NTNU – Trondheim
Norwegian University of
Science and Technology

Mobile Apps Mobility

An evaluation of HTML5 for cross-platform development

Hanne Oustad

Master of Science in Informatics
Submission date: June 2013
Supervisor: Jon Atle Gulla, IDI

Norwegian University of Science and Technology
Department of Computer and Information Science

Problem description

Mobile apps are often restricted to one platform (Android, iOS, Windows Phone), and one market (i.e. localization). This task will analyze how to efficiently enable service development across platforms and markets for telco-enriched apps and services. In particular, we would like to evaluate the use of HTML5 as part of a cross-platform development strategy. A case study from mobile news recommendation is used to investigate mobile service development with HTML5.

Research questions:

- What are the advantages and limitations of using HTML5 as compared to native development on mobile platforms?
- What is the quality of current tools for HTML5 cross-platform service development?
- What methodological approaches are suitable for HTML5 app development?

The deliverable includes a prototype that demonstrates the use of HTML5 for a mobile news recommendation system.

Preface

This report is the documentation of the work done during the 5th year of my Master Degree in Informatics, at the Norwegian University of Science and Technology (NTNU).

I would like to thank my supervisor Professor Jon Atle Gulla at the Department of Computer and Information Science at NTNU and Erik Berg at Telenor for the interesting assignment, which have given me much new knowledge and experience in HTML5 development. I will also like to thank them for the valuable guidance and feedback during the project, both with the practical experiment and with writing this thesis.

Finally, I would like to thank Daniel Børseth for report proofreading and valuable input in the completion phase.

Abstract

In recent years, there has been a growing interest in cross-platform development and HTML5 as a development tool. The possibility for developers to code once and run on multiple platforms could decrease development costs and make it easier to target more platforms and reach more people. The popularity of HTML5 has attracted developers to create tools that make the development process easier and faster.

The goal of this thesis is to evaluate the use of HTML5 as development tool for cross-platform development, based on a case study where a prototype of a context-aware application for news recommendation is built. This goal is reached by studying current technologies, as well as the implemented solution of the mobile news application. After the theoretical study and implementation of the mobile news application is completed, an evaluation of the findings are carried out to check if HTML5 is suitable for cross-platform mobile application development, based on the given use case. The evaluation results are then used to give a conclusion about how well suited HTML5 is for this purpose.

The mobile news application has been developed both as a pure web application and a hybrid application, where Apache Cordova is used as framework for the hybrid approach. Evaluation of the two development approaches and the findings in this thesis shows that the hybrid approach was slightly better than the pure web approach for the given case, especially when looking at the quality of the application.

Abstract

I de senere år har det vært en voksende interesse for kryss-plattformutvikling, og bruken av HTML5 som et utviklingsverktøy. Mulighetene for utviklere til å kode en gang, og kjøre på flere forskjellige plattformer kan senke utviklingskostnader og gjøre det lettere å nå flere folk på flere plattformer. Populariteten til HTML5 har lokket til seg mange utviklere som lager verktøy for å gjøre utviklingsprosessen enklere, og raskere.

Målet ved denne masteroppgaven er å evaluere bruken av HTML5 som et utviklingsverktøy for kryss-plattformutvikling, basert på et case studie der en prototype av en kontekst-avhengig applikasjon for nyhets-anbefalinger blir laget. Målet nås ved å studere de nåværende teknologiene, i tillegg til å implementere en klientløsning til nyhets-anbefalings-systemet. Etter den teoretiske studien og implementasjonen av nyhets-applikasjonen er ferdig, skal resultatene evalueres for å avgjøre om HML5 egner seg som et utviklingsverktøy for kryss-plattform applikasjonsutvikling. Evalueringen av resultatene blir så brukt for å konkludere om til hvor stor grad HTML5 egner seg til denne oppgaven.

Nyhets-applikasjonen som har blitt laget har blitt utviklet som både en ren web-applikasjon, og som en hybrid applikasjon der Apache Cordova har blitt brukt som et rammeverk. Evalueringen av de to fremgangsmåtene, og resultatene i denne masteroppgaven viser at hybride applikasjoner er hakket bedre enn rene web-applikasjoner for det gitte caset, spesielt når en ser på kvaliteten av applikasjonen.

Contents

List of Figures

List of Tables

List of Abbreviations

API	Application Programming Interface
CML	Context Modeling Language
CORS	Cross-origin resource sharing
CPU	Central Processing Unit
CSS	Cascading Style Sheet
CSV	Comma-Separated Values
DOM	Document Object Model
ECMAScript	European Computer Manufacturers Association
GCD	Generalized Conjunction/Disjunction
GPS	Global Positioning System
HTML	Hyper-Text Markup Language
HTTP	Hypertext Transfer Protocol
IDE	Integrated Development Environment
ISO	International Organization for Standardization
JSON	JavaScript Object Notation
LSP	Logic Scoring of Preference
NTNU	Norwegian University of Science and Technology
OS	Operating System
REST	Representational State Transfer
RSS	Rich Site Summary

SDK	Software Development Kit
SGML	Generalized Markup Language
UI	User Interface
URL	Uniform Resource Locator
VM	Virtual Machine
W3C	World Wide Web Consortium
XML	Extensible Markup Language

1 Introduction

Development of mobile applications has in recent years become increasingly popular. The ever increasing popularity has led to studies of the most effective ways to develop applications, and the concept of cross-platform applications has become a major research area. In this thesis the field of interest is the HTML5 cross-platform development tool, methodological approaches and HTML5 development tools. Another interesting field is news recommendation systems. News recommendation systems can be distinguished from other news systems in that they give the user relevant news, and efficiently helps the user to find these news in the pile of news that is available and published at all times. This thesis looks at how these systems can be developed as cross-platform applications with the use of HTML5.

1.1 Problem

Today, more and more people are carrying around smart devices in the form of smartphones, tablets, and other handheld devices that give the users access to large amounts of information and functionality at request. The typical distribution of these functions are through mobile applications available through application stores such as the Apple App Store on the iOS platform, and Google Play on Android, and the distribution is often tied to the type of device you are using to access content. As a result, the development of an application from a developers stand point is a complex task that requires that the developer, or team of developers have a broad technical insight into the different options and constraints enforced upon them. This has lead to a need for simplifying the development process across development platforms and environments by creating cross platform applications.

The goal of a cross-platform application is to simplify the development process of an application from a development stand point, as well as reach a broader audience by targeting multiple platforms with out investing a larger amount of effort. The challenge lies in creating an application that behaves similarly on different platforms when considering quality, user experience and performance, as well as enabling support for the rapid speed of evolving technologies.

HTML5 is one of the most popular platforms that can be used to build cross platform-applications. These applications are run in web browsers, which means that they can be run on any device that has a web browser. There currently exist two kinds of cross platform applications that take advantage of HTML5: pure web applications and hybrid application. Web applications are basically web pages, that are designed to work on multiple devices. Because the application is on the web, it can be opened with any modern web browser, on any device. Hybrid applications takes advantage of a web view present in a native application, meaning they are built using web technologies, while native code is used to give access to wider functionality of the device. These applications can be stored locally on the device just like native applications. There exist many examples of successfully developed hybrid applications on PhoneGap's web page [1], such as the Logitech Squeezebox Controller.

For evaluating HTML5 as a development tool for mobile applications, a case study from mobile localized news are used. The case study is to develop a context-aware application in HTML5, that takes advantage of the users location as well as some user inputted preferences, to give a better experience for the user.

Together with this case study, there are some research questions this thesis will aim to answer. The research questions are:

- What are the advantages and limitations of using HTML5 as compared to native development on mobile platforms?
- What is the quality of current tools for HTML5 cross-platform service

development?
- What methodological approaches are suitable for HTML5 app development?

The first research question that this thesis aims to give an answer to is the advantages and limitations of using HTML5 as development tool, compared to native development. In which ways are mobile HTML5 development different from native development? What are the advantages and disadvantages with this approach? The biggest differences and the advantages and limitations for each approach will be discussed, and the findings related to HTML5 will be taken into account when developing the context-aware application.

The second research question the thesis aims to give an answer to is what the quality for current tools for HTML5 are. What are the capabilities and stability of these tools? What is the reputation of these tools? Different tools will be studied and discussed, and some of these tools will be used in the development process.

The third research question concern the methodological approach for HTML5 development. Which methodology will be used in developing the mobile news application? The methodology should be based on a real method, and some research will be done in how mobile applications usually are developed. The methodology used, and the experiences with this approach will be discussed.

Together with these research questions, the thesis will also include a prototype of a mobile news recommendation system developed in HTML5. This application is developed client side on top of an existing back-end system.

The overall goal is to see if HTML5 is a suitable development method for the given case study, together with characteristics and methods for this approach.

1.2 Approach

The research approach in this thesis is a theoretical study of HTML5, and a practical experiment of HTML5 in use.

First, the problem is defined and the research approach is determined.

Second, a theoretical study is performed by reading papers about cross-platform development and HTML5, and checking out current methodological approaches and existing tools that is used in HTML5 development. This information is gathered to get an overview of the state of the art, and other related work that is of interest.

Third, a practical experiment is executed by implementing a real HTML5 news application that retrieves data from a dedicated news feed, back-end system.

Last, an evaluation of the system and the findings is carried out, based on some characteristics.

1.3 Results

A goal of the research was to evaluate the use of HTML5 as development tool for news recommendation systems. Development of the system is based on research during the development phase, and evaluated based on some criteria. The results from this evaluation and the rest of the development process will be used to conclude the thesis. In addition to this system, the state of the art is discussed, and the theoretical background is introduced.

The results focuses on the evaluation of both the hybrid application and the pure web application, together with development process and characteristics of HTML5 development.

The results show that the gap between native applications and HTML5 is still present in some areas, while in others they compete equally. Some of

the problems with HTML5 applications have been solved by good resources such as libraries and frameworks that can helt in the development process.

The results related to the develoment methodology show that for small simple aplications, a traditional iteration based agile methodology should suffice, but for larger projects, other considerations have to be made.

1.4 Report Structure

The thesis is structured as follows:

Chapter 2: Theoretical Background introduces the theoretical background of the thesis. Cross-platform applications and native applications are introduced together with an introduction of HTML5, tools, and advantages and disadvantages with both web and native approaches.

Chapter 3: Related Work presents work from related projects and research done to identify the state of the art and current issues in the existing research.

Chapter 4: Case study introduces the case study and context-aware applications.

Chapter 5: Development Approach presents the development approach chosen in this thesis, and how the mobile news application should be built.

Chapter 6: Application design presents how the design of the application should be.

Chapter 7: Realization presents what was done and the application that was implemented.

Chapter 8: Evaluation presents and evaluates the results.

Chapter 9: Discussion discusses the results from the evaluation.

Chapter 10: Conclusion concludes the thesis, and tries to answer the question: Is HTML5 preferred over native development?

Chapter 11: Further Work describes possibilities for further work.

2 Theoretical Background

This chapter will provide a introduction to relevant technologies and concepts that are of interest in this project.

Section 2.1 will begin with an introduction of what an mobile application is, together with an introduction of the two different types of mobile applications: native and cross-platform applications, and their associated advantages and disadvantages. Section 2.2 compares native and web applications against each other, and section 2.3 introduces the web technologies HTML5, JavaScript, CSS and HTML.

2.1 Mobile Applications

A mobile application is a program developed to run on small mobile handheld devices, such as smartphones and tablets. These mobile applications are small software products that lets the device user perform specific task, e.g. playing games or communicate with others in social media applications. There are two ways to develop mobile application, i.e, the native development approach and the cross-platform development approach. Dependant on the chosen development approach, the mobile application can be preloaded into the device, downloaded or accessed through the app store or the Internet later on, or accessible through the mobile web browser.

Development of mobile applications began over 10 years ago, but the big boost in popularity first began when Apple introduced the iPhone AppStore in 2008. After that, both Android, Windows Phone and other platforms has created marketplaces, and the development of mobile applications have increased dramatically [2].

2.1.1 Native Applications

A native application is a software program that can perform a specific function on a particular platform, since they are built specifically for that platform [3]. There exists a number of different platforms, where some of the major platforms today are Android, Windows Phone and iOS [4]. Each of these platforms have their own programming languages, development environments and marketplaces that must be used when developing applications.

The programming language used when developing native applications for Apples iOS devices (iPhone, iPad and iPod) are primarily the Objective-C language. For the Android platform the programming language is Java, and for Windows Phone it is C# [5–7].

The development environment for each of the platforms is called the Integrated Development Environment (IDE). The IDE provides tools for writing, testing and deploying applications into the targeted platform environment. For Android, the officially supported IDE is Eclipse, for iOS it is XCode, and Windows phone uses Visual Studio. Both XCode and Visual Studio costs money, while Eclipse IDE is free to download and use [5–7].

Each of the different platforms have their own application stores, which are marketplaces where applications are published and made available for anyone who have a device with the same platform. The application stores makes it possible to browse and download applications that was developed for the same platform, and install them on the device [4]. The marketplace for Android is called Google Play, Windows phone has the Windows Phone Store, and the iOS marketplace is called App Store [5–7].

Each of these platform specific properties are summarized in table 2.1.

	iOS	**Android**	**Windows Phone**
Programming language	Objective C	Java	C#
IDE	XCode	Eclipse	Visual Studio
Marketplace	App Store	Google Play	Windows Phone Store

Table 2.1: Platform properties

ADVANTAGES AND DISADVANTAGES

There exist both advantages and disadvantages with the native development approach. The biggest advantage with native applications is that they can take advantage of native features like camera and GPS, because of the deep integration with the Operating System (OS) and hardware [4].

In native development ecosystems, there also exist well established guidelines for usability and user experience. These can make the development process easier, especially for inexperienced developers, and the design will be more suitable for the target platform. Native applications are usually responsive, which gives less lag and latency. This is because most of the data in the application is part of the application itself [4].

Native applications also provide one-click access to applications, and they are by default bookmarked and persistent. When an application is installed on the device, the application is always available with its own small icon in the application menu. Because they are installed on the device, there is no need for a persistent network connection to access them. The application is downloaded and installed from the application stores, and these application stores makes it easier to discover and find applications. Because of this, it is easier to make money as a developer on a native application because of the application store, since you set a price on the application, and then people have to pay for it when they download it [3].

The biggest disadvantage with native applications is that they are platform

dependent, meaning that the developers need to develop own versions of the application for every platform that should be reached. This is a big disadvantage because it requires a need for broad programming skills, because of all the programming languages and environments used in the different platforms.

Another disadvantage is that the application can not update itself, but instead, the user has to choose to update to the newest version when it is available [4].

At last, since each platform requires their own version of the application, the development, testing and adaption to each of the different environments, including maintenance and marketing, is costly [3].

2.1.2 Cross-Platform Applications

Cross-platform applications are mobile applications that can run on different platforms without large modifications [8].

There are several ways to create cross-platform applications [8]:

- **Cross-compilation** - Compiles source code into different target environments (platforms).
- **Virtual Machine** - Executes applications as virtual machines.
- **Mobile web application** - Utilize HTML5 to create web applications that can be run on a device.

Cross-compilation is a technique where a cross-compiler is used to separate the build and target environment and the application from each other. The cross-compiler works as a transformer that transforms the source code into a platform specific native application. The resulting application can be executed and used like a native application on the device. The big advantage with cross-compilation is performance since the application behaves as a native application on the device and provides a better user experience. The application will also have access to device features such as the camera and sensors of the device. The problem with cross-compilation lies in

the complexity of the source code that is produced. To be able to support all targeted platforms, the source code has to reflect the differences of these platforms in the source code, which can be difficult to write and maintain [8].

Another way to make cross-platform applications are with a *Virtual Machine* (VM). A VM is used to execute programs like a real physical machine. The advantage with this is that the application becomes very portable, because it is easy to maintain and extend. The disadvantage with this technique is that applications tend to run more slowly in virtual machine environments, because of runtime interpretation latency [8].

The last way of creating cross-platform applications is as a *Mobile web application*, which is the focus area of this thesis. By using HTML5 to create a cross-platform application, the application can run in a mobile web browser, or in a browser view embedded into a native application (hybrid application) on the device [8].

2.1.3 Mobile Web Applications

Web applications are applications that live on a web server, and not on a mobile device itself. These applications can be accessed through the web browser on the mobile device. A mobile web application is built with HTML, CSS, and JavaScript [4], and is usually a mobile application, or a mobile adapted static web site.

When the application is run like a pure web application it is immediately available on the mobile web browser, just by typing in the Uniform Resource Locator (URL) of the application in the browser. This assumes that there exists an active data connection, and that the application server is up and running. Because of this, the application is accessible on any device that has a web browser, which makes the web application device and platform independent [3].

ADVANTAGES AND DISADVANTAGES

The biggest advantage with web applications is the cross-platform compatibility. This means that the web application can be accessed on multiple platforms just by typing in an URL, which results in the advantage that there is no need to download anything. Web applications can be updated continuously, and the most recent version of the application is always accessible in the mobile browser. This means that the user does not need to bother with version tracking and updating applications on the device [3].

Because the application is developed once and with only one code base, the development process is cheap, and the application is easy to build and maintain. The developer can with just small changes, cover several platforms without any big costs.

The disadvantages with pure web applications are often related to user experience. The user experience of a web application can be poor because of restricted access to device capabilities, and it is impossible to emulate a native UI inside a web browser [8].

Another disadvantage is that it is not possible to upload the application to marketplaces like the Apple App Store or Google Play Store, which makes the applications less visible than applications that can be published in application marketplaces. To access a web application, it takes several clicks and text entry input to access the application in the web browser, and because of the limited input methods, this can be a challenge.

Another disadvantage with web applications is that they are not that good with interactive, CPU-intensive and visually rich applications, like games. Also, they generally require a persistent network connection [3].

