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

## 1.1 The mobile device landscape

### 1.1.1 Smartphones

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, 19, 21], smartphone sales have grown 544% since the second quarter of 2008 (see Figure 1.1). Smartphones are becoming ubiquitous and in some regions like the United States, smartphone penetration has already reached more than 50% [24].

Operating systems are heavily subjected to network effects, i.e. the value of a platform is proportional to the number of people using it. This makes it hard for new platforms to gain traction which is visible in Figure 1.1 and Figure 1.2. Android and iOS are currently the most popular platforms and other platforms are either in decline (like Symbian and Blackberry, formerly RIM) or have a hard time getting traction (like Windows Phone).

However, there is so single major platform. The IDC[2] even predicts that Windows Phone will gain a significant market share by 2016 and that 90% of the worldwide smartphone market will then be covered by Android, iOS and Windows Phone [**?** ]. Therefore it is very reasonable to assume that there will always be more than one major platform.

[1] Gartner is an American firm, specialized in information technology research [**?** ].

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

TODO Update Graph

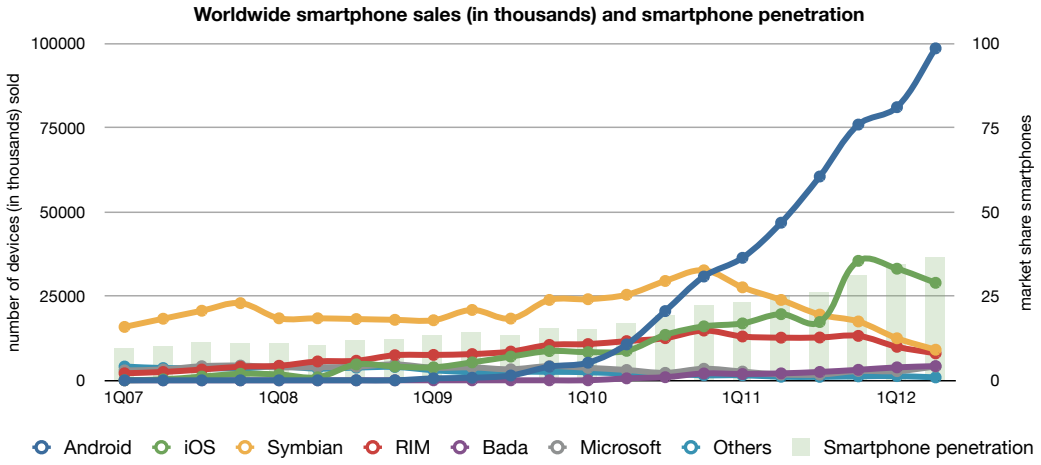


FIGURE 1.1: Growth of worldwide smartphone sales and smartphone penetration. Source: Gartner [5, 6, 7, 8, 9, 10, 11, 14, 15, 12, 16, 17, 18, 19, 21]

TODO Update Graph

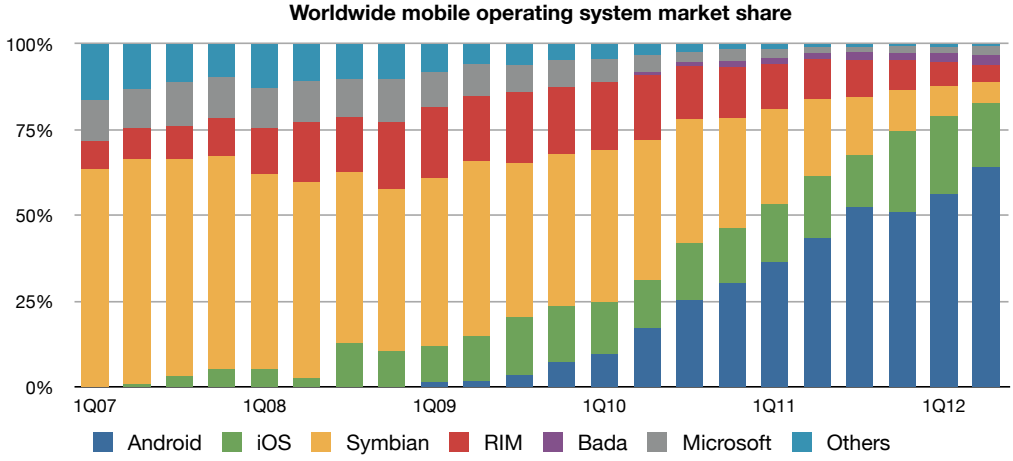


FIGURE 1.2: Growth of worldwide smartphone operating system market share. Source: Gartner [5, 6, 7, 8, 9, 10, 11, 14, 15, 12, 16, 17, 18, 19, 21]

