packages [**?**].

1. **Define the selection criteria.** In the first stage, the evaluator defines the essential requirements for the software. If a certain software package does not meet a selection criterion, it is not a considered to be a suitable candidate and it should not be considered for the evaluation.
2. **Look for potential candidates.** During the next step, the evaluator searches for potential candidates. This step will result in a list of potential candidates.
3. **List the suitable alternatives.** In this step, the evaluator will use the requirements from stage 1 to filter the list obtained from the previous stage.
4. **Define the evaluation criteria.** In this stage, the evaluator has to define the evaluation criteria, arrange them in a hierarchy and define the value scales for each criterion.
5. **Evaluate the alternatives.** During this phase, the evaluator will make a detailed comparison of the alternatives using the criteria obtained from the previous stage. A methodology like AHP can be used in this stage.
6. **Select the best alternative.** In this final step, all alternatives are ranked using the comparison results from the previous stage. Now, the best alternative can be chosen from the list of candidates. In general, a number of software packages will be taken into consideration and a selection will be made after additional steps (like for instance a cost-benefit analysis and/or contract negotiations with the vendor).

In the original paper [**?**], the authors also suggest to include an additional evaluation stage after the selected packages has been implemented and integrated. During that stage, one should verify that the selected package does indeed meet the requirements.

2.3 Strategies for cross platform development

There are already a number of paradigms for cross platform mobile application development [4]. This section presents an overview of the available strategies by comparing different aspects: performance, look and feel, platform access, programming languages, development cost and distribution.

2.3.1 Native App

A native app is an application that is specifically designed to run on a particular platform. It is the default approach to develop applications for mobile devices. Figure 2.2 shows an illustration of the overall architecture of such an app.

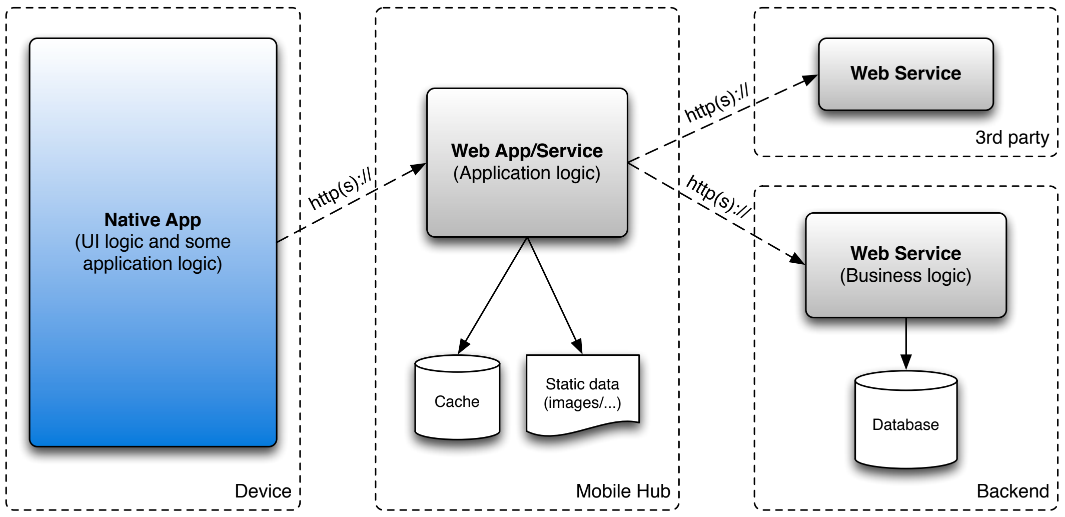


FIGURE 2.2: Overall architecture of a native app.

Native apps are developed with the supplied SDK. Developers will need to get acquainted with the programming language used by said SDK but in return they will get full access to the platform and its features. As a result, the best performance can be obtained with this kind of app.

For the user interface, developers can use lots of interface elements such that they can present a familiar look and feel to the end user.

Native apps can be easily distributed through an online marketplace like for instance the App Store or Google Play.

Because native apps are designed to run on one platform only, this development strategy is not very well suited for cross platform development. If an application should run on multiple platforms, it has to be developed for each platform separately. This is costly.

2.3.2 Web App

Web apps are websites that are optimized for mobile browsers. Since every platform comes with a browser, this is the easiest way to get an application running on all platforms. An overview of the overall architecture for this kind of app is given in Figure 2.3.