2.1.4 Hybrid Applications

Hybrid applications are a combination of native applications and web applications. A hybrid application is developed using web technologies such as HTML, CSS and JavaScript, and then they are wrapped inside a native

container so that they can run as a native application [4]. The framework Apache Cordova (previously named PhoneGap) [9] is an example of such container, that makes the application works like a native application on the device.

ADVANTAGES AND DISADVANTAGES

With hybrid applications it is easy to develop applications with user-friendly tools and familiar programming languages instead of having to learn several programming languages to be able to create applications for several platforms [4].

An advantage is that some hybrid applications can connect to the mobile web, and utilize a native container to access a mobile web application through a native browser view, this means that the applications receives some of the same advantages of mobile web applications, as well as solve some of the problems such as availability. Hybrid applications, like mobile web applications, can update fast and continuously [4].

A hybrid application is stored locally on the device like native applications, and is accessible through a small icon in the application menu, and the need of data connection is removed if the application is run locally from the device. Just like native applications, hybrid applications also have access to native features because of the native wrapper they are contained in.

Hybrid applications are *wrapped* in a native app wrapper before they can be installed on a mobile device. This means that the application is deployed to a specific platform. It is not all hybrid application generators that support all platforms, so you need to pick a generator that wraps in the desired platforms [4]. When a hybrid application is deployed on a platform in this way, and it only run locally, it is important to notice that the continuous updating of the application is removed, since the application has to be manually updated.

2.2 Web versus Native Applications

This section compares the native and web approaches against each other, and looks at advantage and disadvantages with both approaches.

As described in the previous sections, a web application approach is cross-platform, while the native application approach is platform dependent. This means that it is theoretically easier and cheaper to build web applications since they require less knowledge about specific technologies.

When considering the development environments for both approaches, there is a bigger requirement for development knowledge in native application development than in web development. In native development, the developer has to know many programming languages if the application should be accessible on multiple platforms, and this requires expertise in all of the programming languages. This also means that a developer has to actually create code in all of these languages as well, which is time consuming. When developing web applications, the developer only needs to know the HTML5 technologies as development tool. Because of this, there is a bigger development cost in native development that web development. This can also be said for each of the platform Application Programming Interfaces (API). Every framework for creating native applications has different APIs, so in addition to the various programming languages, one must also know the APIs for the various operating systems. And these are usually large.

Another important development environment concern is the IDE that is used when developing the applications. For example, Google has facilitated the use of Eclipse, but this is not required, while Apple require the use of XCode. This can create difficulties if one does not have access to a mac. For web applications, you can develop using whatever development environment you want.

A constraint that enforced on a developer that develops using a web technology approach is that access to the native features of the mobile device is

limited. In a native development environment, these features are accessible and can be used to enrich the application. While some of these features are unavailable, e.g., sensory information, HTML5 has support for others, such as geolocation [10].

Native application development defines guidelines for how the application should look and behave to give the application a good user experience. This is an important usability aspect since most applications will follow the guidelines, and resemble each other. This can also be seen in the user interface libraries and technologies that are used in native development. As an example, Android uses the Extensible Markup Language (XML) to create the user interface. These user interface libraries define pre-defined components that adhere to the guidelines of the platform. Each platform uses different technologies for creating the views in the application. This is also the case in web application development, but one can use one single technology to define the views. By default, the available web technologies do not define component that follow any guidelines for web applications, but it is possible to include third-party libraries such as jQuery mobile and Sencha Touch.

When native applications are published on application stores, they are accessible for everyone that wants to download them and install them on their devices. This is not possible for web applications, since there are not any application stores for HTML5 applications for some of the biggest native platforms. The only application store that consists of HTML5 applications is Firefox Marketplace [11].

By publishing applications to application stores, the applications are always accessible by one-click access in native development. For web applications, the applications is accessible in any web browser, and there is no need for the user to download the application. Because the application is accessible in the web browsers in web applications, the newest version of the application is always accessible. In native applications, the user has to update the application manually.

Web applications usually requires network connection, while native applications are not required to have a persistent network connections since the application code resides on the device, and not on a remote server. In some cases it is also possible to run HTML5 applications from the device.

2.3 Web Technologies

There exist a number of web technologies which helps you develop web applications, like Cascading Style Sheets (CSS), Hyper Text Markup Language (HTML) and JavaScript. HTML5 is a collection of these web technologies. This section introduces HTML5, JavaScript, CSS and the HTML standard.

2.3.1 HTML5

HTML5 refers to the web technologies HTML, JavaScript and CSS [12].

The main idea behind HTML5 was to reduce the need for external plug-ins, improve error handling, get web applications to behave more like native applications by adding support for location-based services, and separating content from presentation in the syntax [12].

The biggest improvement in HTML5 is that the use for external plugins is reduced since native support for playing audio and video has been implemented into the browser. In a desktop environment this has usually not been a problem since several plug-ins such as Adobe Flash, Apple Quicktime and Microsoft Silverlight have provided the necessary functionality. For iPhone, iPad and Blackberry users, this is a problem, since there is no plug-in support. HTML5 solves this problem by allowing built-in support for video in browsers [12].

2.3.2 JavaScript

JavaScript (originally called LiveScript) is a scripting language developed in the early 1990s by Brendan Eich that worked at Netscape. JavaScript

is a client side scripting language, which runs in the browser and defines interactions and animations. JavaScript is usually embedded in the web page or included from a JavaScript file, and it is used to manipulate the Document Object Model (DOM) to the web page [13].

JavaScript also goes by the name ECMAScript (European Computer Manufacturers Association) , which is the standardized version of JavaScript [13].

2.3.3 CSS

CSS is a style-sheet language for stylistic control beyond HTML, it defines style and presentation. The work with CSS started in 1994 at CERN, and the first edition (CSS1) became a W3C Recommendation in 1997 [14]. CSS1 is a style sheet which attaches style to HTML documents. The second edition (CSS2) became a W3C Recommendation in 1998, with the new feature media-specific style sheets [15]. Today, CSS3 is the current CSS module [16].

2.3.4 HTML

HTML was initially created by Tim Berners-Lee in 1989 [12], as a way for researchers to collaborate and share data electronically. HTML was based on Generalized Mark-up Language (SGML), an existing language. The previous revision of HTML came in 1999, and a lot have changed on the Internet since then. HTML is used to define static text and images.

Today, it is heading towards a new revision of HTML, the HTML5 standard. HTML5 (Hyper Text Markup Language revision 5) is the latest HTML standard, and is still in progress. It is developed by the World Wide Web Consortium (W3C) [12].

3 Related Work

Cross-platform mobile application development has been a popular research topic the last years, which has resulted in several studies in this area.

This chapter will give an overview of the state of the art for mobile cross-platform application development tools, including advantages and disadvantages with HTML5, development methodologies and where HTML5 are headed.

Section 3.1 looks into previous research in advantages and disadvantages in web and native development. Section 3.2 looks into reserch on cross-platform development tools. Section 3.3 describes current research in development methodologies for mobile applications. Section 3.4 introduces the future of HTML5, and section 3.5 gives a summary of the related work found.

3.1 Web versus Native

Mobile web application development compared to native application development have both advantages and disadvantages. Andre Charland and Brian Leroux (2011) studied and discussed strength and weaknesses in web and native development, with focus on areas where the gap is closing between the two approaches [10]. Some of the main research areas was Native and Web Code, User Interface Code, User Experience, Performance and Design.

- **Native and Web Code** - When developing native applications for every platform, the programming languages, SDKs, tools, build systems and APIs is different for every platform, which makes the skills

required high. In the case of web development, the only thing the developer needs to know is JavaScript, HTML and CSS to make an application, and the PhoneGap framework can be used to call native device features via a common JS API.

- **User Interface Code** - Native platforms has user-interface controls and experiences to make consistent user interfaces, while in web development this is a bigger job.
- **User Experience** - The context, Hardware, Platform conventions, environment and implementation are all aspects that affect the user experience.
- **Performance** - Both latency and execution time are important performance points.
- **Design** - The web approach has some limitations when it comes to beautiful design.

To summarize, the web approach has not yet achieved the same level of performance as native code, but the gap is getting smaller.

Lionbridge (2012) studied key characteristics of native and web applications, and their advantages and disadvantages [3]. A summary of the advantages and disadvantages with the different approaches are given below.

- **Web application advantages** - They are cross-platform compatible, which mean they reach multiple platforms. They are cheap and easy to build and maintain, and there is no need for any downloads. Because the application is accessible through a URL, the newest version is always available.
- **Web application disadvantages** - There is a limited mobile device functionality support, and hardware and software on mobile devices are often not accessible. They also generally require a network connection.
- **Native application advantages** - They can access hardware and software on mobile devices. They have the ability that they can run offline, because they are installed on the device. They are also

always visible in the application list screen, and the application stores reminds the users to update the applications. It is also easy to make money on native applications, since you set a price for download in the application stores.

- **Native application disadvantages** - Separate versions of the application for every platform that should be reached need to be developed, and the maintainance requires more work. Content publishers also have to share information about their subscribers with the application stores.
- **Hybrid application advantages** - Blends both the web and the native approaches together, and reduces the development time and cost. The application looks and behaves like an native application, but it is developed with web technologies.
- **Hybrid application disadvantages**: The web-based and native functionality should blend together seamlessly, which is a design issue.

To summarize, mobile applications reach a wider audience with less costs, and native applications costs more, but provide a richer user experience. The native applications and the web applications complement each other, and both approaches should be used, but there could be a solution to try the web approach first if companies are new in mobile development.

3.2 HTML5 Cross-Platform Service Development

HTML5 development tools and the quality of those tools are an interesting topic that will be looked into.

Gustavo Hartmann, Geoff Stead and Asi DeGain (2011) looked at cross-platform mobile development [8], with focus on cross-platform approaches, development tools and m-learning development.

Different approaches to solve the cross-platform problem are first discussed

in the paper, which is cross-compilation, a Virtual Machine (VM) or a mobile web application. Then, development frameworks and tools are studied and compared.

The types of frameworks are divided into the following groups: *Libraries*, *Frameworks*, *Platforms and Products/Services*. The libraries are small toolkits that offer a specific functionality. Frameworks are collections of libraries, software components and architecture guidelines. Platforms are sets of frameworks, tools and services, and products/services offers a specific functionality. The most significant players when the research was done is summarized in a table, as shown in figure 3.1.

Framework	URL	License	Type
Rhodes	http://rhomobile.com/products/rhodes/	Open Source	Platform
Phonegap	http://www.phonegap.com	Open Source	Framework
FeedHenry	http://developer.feedhenry.com/	Commercial	Platform
Appcelerator	http://www.appcelerator.com/	Open Source	Platform
Grapple	http://www.grapplemobile.com/	Commercial	Framework
MotherApp	http://www.motherapp.com/	Commercial	Framework
Corona	http://www.anscamobile.com/corona/	Commercial	Product
Sencha Touch	http://www.sencha.com/products/touch/	OS/Commercial	Library
MoSync	http://www.mosync.com/	Open Source	Platform
Resco	http://www.resco.net/	Commercial	Platform
CouchOne	http://www.couchone.com/products	Commercial	Platform
MobileIron	http://mobileiron.com/	Commercial	Platform
WidgetPad	http://widgetpad.com	Open Source	Platform
AML	http://www.amlcode.com	Open Source	Framework
Jo	http://joapp.com	Open Source	Library
xui	http://xuijs.com	Open Source	Library
JQuery Mobile	http://jquerymobile.com	Open Source	LIbrary
JQTouch	http://jqtouch.com	Open Source	Library
QT	http://qt.nokia.com/products/qt-for-mobile-platforms/	Open Source	Framework
QuickConnectFamily	http://www.quickconnectfamily.org/	Open Source	Framework
Bedrock	http://www.metismo.com	Commercial	Platform
WebApp.net	http://webapp-net.com/	Open Source	Framework

Figure 3.1: Framework summary [8]

Some of these frameworks are then reviewed and described in more detail,

these were Rhodes, Phonegap and Appcelerator Titanium. For each of these frameworks the device capabilities are listed.

Finally, different types of m-learning is described with associated recommended technical building approach.

To conclude, there exist a number of tools, and software engineers need to make use of the right set of development tools to create mobile learning solutions quickly and efficiently. The cross-compilation and mobile web applications were the leading cross-platform development techniques. Where cross-compilation is when common source languages are used to develop the application before they are compiled to the different platforms. The mobile web application is built with common web technologies including HTML, CSS and JavaScript, while the application is accessible through the mobile web browser.

3.3 Methodological Approach

When developing mobile applications it is natural to look at the development process and the methodology approach used.

Tony Wasserman (2010) looked at engineering issues related to mobile application development, including the development process [17]. His findings were that agile techniques were the most common in all but the largest and most complex development projects, and that agile methodologies such as Scrum and Test Driven Development were used in mobile development projects, even in single person teams.

The article considers the need for *scaling up* when the complexity of the development process increases. Approaches that work well for single-person teams might no work well in more complex development projects, because requirements can change, which greatly impacts the need for testing and architectural decisions. The focus of the article is on the increasing amount of unique aspects of mobile application development.

Developing mobile application is similar to software engineering for other

embedded applications, but adds some extra requirements that needs to be taken into account when developing for mobile devices. Interaction with other devices, sensor handling, native and mobile web applications, differences in hardware and software, security, user interface guidelines, test challenges, and power consumption are requirements mentioned in the article. The following list summarizes the most important of these requirements:

- **User interface** - The user interface is critical in mobile application development because of smaller screens and different styles of user interaction. This means that the interaction design of the application plays a more important role in the development process.
- **Non-functional requirements** - The success of a mobile application depends on a list of non-functional requirements, especially related to performance, e.g. efficient use of device resource, scalability and responsiveness.
- **Testing** - Because of the fragmentation of devices, it is important to not just test an application in a emulator, but also extensively test the application on different devices with different configurations.
- **Maintenance** - Because the world of mobile platforms is rapidly changing, continuous updates of applications are required, but there is no guarantee that a user or group of users will update their devices and applications.

These requirements introduce more complexity into the process of developing mobile applications and may need extra attention by introducing processes into the agile development approach to target these requirements.

3.4 The Future of HTML5

HTML5 is still under development, making the future of HTML5 interesting.

Peter Lubbers, Brian Albers and Frank Salim (2011) looked at the future of HTML5 in their book, Pro HTML5 Programming [18]. They looked at

where thing are going, and discussed some of the promising features that will come. Some of these new features are SD graphics, the new device element, gestures, touch events, and Peer-to-Peer networking. These new features will when they are implemented in browsers enable developers to create more rich device supported web applications without the need for native containers.

3.5 Summary

In [10] and [3], web versus native applications are discussed. The articles both conclude that web applications at present can not perform and act on the same levels as native applications, but that they are getting better and better.

In [8], various tools for cross-platform development are discussed. The article concludes that the cross-compilation technique, and the web application technique is the preferred method of creating cross-platform applications.

In [17], the impact of mobile application development on existing development methodologies were discussed, and the need for the introduction of new processes into existing methodologies were expressed.

In [18], the feature of HTML5 was discussed in relation to many new features that are yet to be implemented in common browsers.

4 Case Study

To evaluate the use of HTML5, a case study where a context-aware application is built, is used. The applications should be build both as a pure web application, and as a hybrid application.

Section 4.1 will begin by explaining the theory behind context-aware applications, and the Context Modeling Language tool which will be used to show how the mobile news application should work by specifying the context of the application. Section 4.2 describes the context-aware news services and their characteristics. Finally, section 4.3 introduces the HTML5 mobile news application.

4.1 Context-Aware Applications

The concept of a context-aware application is an application that adapts to the context of use. As an example, consider an application that based on the users location, shows a direct route to the nearest restaurant that fits the users preferences.

This kind of application operates in a dynamic environment where the decisions of the application are computed continuously and with minimal interaction from the user, which means that in theory, the content shown on the application is always what the user expects to see.

A context-aware application makes decisions based on four types of information [19, 20]:

- **Sensed** - Highly dynamic information that is prone to noise and errors.
- **Static** - Fixed information that remains the same over the lifetime of the entity that owns it.

- **User-supplied** - Initially reliable information, that degrades over time.
- **Derived** - Information that is retrieved by deriving information from other information sources.

These information sources contribute to the development of context-aware applications, but there exists some problems that arise when the information is wrong or stale. Some of these problems are possible to solve by defining meta-data for information that define values for e.g. freshness or certainty.

4.1.1 Context Modeling Language

The Context Modeling Language (CML) [20] is a tool created to assist in the specification phase of creating a context-aware application. CML provide constructs that can describe types of information, their classifications, meta-data, and dependencies.