### 1.1.2 Tablets

A similar scenario is playing out in the tablet industry. According to other studies by both Gartner [13, 20] and the IDC [**?** ], 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 serve the most devices,

TODO Check whether new predictions are available

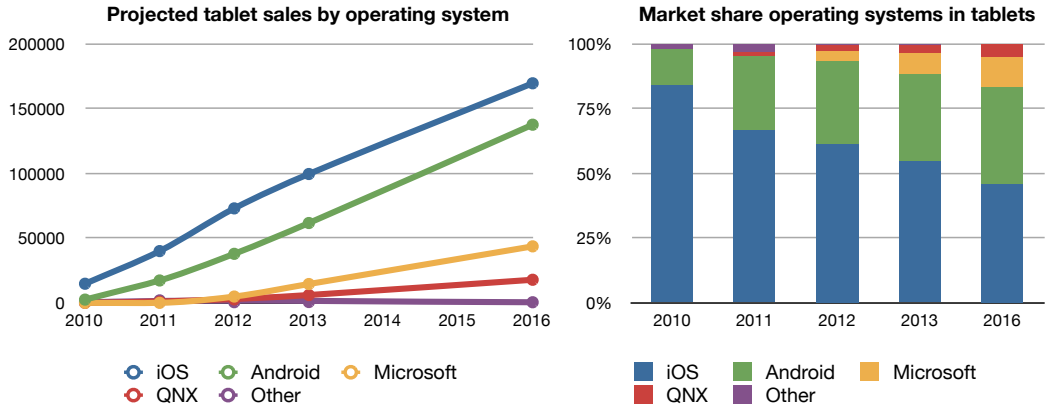


FIGURE 1.3: Prediction of worldwide tablet sales and market share. Source: Gartner [13, 20]

they both predict there will be at least three major platforms: iOS, Android and Windows.

## 1.2 The problem of fragmentation

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

For developers on the other hand, fragmentation is usually a bad thing. They will have to develop and test their applications on multiple devices to be able to guarantee the desired experience. This is expensive and time consuming.

Fragmentation is a multi-dimensional problem. From Figure 1.2 and Figure 1.3 it is already clear that the operating system or platform is one obvious dimension; this is called platform fragmentation. Other dimensions include device configuration, runtime, user interface, etc.

The next subsections will discuss fragmentation issues within iOS and Android.

### 1.2.1 Fragmentation within iOS

Devices running iOS (commonly referred to as iDevices) are available in different shapes and sizes. Yet, there are only a limited number of (similar) device configurations in two form factors: iPhone (covering iPhone and iPod Touch devices) and iPad. Apple uses these form factors to distinguish three kinds of applications: iPhone apps (designed to run on iPhone form factor only), iPad apps (designed to run on iPad only) and universal apps (designed to run on both form factors).

TODO describe runtime fragmentation

Overall, Apple can manage fragmentation quite well and fragmentation is rather low within iOS.

### 1.2.2 Fragmentation within Android

Android is an open source platform, developed by Google. Manufacturers are allowed to tailor it for their devices and many of them have grabbed this opportunity to differentiate their products. On april 24 2013, the Google Play store officially supported 2827 device configurations, distributed across 59 manufacturers. The ability to alter the operating system has largely contributed to the success of Android.

There is however virtually no incentive for device manufacturers to maintain their Android flavours and as a result they do not often provide updates for their devices. This has led to the notorious runtime fragmentation among Android devices (see Figure 1.4).