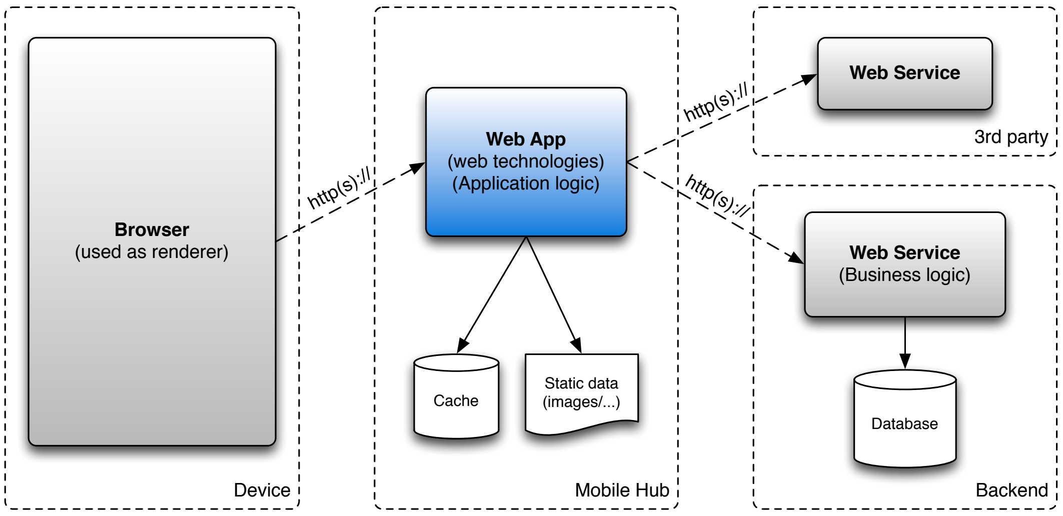


FIGURE 2.3: Overall architecture of a web app.

Web apps are not nearly as powerful as native apps. First of all, the application is not stored on the device. Web apps require an active internet connection which cannot always be guaranteed. Second, they are built with web technologies like HTML, CSS and JavaScript, which have to be interpreted by the browser at runtime. Third, web apps cannot access the system which means they cannot make use of the many unique features of a mobile device.

With HTML5, web apps can get more powerful. They will be able to access device features, like the camera and other sensors [3]. They will not even require an active internet connection because they can be cached on the device. However, HTML5 is still a draft and a lot of mobile browsers lack proper HTML5 support.

From a user interface perspective, web apps can be a problem as well.

Web apps are distributed easily: the only requirement is a valid URL. Web apps cannot be installed on the device though, but there are workarounds using Web Clips on iOS [2] and bookmarks on Android.

2.3.3 Hybrid App

Hybrid applications are the logical next step, combining native apps and web apps. The actual application is a web site, embedded in a web view, part of a native wrapper. The embedded website can access (parts of) the system through a bridge. An overview of the overall architecture is shown in Figure 2.4.

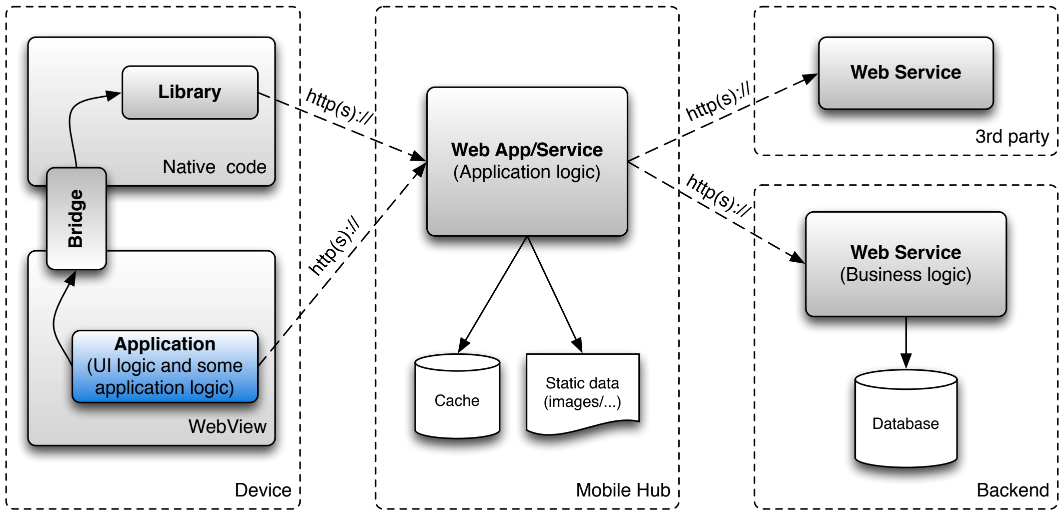


FIGURE 2.4: Overall architecture of a hybrid app.