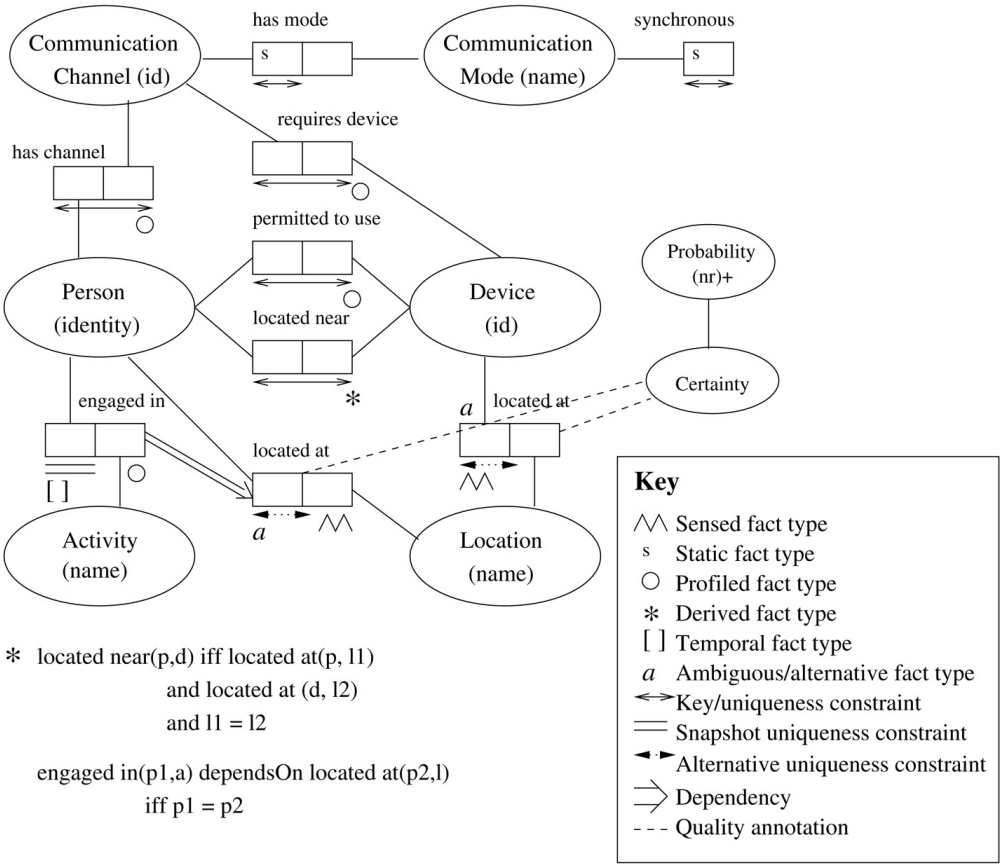


Figure 4.1: An example CML model [20]

Figure 4.1 shows a model example created in CML. The figure shows how objects (ellipsis) are connected together by a role played by an object type inside a fact type. As an example, the *located near* fact type has two roles, one played by the device, and the other played by a person. The fact types are annotated and shows the fact type, e.g. the *located near* fact type is a derived fact, and the role played by the device inside the *located at* fact type is a sensed fact type, and an alternative fact type (since it can have multiple locations) [20].

4.2 Context-Aware News Services

A context-aware news service provides news that can be filtered on context information. These services collect news articles from various news content providers, and deliver them based on the inputted context information. The context could be the news articles category or any other inputted attributes, e.g., the time the article were posted, some location data from where the article is from, or other attributes stored with the article [21].

As an example, smartphones have a limited screen size, so it is more cumbersome to navigate sites and filter out news by eye. In these cases, a context-aware news service can provide a user with news that are tailored to any filters that the user provides, which should give the user easy access to categories or other articles the user wants to access [21].

4.2.1 Characteristics of a News Service

When developing a news service, some considerations have to be made about how the service should be constructed. Below, some characteristics that are important to news services are defined:

- **Content driven** - The focus of a news service is on the content delivered to the user.
- **Continuously updating** - The news service should deliver recent or current news articles, rather than related or past articles.
- **Usability** - A news service should be easy to navigate and use.

These characteristics can be interpreted in various ways. The usability of a news service is dependant on what kind of medium the news are delivered on. As mentioned previously, devices with smaller screens often require more creative solutions for presenting articles than desktop clients. These same constraints also affect the content delivered by the service, i.e. a desktop client is able to show more news than a smartphone.

By adding context-awareness to a news service, the characteristics of the service can change based on the context of use. To continue the previous example of desktop versus smartphone devices, a user using a smartphone can get different or a sub-set of articles, while on a desktop client, the user might receive all relevant news.

4.3 A Context-Aware HTML5 News Application

This case study will design and implement an HTML5 application for delivering news based on contextual information such as location and user preferences.

The application will work as a front-end for a news recommendation system created by NTNU SmartMedia, a program at the Department of Computer and Information Science at Norwegian University of Science and Technology, with close collaboration with the Scandinavian media industry [22].

This section presents objectives and method of the study, as well as the pre-existing system the application will be based on.

4.3.1 Objectives and Method

The goal with this case is to carry out a development process by setting requirements. The requirements affect how HTML5 are evaluated as a method for developing applications. By conducting a case study, subjective impressions of what is important in the application is achieved. This will affect the evaluation later on. What is important is that it will be evaluated.

The process of conducting the case study is accomplished by first create specifications, then design, define, implement, evaluate, discuss, and conclude.

4.3.2 Existing News Recommendation System

The existing system for news recommendation has been created to overcome a set of challenges that limit the effectiveness of recommendation systems in general. Specifically, the identified challenges listed below are related to the use of news recommendation in a mobile context [23].

- **Limited user interfaces** - Mobile devices in general have small screens which make it difficult to show all the information one wants.
- **Short life cycles** - News articles have short life cycles.
- **Cold start users** - Users that request a recommendation without having any preferences.
- **Cold-start news articles** - News articles that are not tied to any users preferences are difficult to recommend.
- **User's desirability** - What kind of news articles a user wants to see are hard to detect.
- **Context** - The context of the user decides the users preference for particular articles.

The recommendation system retrieves information from two sources: *real-time news streams* and *real-time web services*. These two sources are the basis for two separate server side components, the *News data preparation and aggregation* and the *Recommendation engine*.

Figure 4.2 shows the system architecture of the news recommendation system. The details of the system are described in the sections below.

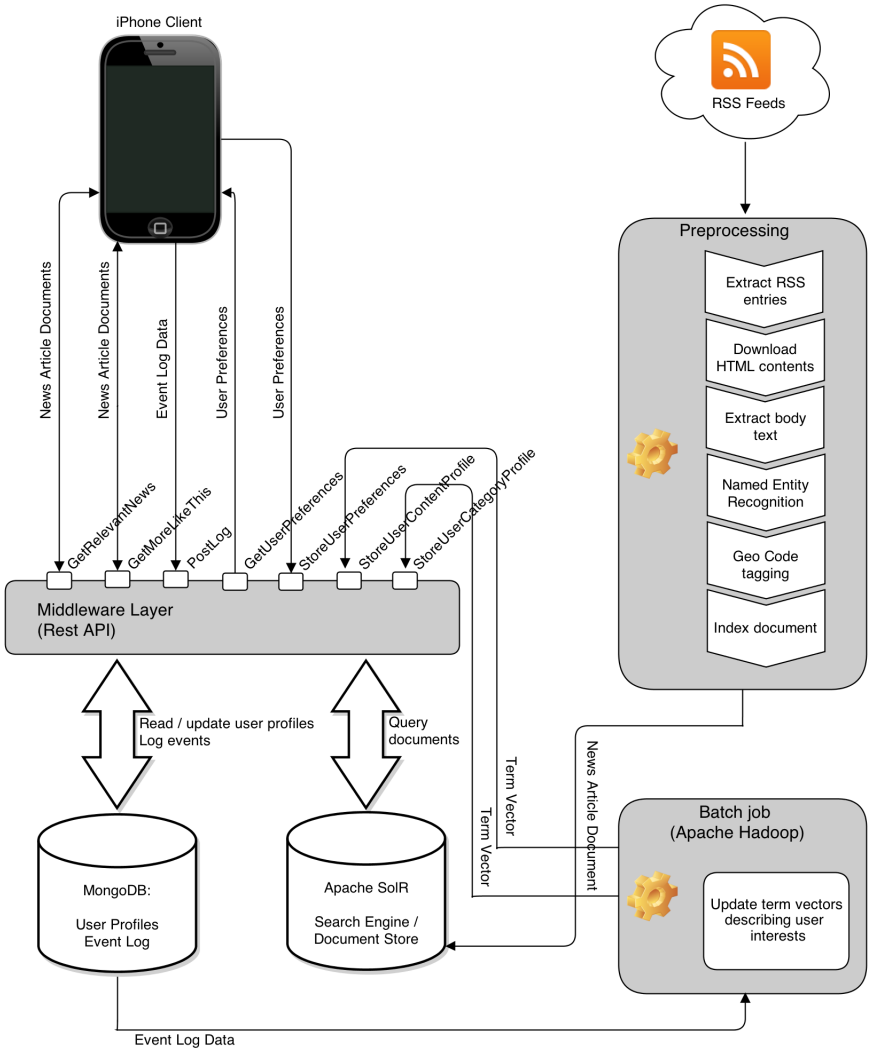


Figure 4.2: News recommendation system architecture

NEWS DATA AGGREGATION AND PREPARATION

When the news recommendation system retrieves data from the real-time news stream, it needs to be processed into indexed documents. This process consists of four steps:

- **Fetching** - News articles are fetched from the RSS feeds. At this point the articles contain entries such as title, lead text, images, URLs, etc. The URLs are used to fetch the entire HTML document

of the article, which is then parsed to get the desired content.

- **Syntactic processing** - The Apache OpenNLP library is used to identify location, names of persons and organizations.
- **Semantic processing** - The Google Maps API is used to find geocodes of the location names found in the previous step.
- **News indexing** - The articles are indexed by using SolR search indexing. This allows client applications to search for content and build queries for retrieving content.

After these steps, some of the information is further analyzed to provide better recommendations. The locations that were identified in the news articles are compared and filtered to only show relevant locations, and the similarity of news is determined based on the context of the user.

RECOMMENDATION ENGINE

The two most important factors of the recommendation engine are *freshness/recency* and *user preferences*.

The preferences of the user are split into short-term and long-term, where the long-term preferences are defined by the users profile, while the short-term preferences of the user is based on the users context and the popularity of the news articles. The users context is determined by the users location, and the users recent preferred news articles. What articles are popular is determined by measuring their popularity by listening to the real-time web services, i.e. Twitter.

4.3.3 Interface

The interface the news recommendation system is available as a REST API. To access the data from the system, one has to send HTTP GET requests with different kinds of filters for querying the news recommendation system for specific news.

This REST interface can deliver content on a number of different formats, such as CSV, XML, JSON, and other formats, but XML is the default

standard. An example of an URL is shown below:

{end-point}?rows={rows}&wt={format}&sort={order}&q={query}

Here the structure of the API URL is shown, where the *end-point* in the above URL is the end point for the request. The *rows* query parameter controls the amount of articles that are returned from the system. If this parameter is not specified, the system returns 10 articles. The *format* query parameter controls the format of the data response, and the *order* query parameter controls the order of the received articles. The last query parameter *query* is a query that determines what fields that must be present. Below, an example query is shown:

lead-text:*%20AND%20text:*%20AND%20title:*%20AND%20cat:*

The above query will only return results that have at least a lead-text, text, title, and category. Other options are also available as defined in [24].

5 Development Approach

This chapter introduces the development approach of how the application should be built. The development approach covers the full scope of the development process, from design to chosen methodology, and makes it easier and more efficient to develop the application.

Section 5.1 starts by describing the chosen development methodology. Section 5.2 introduces the development environment for the planned mobile news application, including targeted platforms and development constraints. Section 5.3 describes the current practices and products available for developing mobile applications, and section 5.4 introduces the use of requirements in the development process. Section 5.5 describes how the specifications and design will be performed.

5.1 Methodology

The development process will use a agile methodology where small pars of the mobile application will be designed and created in separate phases. After each phase (iteration), the functionality created is evaluated and used as a base for the next iteration.

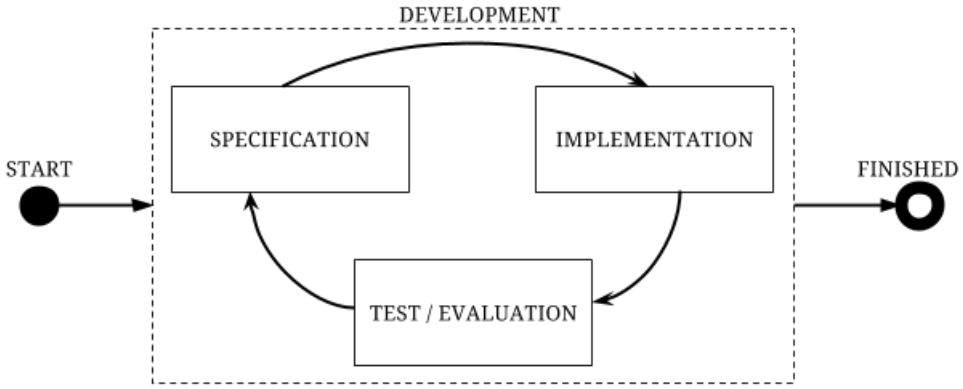


Figure 5.1: Method diagram

Figure 5.1 show a flow chart of how the work will be done methodically. Each iteration will be in two weeks, where a functional requirement that will be developed is specified, implemented and tested/evaluated. The evaluation is based on the specified function, and how it is decided that it should work.

The selection of requirements is based on dependencies, and the requirements that must be completed before others to work is first developed. For example, If a user wants to open a single news article, the list of news articles has to be implemented for this choice to be available.

The testing and evaluation will be performed on the different targeted platforms, and they will be checked against the requirement too see if they are met.

This methodology will allow the implementation of the application to be a working application after each iteration. In this way, risks can be reduced by not starting on implementation specifics that will take too much time.

5.2 Environment

The purpose of the case study is to develop a mobile application by using HTML5. In essence, a mobile HTML5 application is a standard web application that is adapted for mobile platforms and devices. This means that

the application environment is shaped by the strengths and limitations of both a standard web development environment, and a mobile development environment.

As an example, although most web applications are accessible through a mobile device, a lot of quality aspects are lost as a result of various constraints, e.g. smaller screens can only show limited amounts of information without compromising on readability.

When considering the development environment, some considerations have to be made. Since one of the problems with native development is device and platform fragmentation, the application should solve this to a certain degree. This means that the platforms the application should be developed for have to be identified, and any unique features and limitations of the platform must be considered.

5.2.1 Platforms

One of the important aspects of HTML5 applications is that the applications should be able to run on all platforms able to render HTML and evaluate JavaScript and CSS. This case study will only consider some of the most popular platforms, i.e. Android, iOS, and Windows Phone.

When developing native applications, each of these platforms require a specialized development environment, but when developing a HTML5 application one can use a simple text editor to create an application.

The development of the application will primarily be performed in the Eclipse IDE [25]. The application code will consist of the HTML, JavaScript and CSS required to run the application.

One important aspect of mobile web applications is that they can be run and tested in any web browser. This enables a continuous feedback loop when developing the application.

5.2.2 Constraints

Since all the targeted platforms have different development environments, it is problematic to develop an application on a single development machine. In the case of android, the development is made easy by using free software such as Eclipse for development, but iOS requires the XCode IDE and an Intel based computer with Mac OS X Lion as a development environment. The same constraints apply to Windows Phone which requires Visual Studio for development.

Other difficulties relate to testing the application during development on different devices and configurations. The device fragmentation related to smartphones is huge, and devices come in lots of various configurations. To be able to efficiently test the application on these configurations an emulator has to be used.

When developing web applications, the browser enforces constraints on the web applications by limiting the execution of scripts, and sending and retrieving of information by origin. This constraint is called the *Same-origin Policy* [26]. This means that all XMLHttpRequests [27] from the application to the news recommendation system will be blocked, since they have a different host name and port (originator). To work around this, a Node.js [28] proxy server will be made to act as an intermediary between the web application and the news recommendation system.

5.3 Current Practices and Products

The web development and mobile development community is currently experiencing a rapid development of new technologies that can be used to create applications more efficiently. This means that as a developer, the amount of frameworks and libraries that are available for use is large, and new and better solutions continuous to emerge. This also means that a large number of technologies are orphaned when they become obsolete or another frameworks solves the problem better. When considering frame-

works and libraries, it is important to consider these factors.

Apache Cordova [9] is a platform for creating hybrid applications by using HTML, CSS, and JavaScript. It is currently the only platform for creating hybrid applications that support Android, iOS and Windows Phone, that is also free. Other platforms such as Appspresso [29] and Application Craft [30] have limitations or cost money.

When using a platform for creating hybrid applications, as well as when creating mobile web applications, mobile UI frameworks such as jQuery Mobile [31] and Sencha Touch [32] are used to create UI components that try to stick to the design guidelines defined by the various device platforms.

5.4 Requirements

To be able to create an applications that can act as a client for the news recommendation system described in section 4.3.2 some requirements have to be defined. These requirements should reflect the needs of the system, and will also be used in decisions in the development effort related to functionality.

Both functional and non-functional requirements will be defined, the functional requirements will describe how the news recommendation application should work, as well as how this affects the design of the application. The non-functional requirements should define a set of minimal requirements related to the quality of the application.

5.5 Specification and Design

The defined requirements will be used in creating a specification of the logic of the application, as well as in defining navigational structures within the application.

The specification will be defined as use cases that describe how the user will interact with the application, and how the application will respond.

Each use case will be connected to a functional requirement, and one or more non-functional requirements.

6 Application Design

This chapter introduces the planned mobile news application design, and gives an overview of how the implemented solution will work. The purpose of this chapter is to give an introduction, and define the specifics of the application by creating requirements. These requirements are the base for a set of evaluation criteria that are made to reflect the requirements.

Section 6.1 gives a Context Model of the system which displays the structure of the system. Section 6.2 defines the functional and non-functional requirements for the application, and section 6.3 describes the associated use cases. Section 6.4 describes the characteristics the application should be evaluated on.

6.1 Context Model

Here, a model of the application by use of the Context Modeling Language (CML) discussed in section 4.1.1 is described. The purpose of the model is to create an image of the different entities that participate in the contextual aspects of the system.

Figure 6.1 shows a CML model of the application.