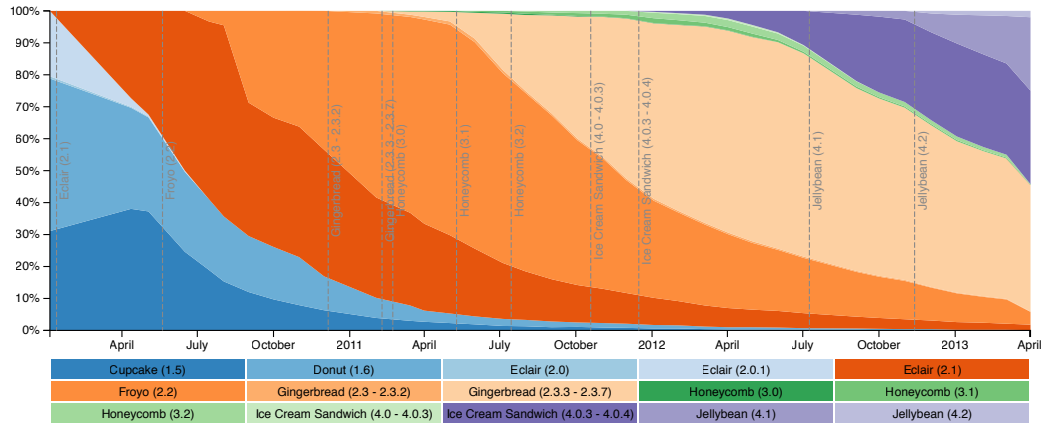


FIGURE 1.4: Historical runtime fragmentation on Android. The data from this graph was aggregated from [1] using the Internet Archive (http://archive.org).

These flavours also contribute to user interface fragmentation because manufacturers ship their devices with a custom user interface to differentiate their product.

Because of the sheer number of Android devices, there is also a large number of CPUs, screen sizes, display resolutions, pixel density configurations and many more. All these differences contribute to device fragmentation.

Long story short, the Android platform is characterized by fragmentation in all dimensions.

## 1.3 Cross-platform tools to the rescue

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 [**?** ].

Cross-Platform Tools 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 or screen (for instance, television sets or car consoles) is often hard. Developers need to learn yet another SDK and/or programming language in order to deliver applications for the platform or screen. Using a CPT can drastically reduce the effort needed to target a new platform or screen.
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.

TODO Finish chapter, which can be read where + define scope, make boundaries

Chapter

# 2 Literature Study

TODO Introductory text

## 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 [22, 23].

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.

TODO General remark for all methods: Separate costs from benefits, allowing to make a cost-benefit analysis afterwards.

### 2.1.1 Arbitrary scoring models

TODO Question: Is this an appropriate name?

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

TODO Create graphic

FIGURE 2.1:

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 [**?** ]:

- 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 [22].

### 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) [22], []. TODO Fetch an AHP book from the library to cite here

TODO Add example

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

TODO Write section

Fuzzy logic was introduced in the mid '60s by Zadeh and was later applied to MCDM problems in the late '70s. Fuzzy logic allows to reason with imprecisely specified criteria like very high performance, low performance, etc. In classical approaches like those described above imprecision is dealt with first. When using the fuzzy approach, these imprecisions are only dealt with in the end, and only if necessary.

Fuzzy MCDM is based on three concepts [**?** ]:

1. The set of alternatives is a *fuzzy set*, i.e. the alternatives aren't precisely defined. The decision will thus be taken in a *fuzzy environment*.
2. The decision is based on an aggregation of fuzzy preferences among alternatives.
3. The decision is based on an evaluation of alternatives using a linguistic description.

More concrete, let $A = \{a_1, \ldots, a_n\}$ be a finite set of given alternatives and let there be $m$ imprecisely defined criteria. Every alternative fulfills a criterion in a certain degree or does not fulfill it at all. In other words, for each criterion $G_j$, $j = 1, \ldots, m$ there is a set $G_j \subset A$ containing the alternatives fulfilling the criterion and the degree in which an alternative $a_i$ fulfills $G_j$ is expressed by the membership degree $G_j(a_i)$.

In order to decide the MCDM problem, constraints must be defined for each criterion. These constraints are specified as fuzzy sets as well meaning that a constraint $C_j$ is associated with a set $C_j \subset A$ containing the alternatives fulfilling $C_j$ in a certain degree. Again, a membership degree $C_j(a_i)$ is given.

Consequently, there are two fuzzy sets for every $j, j = 1,\ldots,m$, namely $G_j$ and $C_j$. The decision can now be obtained by composition and is given by the fuzzy set

$$D = (G_1 \alpha C_1)\beta \ldots \beta (G_m \alpha C_m) \tag{2.1}$$

where $\alpha$ and $\beta$ are suitable fuzzy set operations. The alternative $a_k$ with the highest membership degree $D(a_k)$ is the best alternative.