Hybrid apps are part native app, part web app. Performance will be similar to web apps but some parts can be optimized by using native code. The websites inside the hybrid app are also much more powerful because they can access many device features that aren't available in HTML(5) through the bridge.

When it comes to the user interface, hybrid apps suffer from the same problem as web apps.

Because hybrid apps are wrapped in a native container, they can be distributed just like native applications, through online marketplaces.

2.3.4 Interpreted App

In an interpreted app, instructions in some language are translated to native instructions at runtime. Figure 2.5 shows the overall architecture of an interpreted app.

Performance of interpreted apps depends on the interpreter and interpreted language but is better than web apps on average, though not as good as native apps.

In an interpreted app, the user interface description is interpreted and rendered on the device using native interface elements. An interpreted app will have a familiar look and feel.

From the outside, interpreted apps – just like hybrid apps – look like native apps and can be distributed through online marketplaces.

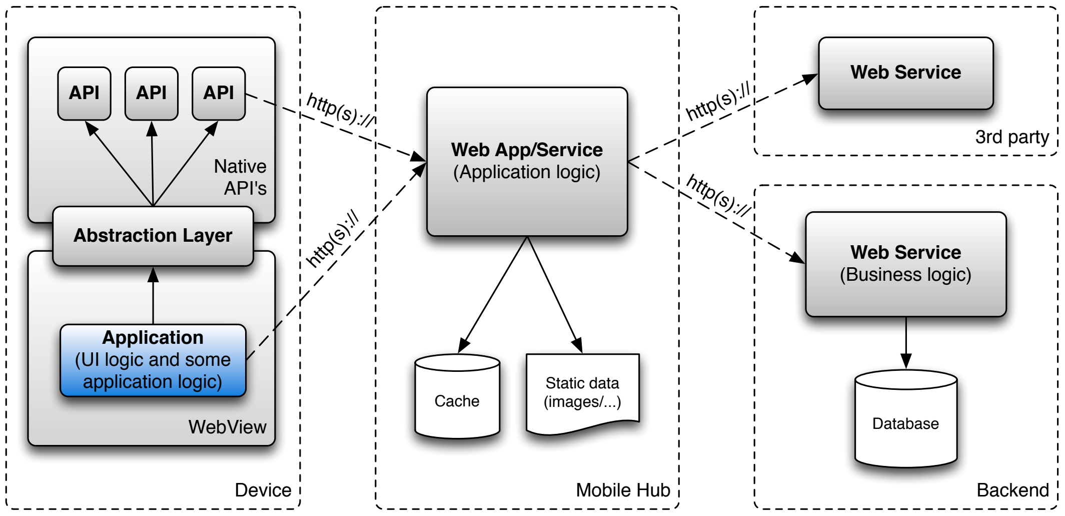


FIGURE 2.5: Overall architecture of an interpreted app.

2.3.5 Cross Compiling

Instead of translating instructions at runtime, one could translate instructions at compile time. The process is called cross compiling and the result is a truly native app. The overall architecture is sketched in Figure 2.6.