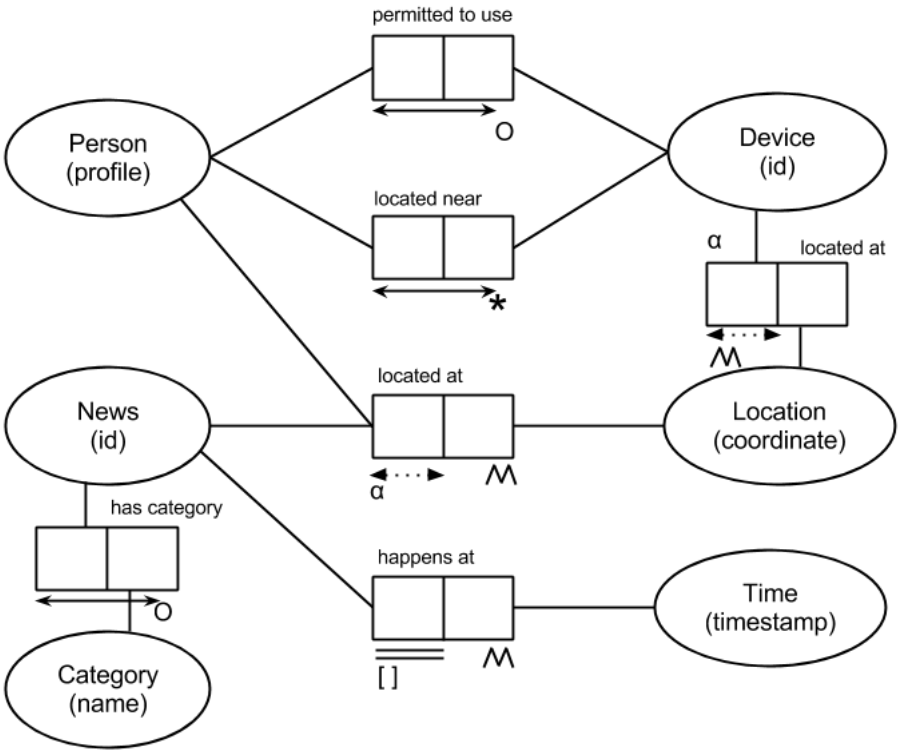


Figure 6.1: CML

As seen in the figure, there are six object types, *Person*, *Device*, *News*, *Location*, *Category* and *Time* represented as ellipsises. Each of these object types has their representation values in parentheses, e.g., News has the representation value *id*. Each of the ellipsises are connected together with fact types and boxes that represents roles.

To give an example from the figure, an instance of the *has category* fact type can look something like has_category[12345, Sport], where the first value represents the id of the news article, and the second value represents the category of the news article.

Each of the fact types have annotations that signals if the fact type is sensed, static, profiled, derived, temporal or ambiguous/alternative. In the previous example, the News articles category fact type are profiled (user-supplied).

Both the person and the device has both sensed and alternative facts,which means that for example a device can have multiple locations.

6.2 Application Requirements

This section presents the requirements for the mobile news application. These requirements are an essential part of the development process, and will be used in the project evaluation.

The following sections will describe both the functional and non-functional requirements that have been defined for the mobile news application. Since the focus of this thesis is to evaluate HTML5, and not the resulting application, some of the requirements will not be considered since they represent properties that are tightly coupled to the implementation of the application, and not the technologies the application is built on. The requirements that will be explored shallowly, or omitted are for the most part non-functional.

6.2.1 Functional Requirements

The previous section showed how the different objects in the news aggregation system are connected by contextual information. By analyzing this model, it is possible to create a list of functional requirements needed for the application to provide basic functionality.

The news application should deliver relevant news articles to a user, and the delivery of the articles are based on filters that personify the content delivered to the user. The filters are supplied by the user, and can be used to request specific content from the news recommendation system, e.g., a user can tell the application that it should only fetch news that are described as *sport*. These filters are used in the server-side implementation of the news recommendation system described in section 4.3.2. The news application should act as a client in a client-server infrastructure.

The items listed below describes the functional requirements of the appli-

cation:

1. **Show all news articles**
   1.1. Show articles in list
   1.2. Show articles on map
2. **Show a single news article**
   2.1. Show detailed article
   2.2. Show article on map
3. **Enable filtering of news articles**
   3.1. Enable filtering based on user-supplied information
   3.2. Enable filtering based on sensed information

All the above requirements are equally important to achieve an application that works with the news recommendation system, but they have been prioritized to satisfy any dependencies between the requirements. A further exploration of these requirement can be found in section 6.3.

6.2.2 Non-functional Requirements

Mobile applications are dependent on non-functional requirements to function properly. Since mobile applications usually have more constraints than traditional desktop applications, it is important to consider these requirements in every aspect of the mobile application. The most important constraints a mobile device has are related to: device resources, networking, and user interface design.

Some of the most important non-functional requirements when developing mobile applications are performance issues like efficient use of device resources, responsiveness and usability. These requirements allow users to use an application smoothly. Section 6.4 will describe the evaluation criteria related to the non-functional requirements of the application.

6.3 User Interface Design

The requirements defined in section 6.2.1 and section 6.2.2 have been formulated into use cases [33] that describes the overall functionality of the application. The use cases are explained in detail in Appendix A, but table 6.1 is a summary of each use case, its corresponding description and the requirements it satisfies.

Use case	**Description**	**Requirement**
UC1	A user opens the web application, and a list of news articles are shown.	F1.1
UC2	A user opens a news article from the list of articles.	F2.1
UC3	A user opens a news article in a map.	F2.2
UC4	A user opens the map with news articles.	F1.2, F3.2
UC5	A user opens the settings menu, and selects a settings option.	F3.1

Table 6.1: Use cases

For each of the above use cases and the functionality they describe, screen mock-ups have been made that illustrate the proposed design of the application, and the accompanying behaviour inside and between screens. The rest of this section will describe each use case from the perspective of a screen mock-up and describe how the news application will satisfy both functional and non-functional requirements.

The use case **UC1. Opening a list of news articles** describes the opening of the application. The main page is the first page the user will see when the application is started and contains all the news articles that have been fetched. Figure 6.2 shows a sketch of the main page, where the news articles are listed.

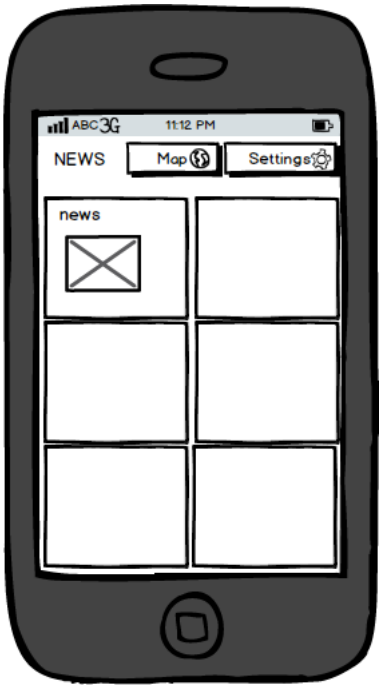


Figure 6.2: Listed news

The way a user opens the application is dependant on how the application is deployed. In the case of a standard mobile web application, it will be reachable by inputting an URL into the mobile browser, which will start the application. In the case of a hybrid application, the news application will have its own icon and be available in the application menu on the mobile device. When the application is started, the news articles should be listed as clickable areas in two columns in the main region of the site. The news articles will have different background colors based on the category of the article. These categories are Technology, Sport, Entertainment, Economy etc.

The rest of the main page consist of a header region with two buttons, the filter menu and the options menu, which will be explained later.

As for non-functional requirements, both performance and robustness is important for the main page to work properly. The application should load the news articles with little waiting time, and if there isn't any news

articles available, or if there is no network access, the application should give the user correct feedback.

The use case **UC2. Opening a single news article** describes how the user access a single news article, and gets access to more information regarding this article. How the user opens the article is integrated and illustrated in Figure 6.3.

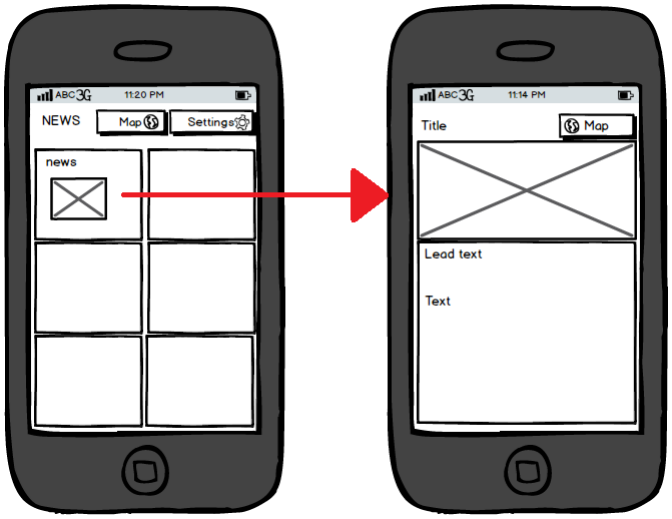


Figure 6.3: Extended news

When the user touches the wanted news article in the main page, the site refreshes and the article is extended. In the extended view, news title, picture, lead text and text are information from current article that will be displayed. The user can from here choose to go back to the main page, or view the location of the current news article in a map. This is done by the map button in the header. If the article don't have any location, the map option should be missing.

The non-functional requirements performance, robustness and quality are all important for the extended news to perform properly. The wanted news article should open without delay, and the application should give proper feedback if there occur any errors. Desired news article should also be easy to select from the main page.

The use case **UC3. Opening a single news article in a map** describes the opening of a map with the localization for a single news article. It is realized with a map button in the article that redirects the user to the map view. A marker in the map gives the localization of the news article, together with a marker of the users current location. Figure 6.4 illustrates how the application will look like when the user touches the map button, and the map is displayed.

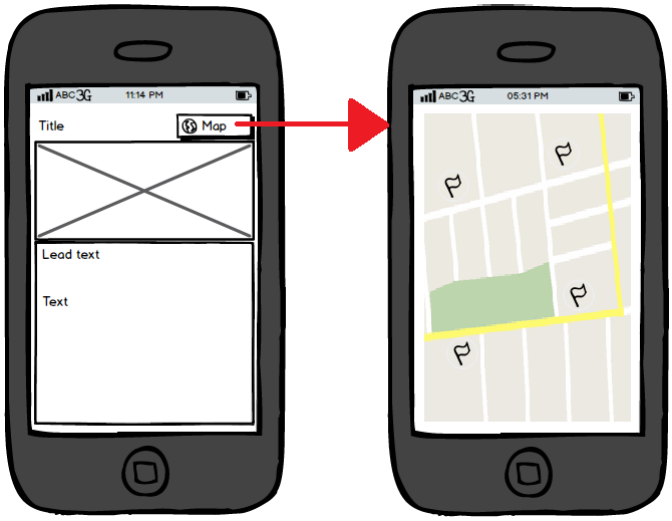


Figure 6.4: Extended news in map

The user can touch the screen to change the zoom level of the map to see a wider or smaller area. From here, the user can go back to the chosen article.

The non-functional requirements performance, quality and robustness are all important for the application to function properly when the map is loaded. The application should respond and open the map without any big delays. The map option should be intuitively and easy to understand, which is done by a map button. If the map or location is not available, the application should failure in a proper way.

The use case **UC4. Opening news articles in a map** describes the opening of a map with markers for the location of news articles. Figure 6.5

illustrates how the application reacts when the user choose to open news articles in a map by clicking on the map option.

Figure 6.5: News in map

In the settings menu, there are a setting that decides if the map contains markers for all news, or news located nearby the users current location. Both the settings menu and the map option is accessible in the main page. The user can choose to open the map directly in the main page, or in the settings menu if the user wants to change the map option before opening the map.

The map page in the application will look the same for both all news and local news, but with different zoom area. The users current location will also be in the map. From here, the user can choose to to go back to the main page, open the settings menu and change the settings, or open a single news article from the map.

Quality, robustness and performance are all non-functional requirements that is important for the application to function properly when the map is opened. It should be easy to choose and switch which map option that is suitable, and the map should be easy accessible. It is not necessary that the user choose the map option each time the map is opened, this is

therefore in the settings menu. The robustness requirement is important since application should give proper feedback if there occur any failures. Performance is included since the application should have low response time, and minimum delays.

The use case **UC5. A user opens the settings menu, and selects a settings option** describes how user-supplied information affects the application. Figure 6.6 illustrates how the application will look like when the user choose to open the settings menu for the news application.

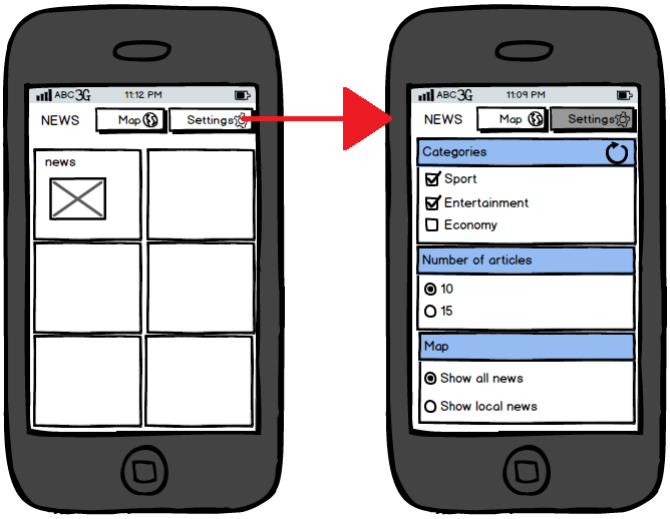


Figure 6.6: Settings menu

The user can select one or more categories from the categories list, to show in the news article list. This makes it easier to filtrate out relevant and interesting information for the user. The categories of the news articles are given their own individual color, so that they are easier to recognise in the news list. After the user has chosen the desired news categories, the main page will refresh and only display news from the chosen categories.

The user can also choose to change how many news articles that will appear in the news list in the main page. Radio buttons are used to illustrate that the user can only choose one number of articles that will be displayed.

At last in the settings menu, the user can choose the map option as described earlier.

Performance, Robustness and Quality are all non-functional requirements that is required for a good experience when the user changes settings in the application. When the user choose to change the settings, the application should respond and set the changes immediately. In the settings menu, settings that is not possible should not be possible to choose by the user. As example, radio buttons is used so that the user only can choose one of some alternatives when setting number of articles and the map option. The radio button is used to set categories, since the user can choose one or more categories. It should not be possible to set zero categories. The settings option should be easy to recognize and easy accessible.

6.4 Evaluation Criteria

The preceding sections defines requirements for both the functional and non-functional aspects of the mobile news application. By using these, it is possible to define criteria that the application should be evaluated on. After the application has been created, these evaluation criteria will be used to evaluate the suitability of the application from the context of a mobile news application.

The application should be evaluated from a developers stand point, which means that the evaluation criteria that are not related to the previously defined requirements describe evaluation criteria that are mostly based on the development process.

Since the purpose of the application is for consumption of information, and not production, some aspects of mobile applications do not need to be evaluated, e.g. user experience related to inputting of data.

It is expected that the application should perform equally on all mobile platforms, i.e. Android, iOS, and Windows Phone.

This section describes the characteristics of Quality and Development.

6.4.1 Quality

The quality of a system can be determined by a set of *characteristics* and *sub characteristics* that are further decomposed into *attributes* that are measurable. These attributes can be computed by using a *metric* [34].

The International Organization for Standardization (ISO) has defined ISO/IEC 9126 [35] which defines a quality model for use as a framework in software evaluation. There are some definitions in this model that do not fit with the evaluation of a technology for developing a product, since the quality model defines a lot of characteristics that are defined in the context of use. In addition, each characteristic contains a sub-characteristic that is related to regulatory compliance that is not needed in the evaluation performed in this thesis.

The following sections will describe de characteristics and sub-characteristics from ISO/IEC 9126, and how they will be used in the evaluation of HTML5 as a cross-platform development strategy. The purpose of using an adapted quality model for evaluating is to gain a sense of what quality characteristics a finished product developed using HTML5 will have.

FUNCTIONALITY

> *The capability of the software product to provide functions which meet stated and implied needs when the software is used under specified conditions* [35].

The sub-characteristics of the quality characteristic functionality are described below:

- **Suitability** - If the product provides appropriate functionality.
- **Accuracy** - If the product provides the appropriate results with needed precision.
- **Interoperability** - If the product is able to interact with other systems.

- **Security** - If the product can protect data and information from unauthorized entities.

The above sub-characteristics describe how the functionality of a software product can be evaluated according to its functionality. When considering these characteristics in the case of the mobile news application, it is possible to omit *Accuracy* from the evaluation.

Accuracy in the mobile news system is not delivered by the mobile news application, but from the news recommendation system, so it is out of the scope of the evaluation.

The *Suitability* will in this case be the products ability to provide appropriate functionality, where the functionality will be the necessary functions to create an application that can work as a news application.

The *Interoperability* of the mobile news application will in this case be its ability to interact with the news recommendation system. This is important for the system to function properly.

The *Security* can be measured by the applications ability to protect the privacy of its users. Since the application is meant for consumption of information, the security aspects from a users point of view are not as important as in other applications.

RELIABILITY

> *The capability of the software product to maintain a specified level of performance when used under specified conditions* [35].

The sub-characteristics of the quality characteristic reliability are described below:

- **Maturity** - If the product is able to avoid failures.
- **Fault tolerance** - If the product is able to tolerate failures.
- **Recoverability** - If the product is able to recover from failures.

The reliability of a system is important for it to work properly. This in-

cludes how robust the system is, and that the system reacts in a satisfactory way.

When considering the above characteristics, it is clear that they are defined to represent how a product is made, and the products ability to act on failures. If these characteristics are to be used in evaluating HTML5 applications, they must be adapted.

If we prepend the definition of the reliability characteristic to represent the ability of the product to facilitate functions or tools so that a system can provide *Maturity*, *Fault tolerance*, and *Recoverability*, it is possible to evaluate the quality of the mobile news application.

USABILITY

> *The capability of the software product to be understood, learned, used and attractive to the use, when used under specified conditions* [35].

The sub-characteristics of the quality characteristic usability are described below:

- **Understandability** - If the product enables the user to understand how the product solves a particular task.
- **Learnability** - If the product enables the user to learn how the product works.
- **Operability** - If the product enables the user to control the product.
- **Attractiveness** - If the product is attractive.