TODO paraphrazed same source in previous three paragraphs, need a reference on each?

TODO Add an example: For example, when buying a house these criteria could be *nice neighbourhood, great parking facilities, reasonable price*, etc.

## 2.2 Software evaluation methodology

Based on their literature review [22], the authors have proposed a generic, six-stage methodology for the selection of software packages [23].

1. **Define 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 further evaluation.

2. **Identify potential candidates.** During the next step, the evaluator searches for potential candidates. This step will result in a list of potential candidates.

3. **List selected alternatives.** In this step, the evaluator will use the requirements from stage 1 to filter the list obtained from the previous stage.

4. **Define 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 selected 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 most suitable 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 [22], 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 Mobile Architectures

TODO Explain Mobile Hub/Mobile Orchestrator

TODO discuss cost for each architecture

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.

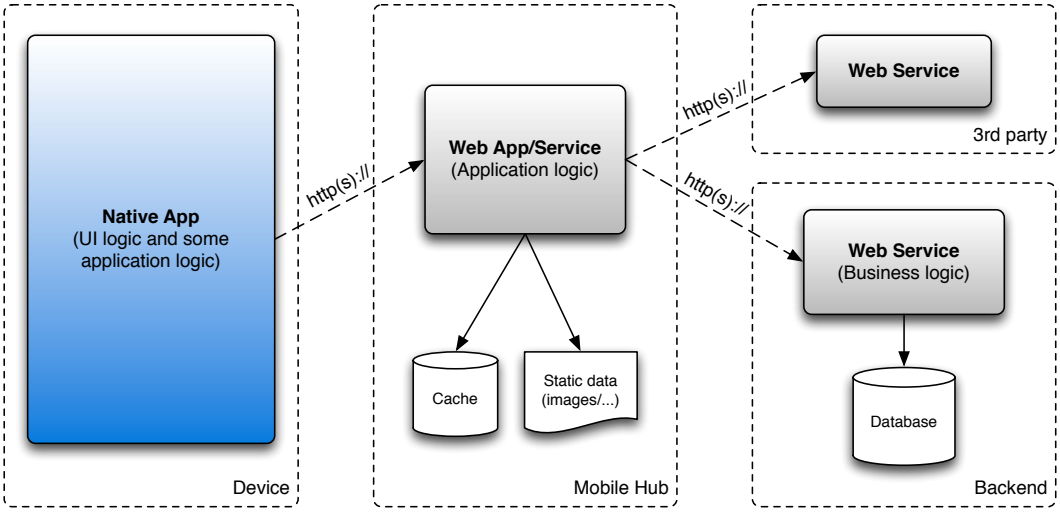


FIGURE 2.2: Overall architecture of a native app.

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.

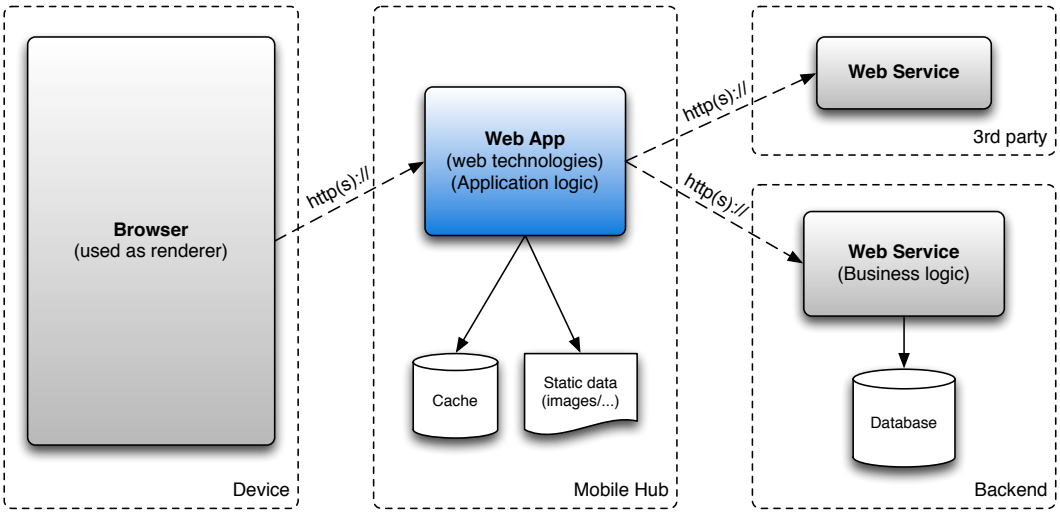


FIGURE 2.3: Overall architecture of a web app.