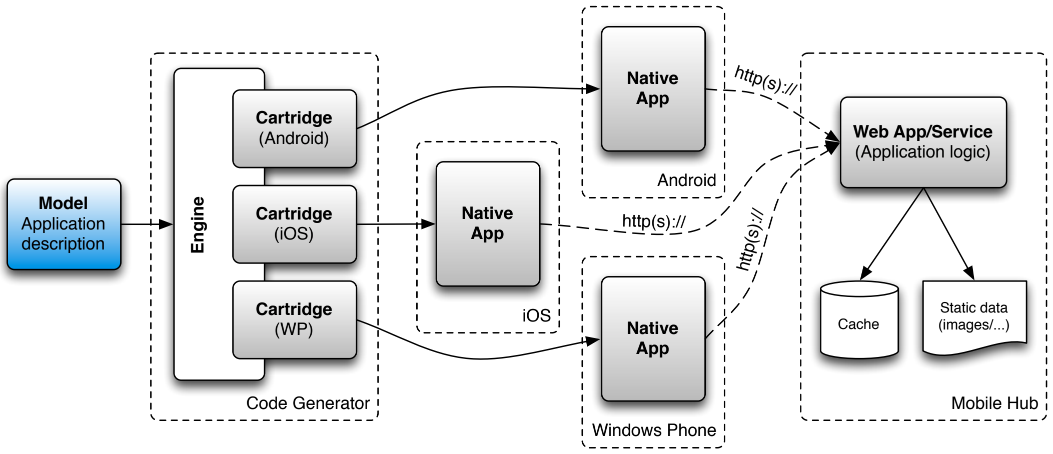


FIGURE 2.6: Overall architecture for cross compiled apps.

Summary

Table 2.1 summarizes the results of the discussed strategies. It is important to note that there is no universal strategy that fits all use cases. A strategy must be chosen carefully, taking into account the client's wishes.

	Native	Web	Hybrid	Interpreted	Cross Compiled
Performance	high	low	rather low	average	high
Platform Access	✓	× / ✓	✓	✓	✓
Look & Feel	native	non-native	non-native	native	native
Distribution	marketplace	URL	marketplace	marketplace	marketplace
Development cost	high	rather low	average	average	average

TABLE 2.1: Summary of cross platform mobile application development strategies.

Summary

Chapter 3

Methodology

Based on the insights gained from the literature, this chapter describes the research methodology used to evaluate the selected cross-platform tools.

In [**?**], the authors present a generic stage-based methodology for the selection of software packages.

1. Determine the need for purchasing the system and preliminary investigation of the availability of suitable candidates.
2. Short candidate listing
3. Eliminate

3.1 Define selection criteria

3.2 Select

3.3 Define evaluation criteria

3.4 Evaluation

3.5

Chapter

4

Studied Tools

Chapter

5

Comparison Results

Chapter

6

Conclusion

Appendices

Bibliography

[1] Android. Platform Versions, Screen Sizes and Densities and Open GL Version. URL http://developer.android.com/about/dashboards/index.html. Accessed on 02/11/2012.

[2] Apple. Configuring Web Applications, 2010. URL http://developer.apple.com/library/safari/#documentation/AppleApplications/Reference/SafariWebContent/ConfiguringWebApplications/ConfiguringWebApplications.html#//apple_ref/doc/uid/TP40002051-CH3-SW4. Accessed on 02/11/2012.

[3] Maximiliano Firtman. Mobile HTML5, 2012. URL http://mobilehtml5.org/. Accessed on 02/11/2012.

[4] Peter Friese. Cross platform mobile development, 2012. URL http://www.slideshare.net/peterfriese/cross-platform-mobile-development-11239246. Accessed on 02/11/2012.

[5] Gartner. Gartner Says Worldwide Smartphone Sales Grew 16 Per Cent in Second Quarter of 2008, 2008. URL http://www.gartner.com/it/page.jsp?id=754112. Accessed on 02/11/2012.

[6] Gartner. , 2008. URL http://www.gartner.com/it/page.jsp?id=827912. Accessed on 02/11/2012.

[7] Gartner. Gartner Says Worldwide Smartphone Sales Reached Its Lowest Growth Rate With 3.7 Per Cent Increase in Fourth Quarter of 2008, 2009. URL http://www.gartner.com/it/page.jsp?id=910112. Accessed on 02/11/2012.

[8] Gartner. Gartner Says Worldwide Mobile Phone Sales Grew 17 Per Cent in First Quarter 2010, 2010. URL http://www.gartner.com/it/page.jsp?id=1372013. Accessed on 02/11/2012.

[9] Gartner. Gartner Says Worldwide Mobile Device Sales Grew 13.8 Percent in Second Quarter of 2010, But Competition Drove Prices Down, 2010. URL http://www.gartner.com/it/page.jsp?id=1421013. Accessed on 02/11/2012.

[10] Gartner. Gartner Says Worldwide Mobile Phone Sales Grew 35 Percent in Third Quarter 2010; Smartphone Sales Increased 96 Percent, 2010. URL http://www.gartner.com/it/page.jsp?id=1466313. Accessed on 02/11/2012.

[11] Gartner. Gartner Says Worldwide Mobile Device Sales to End Users Reached 1.6 Billion Units in 2010; Smartphone Sales Grew 72 Percent in 2010, 2011. URL http://www.gartner.com/it/page.jsp?id=1543014. Accessed on 02/11/2012.