The usability of a mobile application is important because the screen sizes of various mobile devices are smaller than on desktop computers. This means that new navigational paradigms must be used to accomplish tasks.

When considering the *Understandability* of a product, it is possible to think of how an application resembles other applications with the same purpose and applications on the same platform. In native development this is a big deal, and guidelines and rules are created to give applications similar

look and feel. *Learnability* is also based on these same concepts, and the familiarity of applications may enable users to learn how the applications work quickly.

When evaluating these two characteristics, the important aspects are how HTML5 and the web enables a developer to create solutions that enable understandability and learnability.

The two last characteristics, *Operability* and *Attractiveness* are mainly dependant on specific implementation details of the application, and are not interesting in the context of this evaluation.

EFFICIENCY

> *The capability of the software product to provide appropriate performance, relative to the amount of resources used, under stated conditions* [35].

The sub-characteristics of the quality characteristic efficiency are described below:

- **Time behaviour** - If the product provides appropriate response and processing times under normal operation.
- **Resource utilization** - If the product provides an appropriate use of resources under normal operation.

The efficiency of a product is a characteristic that has to be measured in use. It mainly consists of measurements related to time when reading and writing information, and how the resources of the system is used.

When evaluating efficiency, the measurements have to be done on an already implemented application to get an insight in how the resources of a device are used, and how this impacts the normal operation of the application.

The measurements that are important in mobile applications have big implications on the responsiveness of the application, which can have a big effect on usability. When evaluating these characteristics, the delays in

time, and the use of resources should not impact the user experience of interacting with the application.

MAINTAINABILITY

> *The capability of the software product to be modified. Modifications may include corrections, improvements or adaptation of the software to changes in environment and in requirements and functional specifications* [35].

The sub-characteristics of the quality characteristic maintainability are described below:

- **Analyzability** - If the product can be diagnosed for failures and deficiencies.
- **Changeability** - If the product can be modified.
- **Stability** - If the product can tolerate modifications without unexpected effects.
- **Testability** - If the product can be tested.

The above characteristics describe properties of a product that are important during the development phase, and after the product has been deployed. In the context of mobile web applications, these properties can be evaluated in the form of available tools and libraries that can be used to diagnose and test the application.

The *Stability* characteristic describe properties of a system that is very dependant one the implementation of the system in question. During this evaluation, the stability of the mobile application is of less interest than the characteristics that can lead to a stable system, e.g., if there are ways to test the software.

Of the above characteristics, *Testability* and *Analyzability* can be evaluated by looking at the available options in the form of tools and frameworks that can be used to achieve maintainability. The last characteristic, *Changeability* describes a quality that describes how patchable, or updateable the

system is.

PORTABILITY

> *The capability of the software product to be transferred from one environment to another* [35].

The sub-characteristics of the quality characteristic portability are described below:

- **Adaptability** - If the product can be adapted for other environments.
- **Installability** - If the product can be installed in a specific environment.
- **Co-existence** - If the product can co-exist with other products in the same environment.
- **Replaceability** - If the product can be used instead of another product in the same environment.

The portability characteristic is important in the context of this case study. A big part of the motivation behind evaluating HTML5 as a cross platform strategy is the portability of the applications.

The *Adaptability* and *Installability* characteristics are a very important part of the evaluation since they can have such a large impact on the final evaluation.

In contrast, if the application can *Co-exist* or replace other products is of less importance.

6.4.2 Development

A big part of the evaluation process should focus on the development of the mobile application. The development phase takes into consideration characteristics of HTML5 that are not related to the quality of the end product, but characteristics that describe the ecosystem around HTML5 and cross-platform development of applications.

PLATFORM ECOSYSTEM

The ecosystem describes the whole environment around an application, and is an integral part of the development process. In native application development, the ecosystem is mostly controlled by the providers of the platforms, which mean that they have an incentive to keep documentation, tutorials and guidelines updated and of good quality.

An important part of the ecosystem is *technical costs*. These include the costs of implementation, and all other costs of things that need to be done in the development process. Some examples of technical costs are the learning of new platforms, and tools that will be used. Some platforms require that their

Another important aspect of the ecosystem is the *resources*. Good resources can produce extension, documentation, tutorials, guidelines and other resources that can be helpful when developing a new system.

7 Realization

The implementation of the mobile news application consists of an HTML5 web application which can run in a standard browser as well as a mobile browser, and a hybrid application that is run in a native container.

The purpose of the application is to deliver news to a user based on the context of the user, i.e., the location and other user-specified parameters. These news articles should be viewable as markers on a map, or as a traditional list of articles. The user-supplied parameters are filters related to the desired category and number of articles.

There exist a large number of possible tools and frameworks that can be used to develop HTML5 cross-platform applications, and help provide consistent HTML5 applications across multiple mobile browsers. The following sections describe the process of developing the applications, and some of the considerations that had to be made to satisfy the requirements that were defined in section 6.2. Since there exist a lot of frameworks, only the most relevant will be presented in the following sections.

7.1 Tools and Libraries

Since the server-side implementation is not part of this thesis, the only implementation is the HTML5 web application and the hybrid equivalent. To be able to create these applications, a range of tools and libraries was used to help in the development process. The following section will describe these.

Section 7.1.1 will describe the development environment that was used to create the applications. Section 7.1.2 will describe the process of choosing an appropriate framework as a basis for the architecture of the applications.

Section 7.1.3 will describe the process of choosing appropriate frameworks an libraries for the user interface of the applications.

7.1.1 Development Tools

The development of the mobile news applications was done using the Eclipse IDE, which is an open source development platform for building, deploying and managing software [25]. It exist a number of different Eclipse packages available for download, and the package used for developing the application was the Eclipse Java EE IDE for Web Developers, the Juno version.

Since the application run client-side and are based only on web technologies, the application could be tested right in the web browser. All web browsers has a developer console, where it is possible to inspect what is going on in the browser. That includes the stylesheets used, the markup of the page, script files and requests and response. In this study, the Google Chrome browser has been used as environment for the debugging since it is based on WebKit [36] like the browsers on Android and iOS.

To test the application in its real environment, some mobile devices was used, these are listed in table 7.1.

Device	platform	version
Samsung galaxy S2	Android	4.2.2
Samsung Galaxy Note 2	Android	4.1.2
Apple iPad 2	iOS	6.1.3

Table 7.1: Development testing environments

Because of the difficulties of creating iOS and Windows Phone applications without access to a Mac and Visual Studio development environment, an emulator that can emulate the applications on the different platforms was used. This was the Opera Mobile Emulator [37], which runs the same code as its mobile phone versions. In the emulator, it is possible to choose one of the preconfigured popular phones, such as Samsung Galaxy S3, or custom

made your own emulator environment by setting Resolution, Pixel Density, User Interface, User Agent String, Window Scale, and Arguments.

7.1.2 Architecture

One of the consequences of creating a mobile application using HTML5 and its related technologies is that the client-side becomes thicker and more dependant on logic. This means that the client-side code becomes just as important as the server-side code when considering the performance and maintainability of the application.

As a result of this, large numbers of frameworks and libraries that help developers with the structure of the application by providing JavaScript constructs and functions that contribute to the development process.

To be able to satisfy some of the defined requirements, such a framework is required to create code that is testable and maintainable to reduce bugs and performance issues in the application.

Framework	**Size**	**StackOverflow**	**GitHub**
Backbone [38]	6.3Kb + 4Kb	9030	14267
Ember [39]	51Kb	3911	7072
Knockout [40]	14Kb	5744	3733
Angular [41]	29Kb	6096	10073

Table 7.2: JavaScript web application frameworks

Table 7.2 shows four popular web frameworks and some statistics related to them. The size column represents the file sizes of the source files that have to be included with the web application. In the case of Backbone, the utility library *Underscore* has to be included. The StackOverflow column represents the number of questions on StackOverflow, which in addition to the number of GitHub stars might indicate the popularity of the frameworks, as well as be an indicator of the community of each of the frameworks.

7.1.3 User Interface

The user interface implementation of the mobile news applications try to satisfy the non-functional requirements related to usability as defined in section 6.4.1. This section focused on the *understandability* and *learnability* of an application as important aspects from a user standpoint.

If an application wants to achieve these properties, it should be able to deliver applications that the user expects, i.e, an iOS application should look and behave like an iOs application, and an Android application should look and behave like an Android application.

Since the HTML5 ecosystem does not provide any predefined components and guidelines that adhere to the standards of the native platforms, developers need to utilize third-party libraries that provide these capabilities.

Framework	**Android**	**iOS**	**Windows Phone**
jQuery Mobile	Yes	Yes	Yes
Sencha touch	Yes	Yes	Yes
jQTouch	Yes	Yes	No

Table 7.3: Mobile UI frameworks

Table 7.3 shows three popular mobile ui frameworks that provide user interface elements and animations. Two of these frameworks support all the platforms that this thesis aims for.

One of the problems with these UI frameworks is that they do not provide a native look and feel, but provide their own version of something that looks like bits and pieces from different platforms. As an example, each of the UI frameworks provide a back button at the top of the application that resembles the back button from iOS. If an HTML5 application was developed for the iOS platform, this would not be a problem, but on most Android phones, the back button (either a hard or soft button) at the bottom of the screen is used. One of the options one has when using UI frameworks like this is the ability to customize the user interface with

themes and other user interface elements, but it takes more work to create something that is slightly different than what the framework defines.

7.1.4 Device Features

In the context of the mobile news application, it is important that some of the device features are available, especially when developing hybrid applications. To be able to utilize the functionality of a mobile device, an API to the device features is needed.

Framework	**Architecture**	**Device**	**UI**	**Free**
Apache Cordova	No	Yes	No	Yes
M-Project	Yes	Apache Cordova	jQuery Mobile	Yes
App Framework	Yes	Yes	Yes	Yes
Titanium	Yes	Yes	Yes	Yes

Table 7.4: Mobile web frameworks

Table 7.4 shows four mobile web frameworks that offer varying degrees of functionality for the development of mobile web applications. Some of these frameworks offer complete solutions consisting of the desired development constructs needed to create cross-platform applications. The table shows that both App Framework, and Titanium offer full solutions, while Apache Cordova only works as a library for accessing the device features of a mobile device.

7.2 Implementation

At the beginning of the development process, some decisions for how the news application should be developed had to be made. To be able to look at two different aspects of mobile html5 application development, both a pure web application and a hybrid application was implemented.

Instead of using one of the mobile specific JavaScript frameworks, the choice fell upon a more self-deployed method. For creating the application architecture, Backbone was used in conjunction with Backbone.Marionette [42] to ease the construction of views in the application. These JavaScript libraries are small and work in all web browsers, which reduces the footprint of vendor code in the application. To create the views and design of the application, the Amazium CSS library [43] was used to provide the correct views settings for different screen sizes.

This approach enables the developer to have full control of the complexity of the application by only including libraries that do one thing. The disadvantage with this is that it requires the developer to write more code (especially in complex applications), but in the case of an information consumption application that only has to show data, it should be sufficient.

The reason why none of the UI frameworks, i.e, jQuery Mobile, Sencha Touch and jQTOuch was chosen is because they are big frameworks with many dependencies and peculiarities. And even though these frameworks advertise that they can provide the native look and feel for Android, iOS and Windows Phone, it requires extra work even when they are used. When using this frameworks, you also lock yourselves to their way of developing applications, and maintenance in the case of updates is more difficult.

Developing the hybrid application for the Android platform was done using the Apache Cordova framework and the Android SDK. Apache Cordova was chosen because of the wide support of tools and the big community around it.

7.2.1 Architecture Design

The application is developed in HTML5, which means that JavaScript files interact with the device, HTML files adds structure and CSS3 files defines the page appearance. The hybrid application is created with HTML5, just like the web application, and it is wrapped with Apache Cordova.

Because of browser limitations, a proxy server had to be made to route all AJAX-requests to the news recommendation system. This is because of the same-origin security policy in a browser which limits cross-domain AJAX-calls.

The proxy server is a bare bones Node.js server with a single REST endpoint that routes the request from the mobile news application to the news recommendation system.

DATA FLOW DIAGRAM

To illustrate how the data flows through the system, a data flow diagram is made.

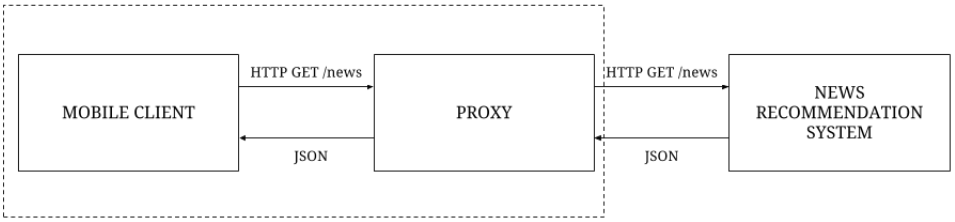


Figure 7.1: Data flow diagram

Figure 7.1 shows how the the implemented application works in relation to the rest of the system.

First, the mobile client sends a HTTP GET request to the proxy server which accepts the query parameters specified in the client, and builds an URL that the news recommendation system can use. As an example, the client sends a request to

{server-endpoint}/stories?rows={rows}&category={category}

The query parameters are read and used to build another URL as specified in section 4.3.3. The proxy then calls the endpoint of the news recommendation system, which returns a JSON data object that is routed through the proxy and into the client.

At the client, the JSON object is parsed into the desired format, and showed as a list of news articles for the user.

STATE MACHINE DIAGRAM

A State Machine Diagram models the behaviour of a system, and specifies which states the system can be in [33]. This technique is used and illustrated in figure 7.2, which illustrates how the application behaves at all times.

The initial pseudostate indicates which state the applications is in when the application is started. The initial pseudostate has an arrow that points to the initial state, the main page.

The figure illustrates that the application can be in four states, the home page, extended article, the settings menu and the map view. The states are connected by lines that gives the transitions, which tells how the application changes from state to state. Each transition is written in the following way, where each part is optional: *trigger-signature [guard]/activity*, where trigger-signature is the event that triggers the change of state, [guard] is an boolean condition that must bet true for the change to take place, and the activity is what happens in the transition [33].

In State Machine Diagrams there are usually an final state that indicates the termination of the system, but in this application there does not exist any final state, because it is a web application and the application is terminated just by closing the browser or by opening another URL.

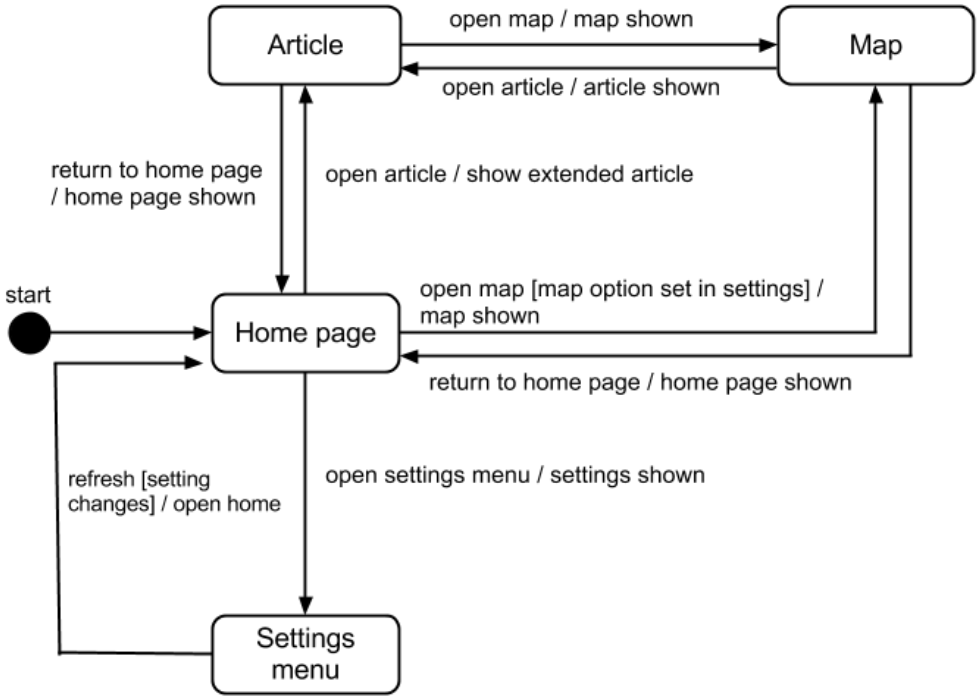


Figure 7.2: State machine diagram

INPUT METHOD

When the use wants to perform some actions in the news application, it requires some input from the user. The chosen input method is buttons interaction, where the user touches buttons and the application makes action based on which button that is touched. As an example, if the user opens the settings menu and selects a category, the application responds and fetches news from that category instantly.

In addition to buttons, when selecting articles from the list of articles, touch presses are also used to select the desired article.

7.2.2 The Implemented News Application

This section presents how the news application was implemented, and provides a detailed representation of the user interface of the final result. For

each of the use cases covering the functional requirements described in appendix A, the implemented solution for these requirements are presented and described.

The screenshots in the following sections are only from the web implementation of the application since any screenshots from the hybrid application would look identical.