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. TODO complete paragraph

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.

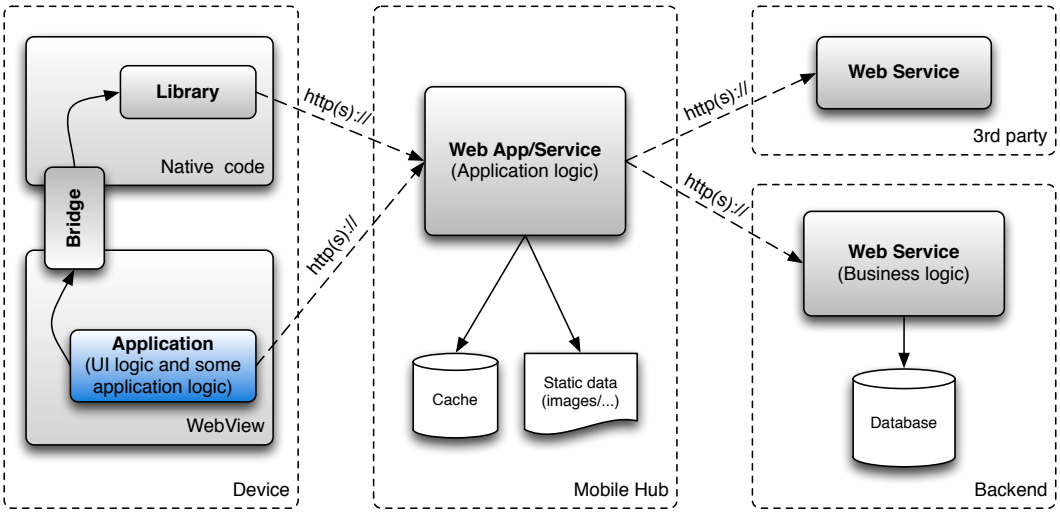


FIGURE 2.4: Overall architecture of a hybrid app.

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.

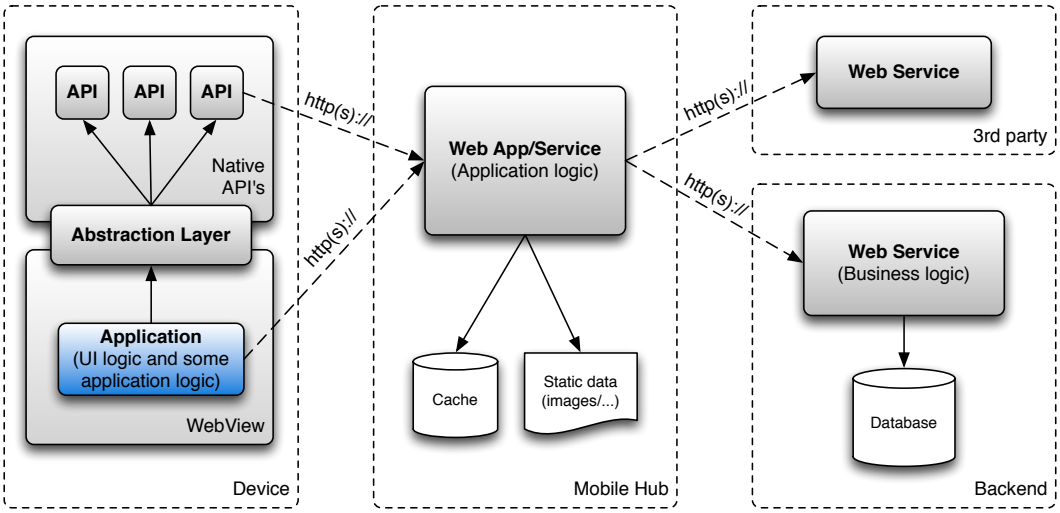


FIGURE 2.5: Overall architecture of an interpreted app.

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.

