Chapter 1

Background

The first section of this chapter, terminology 1.1, explains terms that are used throughout this thesis. The next section, Android development methods 1.2, explains different methods of developing an Android mobile application. The following section, lines of code 1.3, describes lines of code as a software metric and what it measures. Lastly the section, related work 1.4, suggests related work to this thesis for further reading.

1.1 Terminology

Mobile application
: Refers to the application being developed, excluding code loaded from remote websites.

Web application
: If nothing else is specified it refers to the client-side of the web application.

Development methods
: Refers (in the context of this paper) to the methods of mobile application development methods being evaluated. I.e developing the mobile application with PhoneGap or natively in Android.

Native function
: A hardware function which a device has. Lets say a mobile has a camera and an accelerator. When writing software for that mobile there are functions to interact with the camera and accelerator. Those functions are called native functions.

Web/Mobile application layer
Refers to the application layer belonging specifically to the mobile application or the web application. A mobile application can encapsulate a web application which then is a a part of the mobile application. The mobile application then consists of a mobile and web application layer.

Native application
A native application is an application that is developed to be used on a particular device or plattform. For example, a native application for Android is an application that is written to run on the Android operating system.

Hybrid application
A hybrid application is a native application which is partially written with web technologies. I.e HTML5, CSS and JavaScript. The part of the application written with web technologies runs within a browsers engine in Android the browser engine is called WebView. A web application running in a browser does not have access to a device native functions such as the bluetooth. However a website running within a WebView can through interaction with the application's native code access a device native functions.

1.2 Android development methods

Android is an operating system used on a wide range of devices [7]. For example a mobile device or tv device. Developing Android applications are usually written in the programming language Java using the Android SDK.

A software library for a system is a set of functions which can be used when developing applications for that system. A framework is a layered structure to help or simplify for the developer to develop software for a system. A framework differs from a library in the following way [10]:

- The programs flow of control is dictated by the framework instead of the caller.
- A framework has a useful default behavior.
- The framework can be extended by the user.

1.2.1 Android application framework

The Android SDK includes a set of tools [8]. Tools such as virtual device tools, development tools, debugging tools and buid tools. An Android application that is developed natively is an application that is developed using the Android SDK. For developing Android applications Android provides an application framework [9].

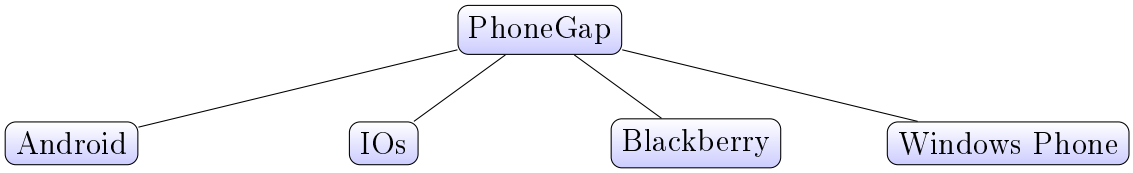

Figure 1.1: PhoneGap can be compiled into multiple plattoforms.

1.2.2 PhoneGap framework

PhoneGap is a framework for developing mobile applications. The code is written with web technologies. The code can be compiled to different platforms, such as Android or IOs specific code. The resulting application is a hybrid application. An example of a mobile application built in PhoneGap is Wikipedia's mobile application.

One of the huge advantages developing in PhoneGap compared with developing natively is that PhoneGap can be compiled to multiple platforms, see figure **??**. Making PhoneGap an attractive development framework for companies with less development resources. If a mobile application is developed natively for Android, IOs and Windows Phone, maintaining the mobile application must be done in three different native applications. If the mobile application is instead developed in PhoneGap only one application needs to be maintained.

A disadvantage of developing in PhoneGap is when their is a use of many native features. PhoneGap relies on a development framework and the provided features when building a mobile application. Hence, if the framework is not up to date with the latest new features, the developer will not be able to partake of the features until the framework is updated. If the application is dependent on many native features a hybrid application may have limitations[3].

1.2.3 Other frameworks

1.3 Lines of code

Source lines of code is a measurement tool for software development. Source lines of code, also abbreviated as SLOC, is very easy to obtain and is a fairly accurate predictor of development effort[1, p. 63]. Measuring SLOC simply means you count the number of lines of code. There are many different ways to measure SLOC, such as Halsted's approach, function points, physical SLOC and Logical SLOC.