OPENING A LIST OF NEWS ARTICLES

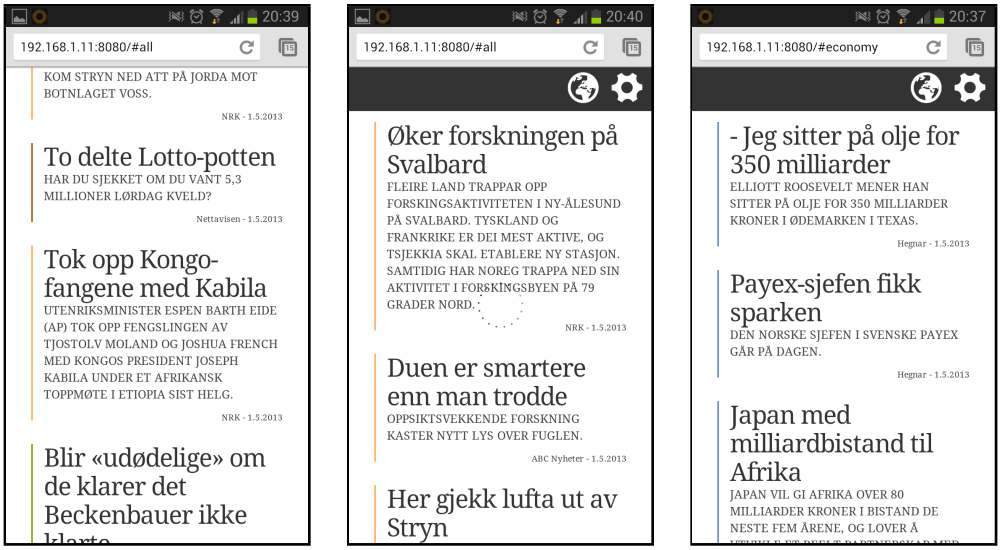
When the user opens the main page of the application, the user will see the news listed. When accessing the application, there is no need for signing in or other user-supplied information to see this page. All that is required is that the user enters the URL for the application in the web browser, or downloads it from the app store in the hybrid approach.

Figure 7.3 shows the main page that appears when the user opens the application, and the list of news articles appears. The figure shows three different stages of the main page. Figure 7.3a shows how the main screen of the application is when the user first enters the application. The application is filled with the ten most recent news articles. If the users selects another category, a loading indicator will pop-up as shown in figure 7.3b, and then the main screen will be loaded with articles from the selected category as shown in figure7.3c.

As shown in the figures, the news articles are listed listed in a row, where the article title and lead text are shown for each of the news articles. The article also has a publisher and published field that shows the recency of the article.

OPENING A SINGLE NEWS ARTICLE

When the application user clicks on a article in the news feed in the main page, it will appear a new page with expanded information about this article. In addition to the news article title and lead text, pictures and full text will be displayed in the expanded view. Figure 7.4 shows how this



(a) All categories (b) Loading (c) Filtered on category

Figure 7.3: The main page of the mobile news application

was implemented.

Figure 7.4: An opened news article

The user can scroll the page to read the whole text bu using swipes on the screen.

OPENING A SINGLE NEWS ARTICLE IN A MAP

When the user is in the detail view of an article, it is possible to press the globe at the top of the screen to show that article on a map. As shown in figure 7.5, the map shows the locations of the news article as markers on the map.

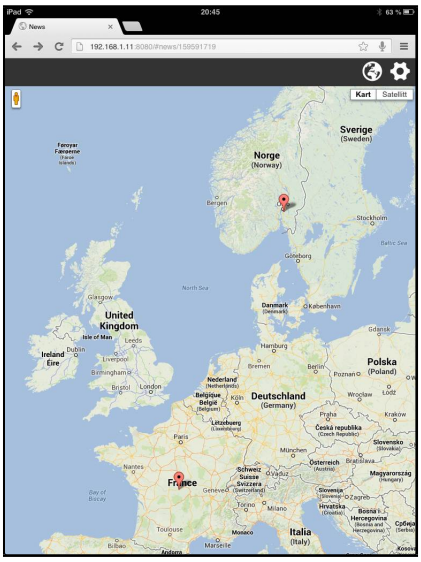


Figure 7.5: Showing the locations of an article on a map

The Google maps API was used to plot and show news on a map. When a news article is extended, both the users location and the story location is given in the map.

OPENING NEWS ARTICLES IN A MAP

When the user is in the list of all news articles, the user can choose to plot all news on a map by pressing the globe at the top of the screen. Figure 7.6 shows the implemented solution of all news on the map.

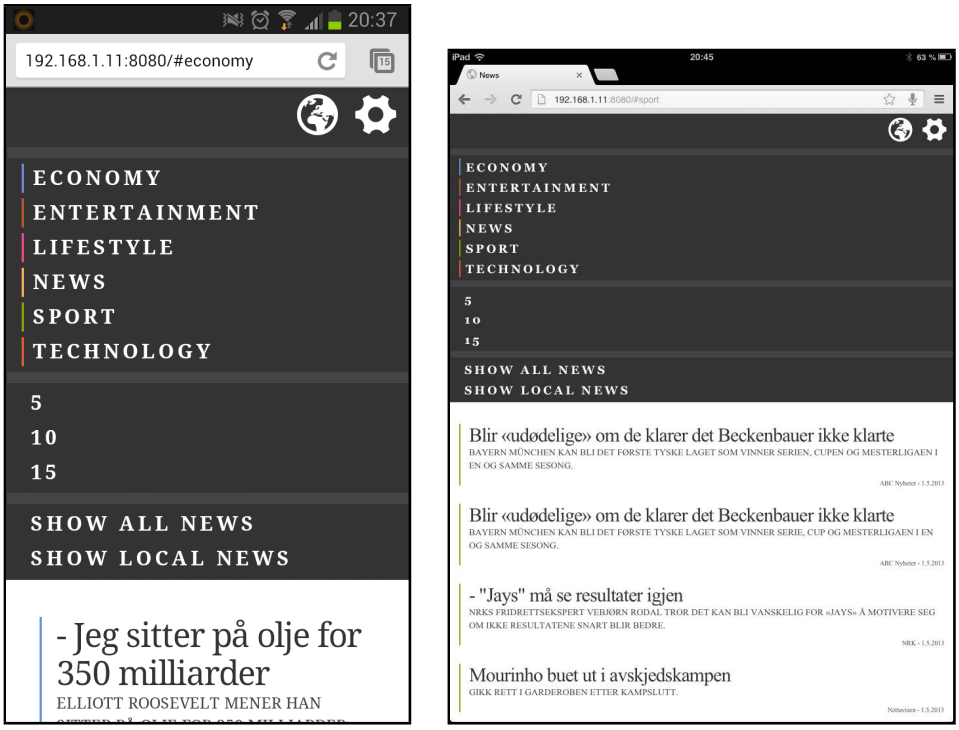
Figure 7.6: Data flow diagram

FILTER NEWS ARTICLES

The filter menu is accessible from all screens in the application. When the user click on the setting icon in the top right of the application, a drop down menu with filter option appears from the top of the screen. In the settings menu, the user can choose which categories to display in the news feed on the main page. The user can choose one of the categories Economy, Sport, Technology, Lifecycle, Entertainment, and News.

After choosing desired categories, the main page will refresh and only display news from the desired category.

Also, the user can choose how many articles to display in the main menu.



(a) Samsung Galaxy Note II (b) iPad

Figure 7.7: A drop down settings menu

Figure 7.7 shows how the filter menu is implemented. the purpose of the menu is to allow the user minimal interaction with the application to get the desired outcome.

7.3 Improvements

There are some improvements that can be done with the news application as it is today. Due to the time constraints, some ideas and design improvements had to be set aside. These improvements concerns improvements on the implemented application that would lead to a better quality and user experience.

One of the other functionality that were described in the functional requirements were the ability to filter news on location. As of now, the mobile news application returns all news. This is also present in the setting menu,

where the buttons for filtering on the locality of news are present, but not implemented.

When a news article has been selected and shown in a map, or if all the articles have been shown in the map, the user should be able to select an article from the map to open the article.

8 Evaluation

In this chapter, the strengths and weaknesses of developing a HTML5 application for the given use case is evaluated. The evaluation is based on the criteria from Section 6.4.

The criteria is evaluated by the use of the LSP method discussed in Section 8.1. First each of the criteria is divided into measurable components with the system requirement tree. Then for each of the web and hybrid approach, each component is given a elementary criteria and a elementary preference between 0 and 1 based on information collected on the Internet and experience from the development process. At last, the aggregation is executed.

8.1 Logic Scoring of Preference

Logic Scoring of Preference (LSP) is a quantitative evaluation methodology for evaluating software quality [44]. The LSP methodology consists of the following steps:

1. Define an evaluation standpoint.
2. Identify evaluation criteria.
3. Determine the elementary preferences.
4. Aggregation of elementary preferences.

The first step is the definition of the *evaluation standpoint*. This is the standpoint from where the system is evaluated from, e.g., software users and software developers have different criteria, and therefore the system needs to be evaluated from their different points of view.

This evaluation standpoint is the base for the creation of a *system requirement tree*, which is used to systematically identify software evaluation

criteria components. These components are decomposed into aggregation blocks until the components can not be divided into smaller components. These components of the lowest level of the tree are directly measurable, and are called *performance variables*.

Each of these performance variables must have a defined *elementary criteria* that determines the *elementary preference* of the variable. This elementary preference determines the level of satisfaction for each value of the evaluated performance variables. The elementary preference score is a value between [0,1] or [0,100%] [44]. The elementary preference can be calculated as follows:

$$E_i = G_i(X_i)$$

where E is the elementary preference, G is the function for calculating E, X is the score of a performance variable, and i is the number of a particular feature. This relationship can also be represented as a *preference scale* [44].

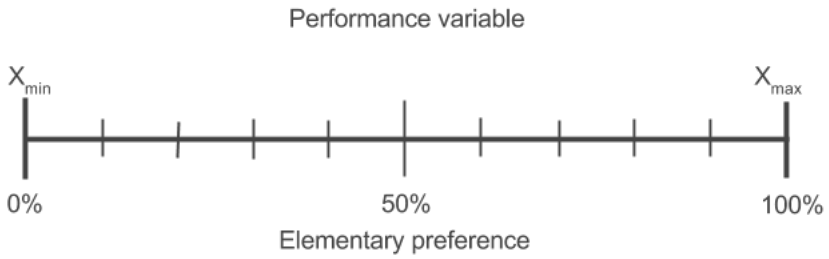


Figure 8.1: A preference scale

Figure 8.1 shows a preference scale for calculating an elementary preference. The top of the figure represents a score for a performance variable, where X_{min} represents the minimum score, which result in an elementary preference of 0%, and X_{max} represents the maximum score which will result in an elementary preference of 100%.

All of the elementary preferences present in an aggregation block are used to calculate the preference score of the feature that contained the elementary preferences. These preference scores can again be used to calculate the preference score of the feature above and so on until the entire system requirement tree has been aggregated into a global preference score. The global preference is defined as:

$$E = L(E_1, \ldots, E_n)$$

where E is the global preference, L is the function for evaluating E, E_n is preference n, and n represents the number of preference scores. The L function calculates the output preference by the formula:

$$e_0 = (W_1 {E_1}^r + \ldots + W_k {E_k}^r)^{1/r}, W_1 + \ldots + W_k = 1$$

where e_0 is the output preference score, W is the weight of the elementary preference, E is the elementary preference of a performance variable, k is the number of performance variables in an aggregation block, and r is a conjunctive/disjunctive coefficient of the aggregation block.

The weights from the L formula are defined as a fraction of 1, and signify the importance of a preference value.

The aggregation function represents a process of *logic aggregation of preferences* that iterates over the system requirements tree from the leaf nodes to the root, and aggregates the aggregation blocks by using suitable *logic aggregation operators*.

There are five logic aggregation operators with specific logic properties [44]:

- **Simultaneity** - When two or more preference variables must be present simultaneously.
- **Replaceability** - When two or more preference variables can be replaced by alternatives.

- **Neutrality** - When two or more preference variables can be grouped independently.
- **Symmetric** - When two or more preference variables affect the evaluation in the same logical way.
- **Asymmetric** - When mandatory preference variables are combined with desired or optional preference variables.

Simultaneity, replaceability and neutrality are three special cases of the *Generalized Conjunction/Disjunction* (GCD) function. Where neutrality (A) represent arithmetic mean, simultaneity (C) represent conjunction, and replaceability (D) represent disjunction. C are categorized as C+, CA and C- to represent if they are strong, medium or weak conjunctions. Further, C– is between A and C-. The same is the case for the replaceability function.

Figure 8.2 shows these function, from total disjunction to total conjunction.

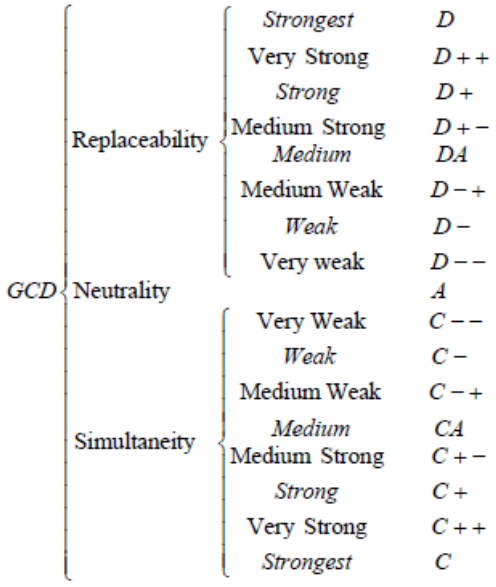


Figure 8.2: The GCD function: 17 levels and their symbols [44]

8.2 System Requirement Tree

The system requirement tree is based on the quality characteristics from section 6.4.1 and 6.4.2.

1. **Quality**
   1.1. Functionality
   1.2. Reliability
   1.3. Usability
   1.4. Efficiency
   1.5. Maintainability
   1.6. Portability
2. **Development**
   2.1. Ecosystem

The above list is a summary of the quality characteristics and characteristics related to the development process. The following sections will present each characteristic, and define element criteria for each of them. The goal is to achieve measurable characteristics, and aggregate these into an elementary preference score.

The aggregation process also contains relative weights for each characteristic. These help in determining which characteristics that are more important than others. These weights are subjective to a certain degree, and might affect the final result.

8.2.1 Functionality

The measurable characteristics related to functionality describe functionality that must be present for the mobile news application to be able to satisfy the functional requirements specified in section 6.2.1.

The following list describe the sub-characteristics of functionality, where the leaf nodes of the tree describe the measurable characteristics.

1. **Suitability**

1.1. Storage
1.2. Connectivity
1.3. Geolocation
1.3.1. GPS
1.3.2. Network
1.4. User interface
1.4.1. Back button
1.4.2. Resolution handling
1.4.3. Orientation handling
2. **Interoperability**
2.1. Data formats
2.2. Communication protocols
3. **Security**
3.1. Privacy

SUITABILITY

The suitability characteristics describe functionality that is commonly available in the development of native applications. Some of these characteristics such as location are essential in the creation of the mobile news application, and should be weighted more importantly than others. For most of these functions, the evaluation can be done on the availability of the function on Android, iOS and Windows Phone.

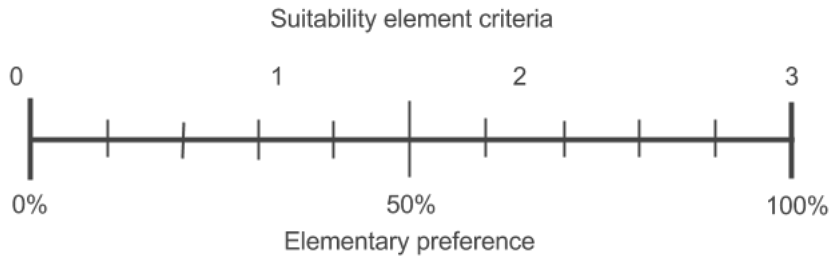


Figure 8.3: Suitability preference scale

Figure 8.3 show how the elementary preference of the suitability character-

istics are calculated in a preference scale. If a functionality is not present on any platforms, the preference score will be 0, and 1 if it is present on all platforms.

INTEROPERABILITY

The interoperability characteristic describes if it is possible for the system to communicate with other systems. If two or more systems are able to communicate, they need to share communication protocols and data formats. In the case of the mobile news application, this is an essential characteristic because the application needs to communicate with the news recommendation system.

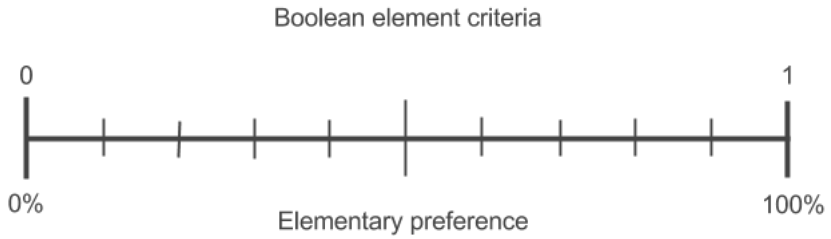


Figure 8.4: Boolean characteristics preference scale

The two measurable characteristics that have been decomposed from the interoperability characteristic are *data formats* and *communication protocols*. The news recommendation system delivers information over the HTTP protocol, so the element criteria for the communication protocol characteristic can be formulated as a boolean property, where an element preference of 0% signifies that HTTP is not supported. The preference scale from figure 8.4 show the graphical representation of this preference.

The news recommendation system sends its data over the following protocols: XML, JSON, CSV and HTML. This means that the *data formats* characteristic can be evaluated based on if these formats are available from the mobile news application. The element criteria for data formats can be formulated as follows: *one point for each data format supported*.

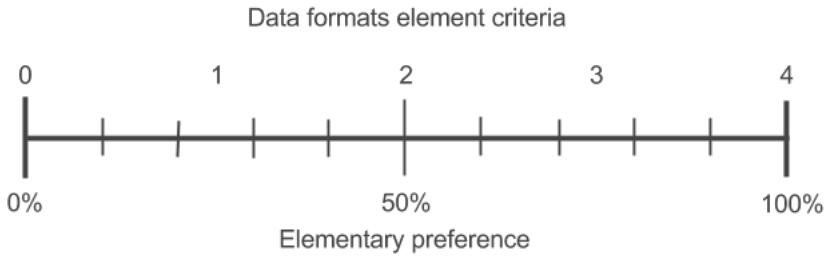


Figure 8.5: Data formats preference scale

Figure 8.5 shows the preference scale for the data format characteristic. If all specified data formats are available, the element preference will be 100%.