### 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.

#### 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.

## Summary

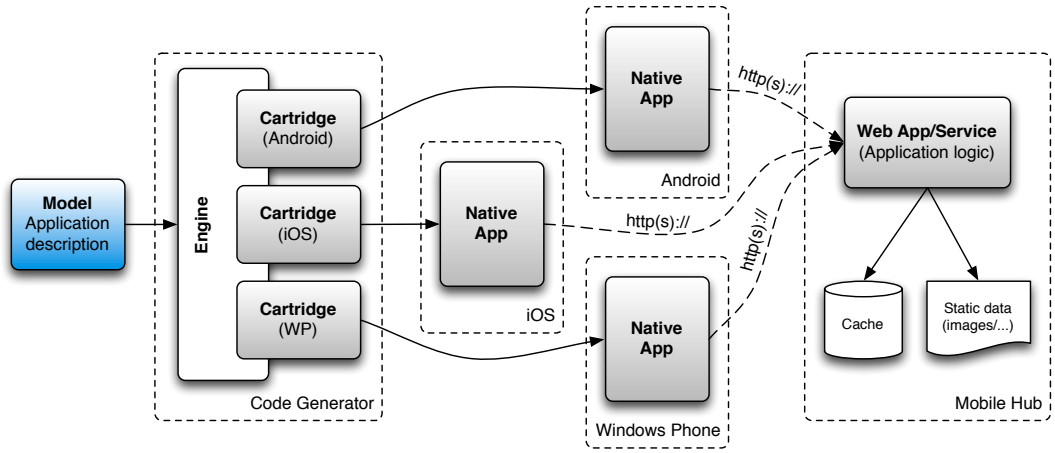


FIGURE 2.6: Overall architecture for cross compiled apps.

| | 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.

Chapter

# 3 Methodology

This chapter describes the methodology used to compare and rank the studied cross-platform tools. The methodology is based on the 6-step approach which is presented in [23] and studied in the literature study (see chapter 2). To recap quickly, these are the six steps:

1. Define selection criteria
2. Identify potential candidates
3. List selected alternatives
4. Define evaluation criteria
5. Evaluate selected alternatives
6. Select the most suitable alternative

Every step will be further expanded in the following sections.

## 3.1 Define selection criteria

In this first step, the selection criteria for the tool are recorded. These requirements will be used later (in step 3) to filter a list of potential candidates.

As the research presented in this thesis is conducted on behalf of CapGemini, the selection criteria are determined by their consultants and were recorded during the kick-off meeting on October 15, 2012.

In order to qualify as a viable cross-platform tools, it has to meet the following requirements:

- **It *must* produce "native" Android *and* iOS applications.** CapGemini focusses mainly on Android and iOS because their clients mainly focus on these platforms. Support for other platforms is desirable but not not a necessity.
- **It *must* be able to produce *both* tablet and smartphone applications, preferably from the same codebase.** Some clients want tablet applications, some clients want smartphone apps, some clients want both.

This list of essential requirements is extended with additional requirements. These are not essential as they can be circumvented in some way though they will generally result in higher productivity.

- **It *should* be usable to create enterprise applications with.** CapGemini specializes in the development of data-driven enterprise applications. Such applications usually contain a lot of forms and don't require high performance graphics (like for instance in 3D games). Even though it is possible to develop an enterprise application on top of a 3D engine, it will probably not result in good productivity.
- **It *should* have a certain degree of maturity** Ideally, the tool should be maintained for as long applications are created with it (and maintained).
- **It *should* have good support, provided by either the vendor or by the community.** In case of a problem, there should be a way to get support.

## 3.2 Identify potential candidates

In this stage, the evaluator tries to identify as much potential candidates as possible. These candidates do not necessarily have to meet the requirements from the previous stage as this is merely a discovery phase. The result of this stage will be a list of potential candidates.

Discovery of cross-platform tools has already been done extensively by VisionMobile. The latest cross-platform tools report [] contains a list of 100 cross-platform tools they tracked as part of their research. Because additional internet searches did not reveal new tools, this list is used as output of this stage.

## 3.3 List selected alternatives

It is now time to make a first selection. In this phase, the candidates obtained from the previous stage are filtered with the selection criteria from the first stage. The result of this stage is a list of alternatives worth investigating.

The list of tools from the previous stage contains a plethora of tools. Using the requirements from stage 1, a large number of tools can be left out:

- tools that do not produce native applications for Android and iOS;
- tools that do not produce tablet and smartphone applications;
- special-purpose tools that are not well suited for the intended use, e.g. specialized 3D engines;
- tools with an uncertain future, e.g. Flash-based systems or cutting-edge tools;
- tools that do not offer good support.