Physical SLOC is the length of the code excluding comments and blanks. Function points measure functionality and can therefore be measured before the design and coding if the requirement specification is complete[1, p. 187]. Halstead's uses

measurable properties such as operands and operators and uses them to identify properties of software. Such as the length, difficulty and effort of the program. Fenton and Bieman describes Halstead's software science measures as a confused and inadequate measurement. Particularly for other attributes then size[2, p. 345].

Logical SLOC measures the number of statements that carry over one or more physical lines. For languages with terminators, this can be counted more easily and quickly. As an example, in Java you could count the logical SLOC by counting the number of line-terminating semicolons and closing curly brackets. Logical SLOC represents the programming instructions and data declarations which are converted into executable instructions, i.e. the implementation of the software design. Another positive aspect of of logical SLOC is that it better handles differences in formatting and style conventions than physical SLOC[1, p. 155].

To compare size between two different languages a size conversion table can be used. The table can be used to estimate how many SLOC a program coded in one programming language would have in another language. In the table constructed by Galorath and Evans you can compare a third generation language, a fourth generation language, Ada, Assembly or Pascal[1, p. 163]. A third generation language compared to another third generation language would have no conversion rate.

Source lines of code is a measurement tool for software development. Source lines of code, also abbreviated as SLOC, is very easy to obtain and is a fairly accurate predictor of development effort[1, p. 63]. Measuring SLOC simply means you count the number of lines of code. There are many different ways to measure SLOC, such as Halsted's approach, function points, physical SLOC and Logical SLOC.

Physical SLOC is the length of the code excluding comments and blanks. Function points measure functionality and can therefore be measured before the design and coding if the requirement specification is complete[1, p. 187]. Halstead's uses measurable properties such as operands and operators and uses them to identify properties of software. Such as the length, difficulty and effort of the program. Fenton and Bieman describes Halstead's software science measures as a confused and inadequate measurement. Particularly for other attributes then size[2, p. 345].

Logical SLOC measures the number of statements that carry over one or more physical lines. For languages with terminators, this can be counted more easily and quickly. As an example, in Java you could count the logical SLOC by counting the number of line-terminating semicolons and closing curly brackets. Logical SLOC represents the programming instructions and data declarations which are converted into executable instructions, i.e. the implementation of the software design. Another positive aspect of of logical SLOC is that it better handles differences in formatting and style conventions than physical SLOC[1, p. 155].

To compare size between two different languages a size conversion table can be used. The table can be used to estimate how many SLOC a program coded in one programming language would have in another language. In the table constructed by Galorath and Evans you can compare a third generation language, a fourth generation language, Ada, Assembly or Pascal[1, p. 163]. A third generation language

compared to another third generation language would have no conversion rate.

1.4 Related work

Bibliography

[1] Daniel D. Galorath and Michael W. Evans, *Software Sizing, Estimation, and Risk Management*, Auerbach Publications, Taylor and Francis Group, 2006.

[2] Norman Fenton, Queen Mary University of London, UK, James Bieman, Colorado State University, Fort Collins, USA, *Software Metrics: A Rigorous and Practical Approach*, CRC Press, Taylor and Francis Group, 3rd edition, 2015

[3] Bernard Kohan and Joseph Montanez, *Native vs Hybrid/PhoneGap App Development Comparison*, `http://www.comentum.com/phonegap-vs-native-app-development.html`, Comentum, San Diego, January 26, 201EffectiveUI5

[4] Online survey from Harris Interactive on behalf of EffectiveUI, United States, September 30 - October 4, 2010

[5] A white paper from Michaels, ross & cole, ltd, *Native mobile apps: The wrong choice for business?* `http://www.mrc-productivity.com/research/whitepapers/NativeAppsWrongChoice.pdf`, Lombard, IL, January 2013

[6] Gartner, Inc *Market Share: Devices, All Countries, 4Q14 Update*, `http://www.gartner.com/document/2985017`, Stamford, Connecticut, USA, March 2015

[7] Jolie O'Dell, *Androids Unite: How Ice Cream Sandwich Will End the OS Schism*, `http://mashable.com/2011/05/12/ice-cream-sandwich/#GIRObYC0nqkO`, May 2011

[8] Android Developers, `https://developer.android.com/tools/help/index.html`, September 2015

[9] Android Developers, `https://developer.android.com/guide/index.html`, September 2015

[10] Dirk Riehle, *Framework Design: A Role Modeling Approach*, Swiss Federal Institute of Technology, 2000