SECURITY

The security characteristic describes the ability of the mobile application to protect it from unauthorized access. Since the mobile application is meant for consumption of information, the security implications of most transactions are small. One important concern is still the privacy of the user, since the user has to share location information.

The *privacy* characteristic can be evaluated if the geolocation implementation follows the W3C Geolocation standard [45] which states that *User Agents must not send location information to Web sites without the express permission of the user*. This means that the preference scale will be a equal to the suitability characteristics as defined in figure 8.3. To get a full score, the geolocation API of the mobile browsers must implement the API correctly.

8.2.2 Reliability

The reliability characteristic describes non-functional requirements of the system so that the system can perform at normal operation even when the system experiences errors and failures.

The following list describe the sub-characteristics of reliability, where the leaf nodes of the tree describe the measurable characteristics.

1. **Maturity**
   1.1. Offline capability
2. **Recoverability**

MATURITY

The maturity characteristic describe the ability of the mobile application to avoid failures. One of the characteristics that are helpful in this sense is if the mobile application has *offline capability*. If the application can be run offline, connectivity issues can be avoided.

For offline capability to be present, both the availability to store state in the client, and the ability to monitor the connection of the client is necessary. The element criteria of the offline capability characteristic can be described as follows:

- For each of the platforms Android, iOS, and Windows Phone
  - 1 point for storage functionality
  - 1 point for connectivity functionality

Figure 8.6 shows the preference scale for the offline capability preference. If both storage and connectivity are present for all platforms, the elementary preference will be 100%.

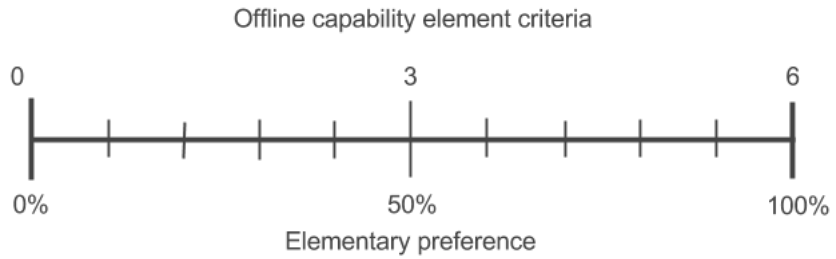


Figure 8.6: Offline capability preference scale

RECOVERABILITY

The Recoverability capability describe the ability of the application to recover after failures. If the applications is to recover, it should be able to resume a previous state from before the failure.

To be able to recover, the application has to be able to save state locally which can later be rolled back to in the case of failures. This requires that the application has storage functionality and can store state information. The element criteria of the recoverability characteristic is similar to that of the offline capability characteristic, but only requires the storage functionality. The element criteria can be described as *for each of the platforms Android, iOS and Windows Phone, add one point if the platform supports storage.*

8.2.3 Usability

The usability characteristic describes non-functional requirements of the system so that the system is useful for the user that is going to use it.

The following list describe the sub-characteristics of usability, where the leaf nodes of the tree describe the measurable characteristics.

1. **Understandability**
   1.1. Navigation
   1.2. User interface

UNDERSTANDABILITY

The understandability characteristic describes the systems ability to be understood. Some of the characteristics that can be derived from understandability are navigational paradigms, i.e., if the mobile application can apply known navigational patterns that the user is used to, such as touch gestures.

The characteristic *Navigation* describes the navigational patterns available

in the mobile application. Some of the patterns that are used in native applications are touch gestures and soft buttons. The element criteria for the navigation characteristics can be described as follows:

- For each of the platforms Android, iOS, and Windows phone
  - 1 point if touch gestures are supported
  - 1 point if soft buttons are supported

Figure 8.7 shows the preference scale for the navigation characteristic. If both touch gestures and soft buttons are available for all platforms, the elementary preference of navigation is 100%.

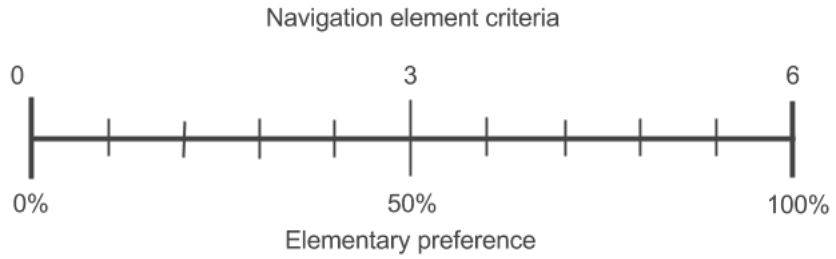


Figure 8.7: Navigation preference scale

The *User interface* characteristic describes how the user interface elements and overall user experience can be created in the application. In native development, there exists guidelines and rules for how an application should look, and how different parts of the application should be created. The element criteria for the user interface characteristic can be described as follows:

- For each of the platforms Android, iOS, and Windows Phone
  - 1 point if there exist guidelines for creating the user interface
  - 1 point if there exist user interface libraries that define predefined components and styles

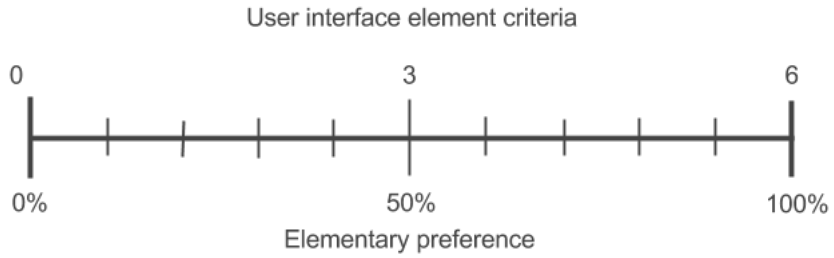


Figure 8.8: User interface preference scale

Figure 8.8 shows the preference scale for the user interface characteristic. If there exist both guidelines and user interface libraries, the elementary preference of the user interface characteristic is 100%.

8.2.4 Efficiency

The efficiency characteristic describes non-functional requirements of the system so that the system can provide a good performance.

The following list describe the sub-characteristics of efficiency, where the leaf nodes of the tree describe the measurable characteristics.

1. **Time behaviour**
   1.1. Latency
   1.2. Response times
2. **Resource utilization**
   2.1. CPU
   2.2. Memory

TIME BEHAVIOUR

The time behaviour characteristic describes the capability of the system to provide appropriate response and processing times under normal operation. Some of the characteristics that can be derived from time behaviour are Latency, Response time and Throughput.

The characteristic *Latency* describes time delays in the system. If there

exist a big latency, the data transmission rate is low. Latency is measured in milliseconds, and a high latency slows down the system and results in a poor system performance.

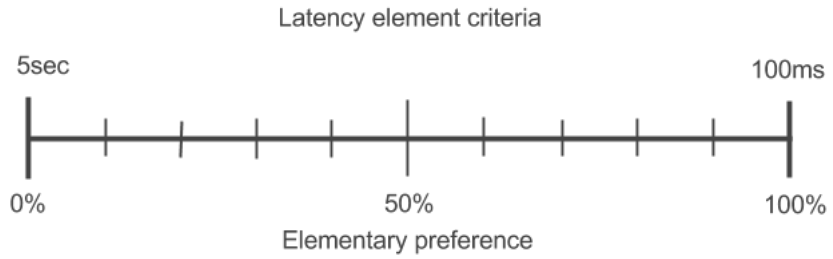


Figure 8.9: Latency preference scale

Figure 8.9 shows the preference scale for the latency characteristic. If the data latency is 100 ms, the latency is not noticeable, and the element preference will be 100%. If the data latency is 5 seconds, the latency is noticeable, and the element preference is 0%.

The *Response time* characteristic is the time it takes from a user action to a system response. The response time is measures based on how many seconds it takes before the user gets a response from the system.

In easy tasks, like movements and button clicks, the application should respond in 100ms. Then the system reacts instantly, and the system response time is good. On the other side, if the system responds too fast, the user cannot keep ut with the response.

In normal tasks, like site navigation, the response time should be under 1 second. If the system uses 1.0 seconds to response, the response time is noticeable, but it is tolerable.

In time consuming tasks, like loading data, the application should respond in under 10 seconds. If the system uses 10 seconds to response, the users attention is affected, but the response time is acceptable due to the amount of data loading.

Figure 8.10 shows the preference scale for the response time preference. If the response time is 100ms, the elementary preference will be 100%. For easy tasks, X_{max} are 200ms, meaning that if the response time is 200ms, the elementary preference will be 0%. For normal task, response time over 3 seconds are not acceptable, and the elementary preference will be 0%. For time consuming tasks, 10 seconds give a elementary preference on 0%.

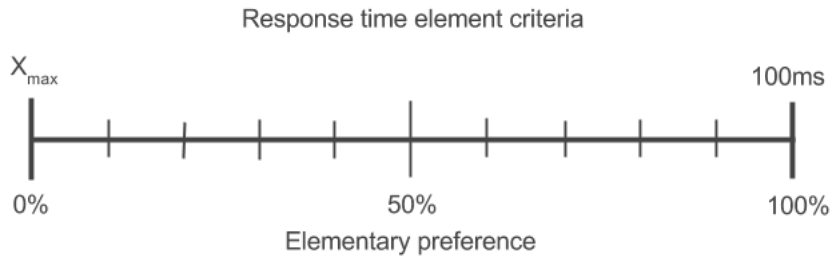


Figure 8.10: Response time preference scale

RESOURCE UTILIZATION

The Resource utilization characteristic describes the capability of the system to handle resources. This is divided into the two sub characteristics CPU and Memory.

The *CPU* characteristic describes the CPU utilization by the news application. Effective use of the CPU allows the application to run faster and more efficiently. The CPU usage is given in a percentage scale (0%-100%), where, 0% is the lowest CPU usage, and 100% is the highest CPU usage. High CPU usage may result in reduced response time, while low CPU usage shows that the device is able to run the program easily. To evaluate the news application CPU utilization, the highest score is set to 100% and the lowest score is set to 5%, since there must be some CPU usage when using the application. Figure 8.11 shows the preference scale for the CPU preference.

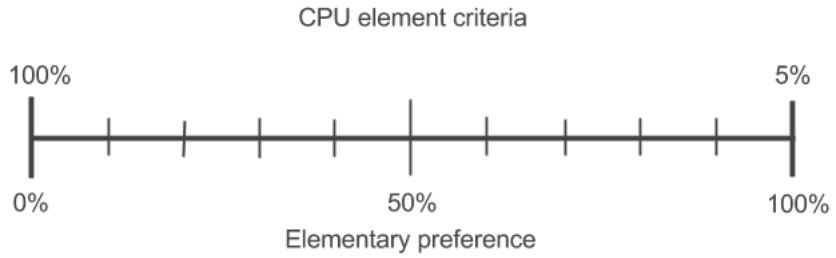


Figure 8.11: CPU preference scale

The *Memory* characteristic describes the amount of memory used by the news application. If the memory usage is high, the device may feel slow, and the user experience gets poorer.

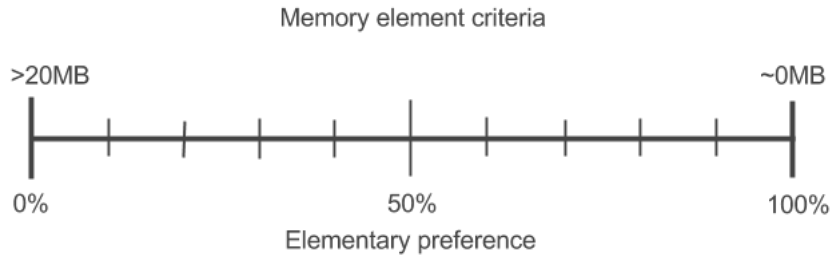


Figure 8.12: Memory preference scale

Figure 8.12 show how the elementary preference of the memory preference is calculated. The application should use less than 20MB of the memory, and more than or approximately 0MB. If the application uses approximately 0MB, the element preference will be 100%, and if it uses 20MB or more, the element preference will be 0%.

8.2.5 Maintainability

The measurable characteristics related to maintainability describe properties that must be present for the application to be maintainable. That includes properties that is important both during the development phase, and after the deployment of the application.

The following list describe the sub-characteristics of maintainability.

1. **Analyzability**
2. **Changeability**
3. **Testability**
   3.1. Availability of test libraries
   3.2. Availability of test runners

ANALYZABILITY

The analyzability characteristic describes the systems ability to be analyzed. This ability is often in the form of diagnostic utilities that can be used to monitor quality aspects of a system such as performance and fault tolerance. To be able to evaluate the analyzability of an HTML5 application, one can look at the availability of tools for diagnostics.

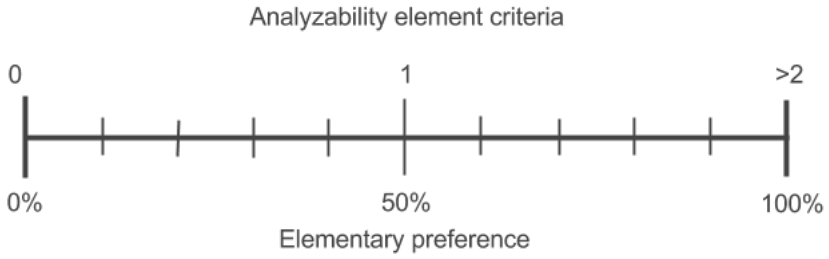


Figure 8.13: Analyzability preference scale

Figure 8.13 shows the preference scale of the analyzability characteristic. The element criteria can be defined as:

- One point for each diagnostic tool that has:
  - Support for Android, iOS, Windows Phone and HTML5
  - Support for monitoring and reporting of crashes
  - Support for behaviour statistics
  - Support for error notifications

If one can find more than two diagnostic tools that fulfill these criteria, an elementary preference of 100% is achieved.

CHANGEABILITY

The changeability characteristic describe if it is possible to update or patch the application. This is an important characteristic because it enables the developers to fix problems that can lead to down times in the application. To further limit the characteristic, the updating of patching of the application must happen in *reasonable time* to limit potential down times.

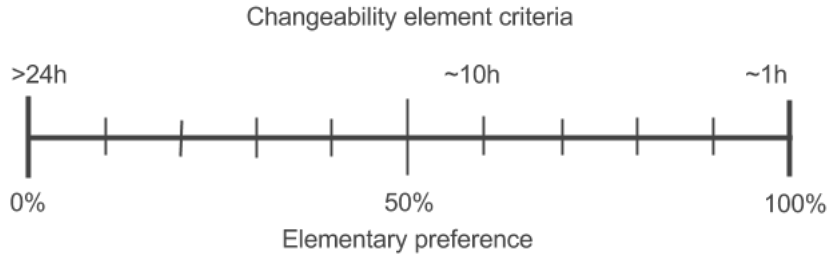


Figure 8.14: Changeability preference scale

The preference scale of the changeability characteristic is described in figure 8.14. The figure shows that if an update takes a couple of hours, the elementary preference is high, and if the update takes longer than 24 hours, the elementary preference is low.

TESTABILITY

The testability describes if it is possible to test the application, and how easy it is.

The measurable characteristics that have been decomposed from the testability characteristic are the *availability of test libraries* and the *availability of test runners*. To be more specific, the test libraries need to be able to provide the basic constructs for creating unit tests, and the test runners need to be able to run test in different environments, and from different environments.

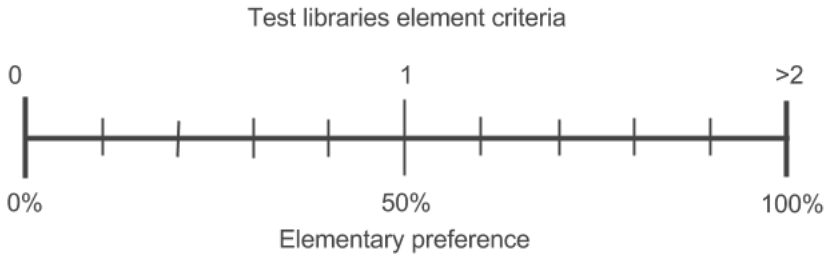


Figure 8.15: Availability of test libraries preference scale

Figure 8.15 shows the preference scale of the availability of test libraries characteristic. The evaluation of test libraries is based on the functionality of the libraries. A solid test library contains at least assertions, mocking capability and support for asynchronous tests. The element criteria is defined as:

- One point for each test library that has:
  - Assertions
  - Mocking
  - Asynchronous support

If there are two or more test libraries that fulfill these criteria, the elementary preference is 100%.

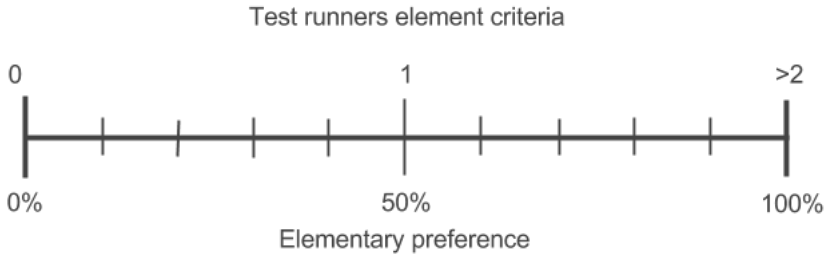


Figure 8.16: Availability of test runners preference scale

Figure 8.16 shows the preference scale of the availability of test runners characteristic. To measure the availability of test runners, the element criteria is defined as:

- One point for each test runners that has:
  - Support for most major test libraries
  - Support for all browsers
  - Support for headless testing

If there are two or more test runners that fulfill these criteria, the elementary preference is 100%.