From the 100 tools listed, only 7 tools remain. They are listed in Table 3.1.

| **Name** | **Architecture** | **Type** |
|---|---|---|
| Apache Cordova | Hybrid | Open Source |
| Appcelerator Titanium | Interpreted | Open Source |
| Motorola Rhodes | Interpreted | Open Source |
| Trigger.io | Hybrid | Commercial |
| MoSync | Cross-Compiled + Hybrid | Open Source |
| Kony | Hybrid | Commercial |
| Xamarin | Cross-compiled | Commercial |

TABLE 3.1: The remaining tools after application of the selection criteria.

In this thesis, two tools will be compared with each other and with the native development kits for Android and iOS. The selected tools are Apache Cordova (formerly known as PhoneGap) and Motorola Rhodes.

Cordova was chosen because of its popularity among developers and Motorola Rhodes was chosen because it focusses on enterprise applications.

Appcelerator Titanium and Xamarin are studied by another student with the same research topic. Trigger.io, however kind to provide an extended free trial, was not chosen because it is very similar to Apache Cordova. Kony, however targeting enterprise applications, did not provide a free trial.

TODO Pick 2 and motivate. Also mention Bert. How? Also mention native baseline

## 3.4 Define evaluation criteria

### 3.4.1 CapGemini Interviews

### 3.4.2 Literature

### 3.4.3 Proof of Concept application

TODO Define evaluation criteria from (1) VisionMobile Report and (2) Interviews with CapGemini + PoC

## 3.5 Evaluate selected alternatives

TODO Using AHP, compare alternatives. Ask Jan to make 1 score table, derive 1 score table from VisionMobile report. Do the comparison twice or give both scoring tables adequate weights

## 3.6 Select the most suitable alternative

TODO Describe the advantages and disadvantages of the three candidates. Make a decision if only these three candidates are available

Chapter

# 4 Selected Tools

## 4.1 Apache Cordova

TODO Elaborate description of Cordova, including architecture, kind of app, supported platforms, used programming languages, costs, financial backing, community support, official support, ...

## 4.2 Motorola Rhodes

TODO Elaborate description of Rhodes, including architecture, kind of app, supported platforms, used programming languages, costs, financial backing, community support, official support, ...

## 4.3 Native SDKs?

Chapter 5

# Comparison Results

TODO Repeat the Methodology sections, this time doing the actual comparison.

Chapter

# 6 Conclusion

# Appendices

# Bibliography


[1] Android. Platform Versions, Screen Sizes and Densities and Open GL Version, 2013. URL http://developer.android.com/about/dashboards/index.html.

[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.

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

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

[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.

[6] Gartner. Gartner Says Worldwide Smartphone Sales Reached Its Lowest Growth Rate With 11.5 Per Cent Increase in Third Quarter of 2008, 2008. URL http://www.gartner.com/it/page.jsp?id=827912.

[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.

[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.

[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.

[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.

[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.

[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.

[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.

[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.

[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.

[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.

[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.

[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.

[19] Gartner. Gartner Says Worldwide Sales of Mobile Phones Declined 3 Percent in Third Quarter of 2012; Smartphone Sales Increased 47 Percent, 2012. URL http://www.gartner.com/newsroom/id/2237315.

[20] 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.

[21] Gartner. Gartner Says Worldwide Mobile Phone Sales Declined 1.7 Percent in 2012, 2013. URL http://www.gartner.com/newsroom/id/2335616.

[22] Anil S. Jadhav and Rajendra M. Sonar. Evaluating and selecting software packages: A review. *Information and Software Technology*, 51(3):555–563, March 2009. ISSN 09505849. doi: 10.1016/j.infsof.2008.09.003. URL http://linkinghub.elsevier.com/retrieve/pii/S0950584908001262.

[23] Anil S. Jadhav and Rajendra M. Sonar. Framework for evaluation and selection of the software packages: A hybrid knowledge based system approach. *Journal of Systems and Software*, 84(8):1394–1407, August 2011. ISSN 01641212. doi: 10.1016/j.jss.2011.03.034. URL http://linkinghub.elsevier.com/retrieve/pii/S016412121100077X.

[24] 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-

# 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