[12] Gartner. Gartner Says Sales of Mobile Devices Grew 5.6 Percent in Third Quarter of 2011; Smartphone Sales Increased 42 Percent, 2011. URL http://www.gartner.com/it/page.jsp?id=1848514. Accessed on 02/11/2012.

[13] Gartner. Gartner Says Apple iOS to Dominate the Media Tablet Market Through 2015, Owning More Than Half of It for the Next Three Years, 2011. URL http://www.gartner.com/it/page.jsp?id=1626414. Accessed on 02/11/2012.

[14] Gartner. Gartner Says 428 Million Mobile Communication Devices Sold Worldwide in First Quarter 2011, a 19 Percent Increase Year-on-Year, 2012. URL http://www.gartner.com/it/page.jsp?id=1689814. Accessed on 02/11/2012.

[15] Gartner. Gartner Says Sales of Mobile Devices in Second Quarter of 2011 Grew 16.5 Percent Year-on-Year; Smartphone Sales Grew 74 Percent, 2012. URL http://www.gartner.com/it/page.jsp?id=1764714. Accessed on 02/11/2012.

[16] Gartner. Gartner Says Worldwide Smartphone Sales Soared in Fourth Quarter of 2011 With 47 Percent Growth, 2012. URL http://www.gartner.com/it/page.jsp?id=1924314. Accessed on 02/11/2012.

[17] Gartner. Gartner Says Worldwide Sales of Mobile Phones Declined 2 Percent in First Quarter of 2012; Previous Year-over-Year Decline Occurred in Second Quarter of 2009, 2012. URL http://www.gartner.com/it/page.jsp?id=2017015. Accessed on 02/11/2012.

[18] Gartner. Gartner Says Worldwide Sales of Mobile Phones Declined 2.3 Percent in Second Quarter of 2012, 2012. URL http://www.gartner.com/it/page.jsp?id=2120015. Accessed on 02/11/2012.

[19] Gartner. Gartner Says Worldwide Media Tablets Sales to Reach 119 Million Units in 2012, 2012. URL http://www.gartner.com/it/page.jsp?id=1980115. Accessed on 02/11/2012.

[20] IDC. Android Expected to Reach Its Peak This Year as Mobile Phone Shipments Slow, According to IDC, 2012. URL http://www.idc.com/getdoc.jsp?containerId=prUS23523812. Accessed on 2/11/2012.

[21] IDC. IDC Raises Its Worldwide Tablet Forecast on Continued Strong Demand and Forthcoming New Product Launches, 2012. URL http://www.idc.com/getdoc.jsp?containerId=prUS23696912. Accessed on 2/11/2012.

[22] Charlie Kindel. Fragmentation Is Not The End of Android, 2012. URL http://ceklog.kindel.com/2012/01/14/fragmentation-is-not-the-end-of-android/. Accessed on 02/11/2012.

[23] Nielsen. Smartphones Account for Half of all Mobile Phones, Dominate New Phone Purchases in the US, 2012. URL http://blog.nielsen.com/nielsenwire/online_mobile/smartphones-account-for-half-of-all-mobile-phones-dominate-new-phone-purchases-in-
Accessed on 02/11/2012.

[24] David Smith. iOS Version Stats. URL http://david-smith.org/iosversionstats/. Accessed on 02/11/2012.

Fiche masterproef

Student: Michiel Staessen

Titel: A comparative study of tools for cross-platform mobile application development

Nederlandse titel: "Een vergelijkende studie van cross-platform tools voor het ontwikkelen van mobiele applicaties"

UDC:

Korte inhoud:

Developing mobile applications (apps) for multiple platforms is an expensive and time consuming process. Therefore, many companies are seeking refuge in Cross-Platform Tools (CPTs) for the development of their apps. In cooperation with CapGemini, this thesis presents a comparison of two such cross-platform tools: Apache Cordova and Motorola Rhodes. Both tools are compared with each other and with native development. The comparison is based on a proof-of-concept application which should work on both smartphones and tablets with respect to both iOS and Android.

Thesis voorgedragen tot het behalen van de graad van Master of Science in de ingenieurswetenschappen: computerwetenschappen, hoofdspecialisatie Software engineering

Promotor: prof. dr. ir. Erik Duval

Assessor:

Begeleider: ir. Gonzalo Parra