8.2.6 Portability

The measurable characteristics related to portability describes functionality that is important for cross-platform mobile applications. Since the application should be port to different platforms, the portability characteristic is a important part of the application.

The following list describes the measurable sub-characteristics of portability.

1. **Adaptability**
2. **Installability**

ADAPTABILITY

The Adaptability characteristic describes if the application can be adapted to other platforms. How much work that is required for adapting the application to different platforms, is the basis of the evaluation of adaptability.

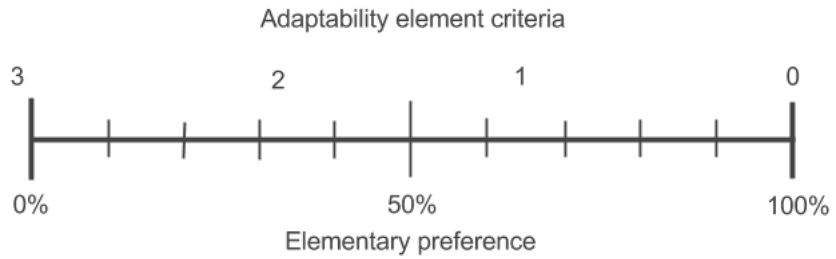


Figure 8.17: Adaptability preference scale

Figure 8.17 show how the elementary preference of the adaptability preference is calculated. Since the application should at least be accessible on the four platforms, Android, iOS, and Windows Phone,the preference score will remove one point for each platform that is easy adaptable. If there is no work needed for adapting the application to the different platforms, the element preference will be 100%. The preference scale takes into account that the application is already created.

INSTALLABILITY

The Installability characteristic describes if the application can be installed on target platforms.

The installability characteristic can be evaluated based on if the application can be installed on a platform or not. This means that for each of the platforms, the application should be installable in a way such that the user can choose the application from a list of applications. For each of the platforms this is possible, a point is given. Figure 8.18 shows the preference scale for this elementary preference. To achieve an elementary preference of 100%, an application should be possible to install on Android, iOS, and Windows Phone. An elementary preference of 0% represents a situation where an application is not installable at all.

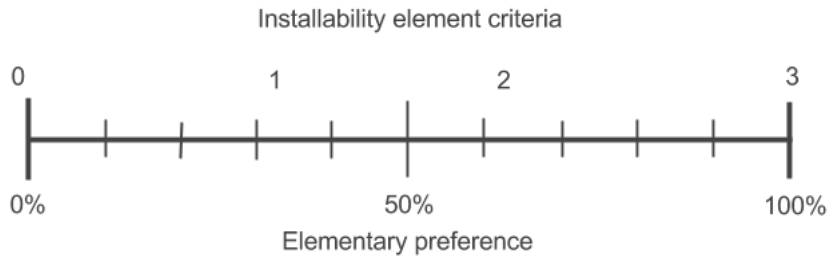


Figure 8.18: Installability preference scale

8.2.7 Platform Ecosystem

The platform ecosystem describes the surroundings of the development platform. In this case, these surroundings are defined by the possibilities and limitations the platform enforces upon the developer.

To be able to have create measurable evaluation criteria, the below characteristics are based on properties that are more or less present in native development platforms.

The following list describes sub-characteristics of the platform ecosystem, with measurable leaf-nodes.

1. **Technological costs**
   1.1. Languages
   1.2. Tools
2. **Resources**
   2.1. Community
   2.2. Development Center
   2.3. Guidelines
   2.4. Tutorials
   2.5. Documentation

TECHNOLOGICAL COSTS

Technological costs are the costs of using the technologies in a development ecosystem. Since specific platform often have very specific requirements that define the development process, this can be a limitation for the developer.

The two measurable characteristics that have been decomposed from the technological costs characteristic are *Languages* and *Tools*. If a development ecosystem defines a full stack of languages that have to be used to create an application, the developer in the worst case have to learn the entire stack of languages. In the case of cross-platform applications, this means that a developer may be required to gain a complete knowledge of

several development ecosystems.

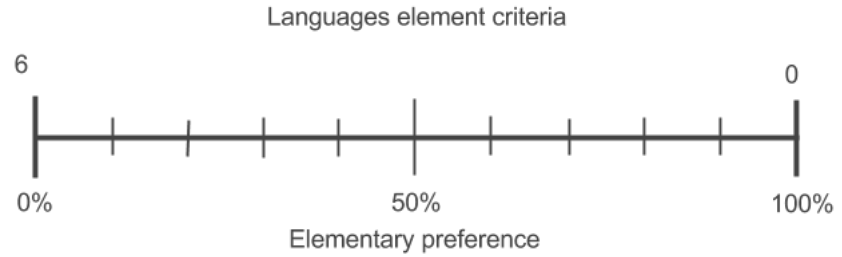


Figure 8.19: Languages preference scale

Figure 8.19 shows the preference scale for the languages characteristic. This characteristic can be evaluated based on how many languages that have to be used to target Android, iOS, and Windows Phone:

- For each of the platforms Android, iOS, and Windows Phone
  - 1 point for each language that need to be used to be able to develop on the platform.
  - 1 point for mandatory APIs that have to be used for each platform.

The tools characteristic is important since some development ecosystems require the use of specific tools for development. This characteristic can be evaluated as follows:

- For each of the platforms Android, iOS, and Windows Phone
  - 1 point for each proprietary platform or software that needs to be used.

Figure 8.20 shows the preference scale for the tools characteristic. The figure shows that the number of proprietary platforms and software reduces the elementary preference of the characteristic.

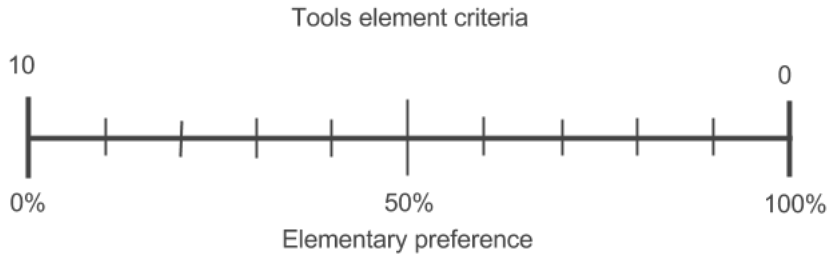


Figure 8.20: Tools preference scale

RESOURCES

The resources related to a platform ecosystem is the main source of documentation and references for a developer. This is where a platform vendor describes the platform and its properties.

Five measurable characteristics that have been decomposed from the community characteristic are *Tutorials*, *Development center* and *Guidelines*, *Documentation*, and *Community*.

The community of an ecosystem is the group of people and resources that surround the ecosystem. A good community can be of good help for a developer that is just starting out in a new development ecosystem, or in situations that require that a developer needs help for solving problems.

- 3 point if the community has a large and active user base.
- 1 point if the community has a mailing list.
- 1 point if the community has a wiki page.

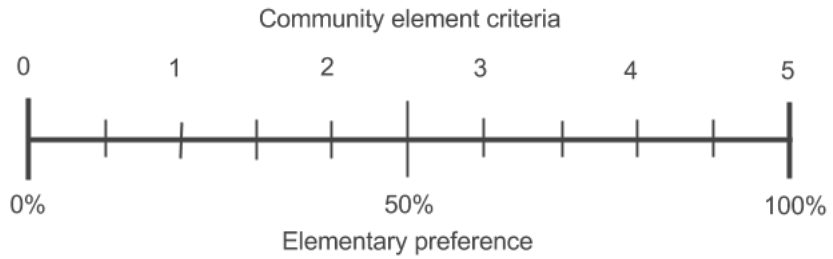


Figure 8.21: Community preference scale

Figure 8.21 shows the preference scale of the community characteristic. An elementary preference of 100% is only achieved if a community fulfills all the criteria.

The tutorials characteristic describes the availability of related tutorials in a community. To limit the broadness of this characteristic, it will only consider *official tutorials* created by the vendor of a platform. The tutorials characteristic will be evaluated on the number of available tutorials and if the tutorials apply to different levels of accomplishment:

- 1 point for easily accessible and structured tutorials.
- 2 points for tutorials aimed at beginners.
- 1 point for tutorials aimed at a intermediary level.

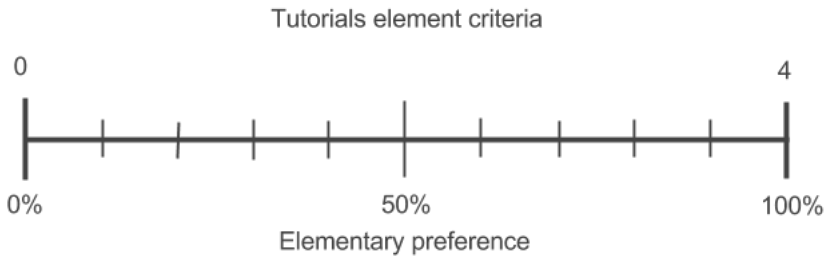


Figure 8.22: Tutorials preference scale

Figure 8.22 shows the preference scale for the tutorials elementary preference.

A development center is a centralized resource platform for gaining new information about developing on a platform, and is used as a resource for developers that need to find specific information about a platform. The development center characteristic can be evaluated as a boolean elementary preference.

An essential part of a platform is good documentation, especially if the platform has an accompanying API. The documentation characteristic considers the availability of documentation that describes standardized

and implementation specific properties of a platform. The following list describes how the documentation characteristic should be evaluated:

- 1 point if the documentation covers the APIs of the platform.
- 1 point if the documentation describes the architecture of the platform.
- 2 point if the documentation describes any peculiarities and subjective conventions of the platform.

Figure 8.23 show the preference scale for the documentation elementary preference that describes the above lists of criteria.

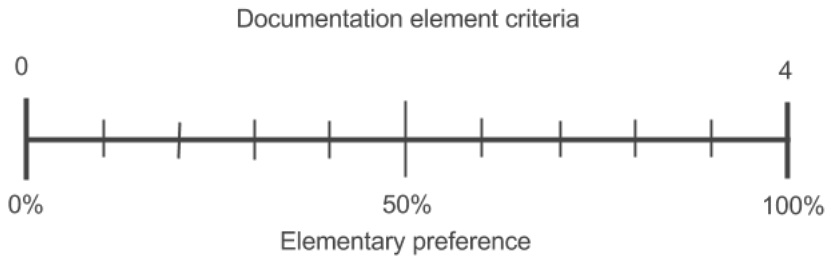


Figure 8.23: Documentation preference scale

The last characteristic is the guidelines characteristic that describes if the platform defines rules or guidelines that describe how the platform is meant to be used, or has to be used. To evaluate this characteristic, the following criteria are used:

- 1 point for user interface guidelines.
- 1 point for code style conventions.
- 1 point for architectural guidelines.

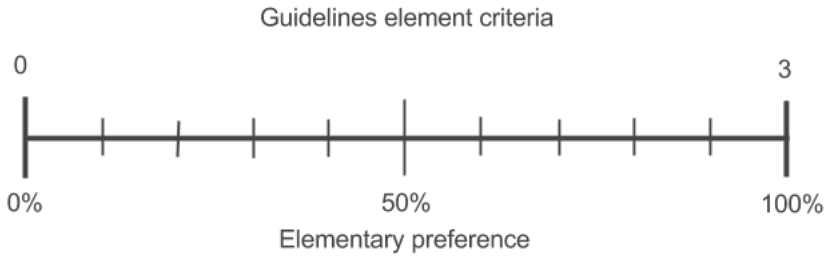


Figure 8.24: Guidelines preference scale

Figure 8.24 shows the preference scale for the guidelines characteristic.

8.3 Elementary Preference Aggregation

The elementary quality preferences are structured to allow the computing of partial and global logic aggregation of preferences. Figure 8.25- 8.31 shows the partial logic aggregation structure for functionality, reliability, usability, efficiency, maintainability, portability and ecosystem, while figure 8.32 shows the global aggregation structure. For all of the requirements, the global quality preference represents the global degree of satisfaction.

The process of aggregation starts at the bottom, and aggregates upward to the global quality preference is found. In the aggregation structure, every aggregation has a CLP operator that is calculated using the calculation method in this article [44]. The actual values used when aggregating the partial preferences are listed in Appendix B.

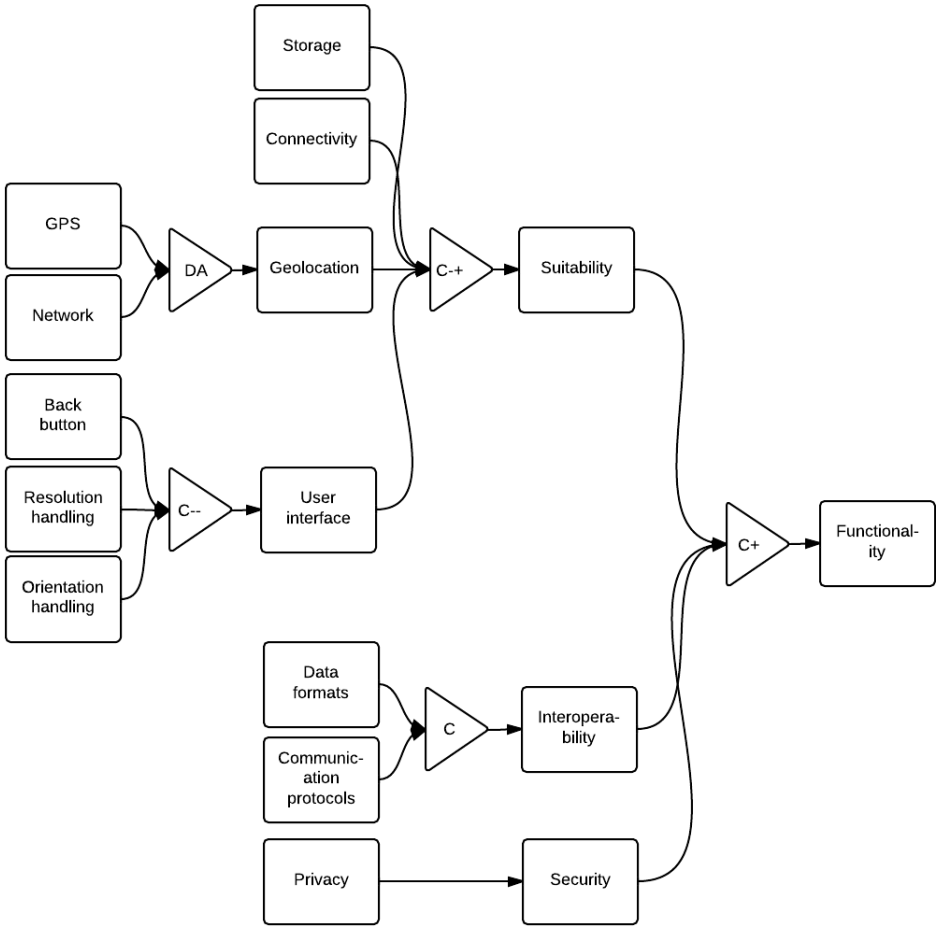


Figure 8.25: Partial logic aggregation for functionality

Figure 8.25 shows the aggregation structure of the functionality characteristic. When calculating the partial aggregation of preferences, the resulting value for HTML5 is 92,5% which signals a really high degree of satisfaction. The degree of satisfaction for the hybrid application was also really high at 98%.

Both the web and hybrid approach scored high on most element criteria. The only characteristic that stands out is the *connectivity* characteristic, which is lacking in some browser implementations on mobile devices.

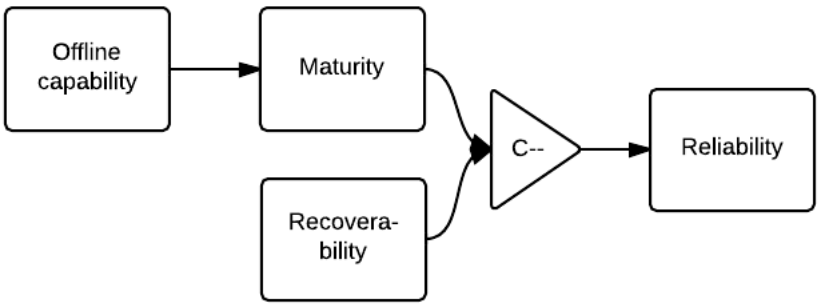


Figure 8.26: Partial logic aggregation for reliability

Figure 8.26 shows the aggregation structure of the functionality characteristic. The values for the partial aggregation of preferences was a degree of satisfaction of 91% for HTML5, and 100% for the hybrid application.

The differences between the two approaches here lies in their offline capability support, and since this is better in a hybrid implementation, it gets a small boost.

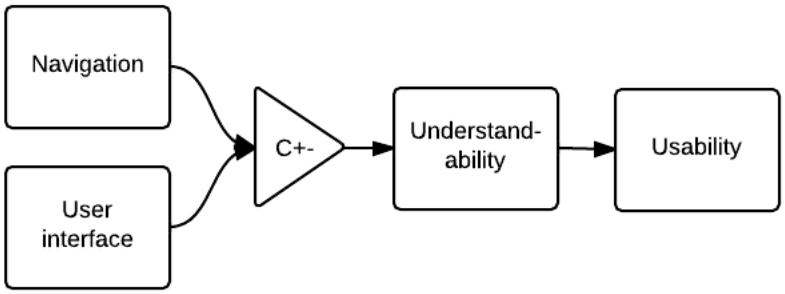


Figure 8.27: Partial logic aggregation for usability

Figure 8.27 shows the aggregation structure of the functionality characteristic. Here, the degree of satisfaction was a bit lower than for the two preceding aggregations. The partial preference of usability was equal at 58% on both the HTML5 application, and the hybrid application.

The partial aggregated preferences for usability are equal because the two approaches uses the same technology for creating the user interfaces, and therefor receives the same limitations as well.

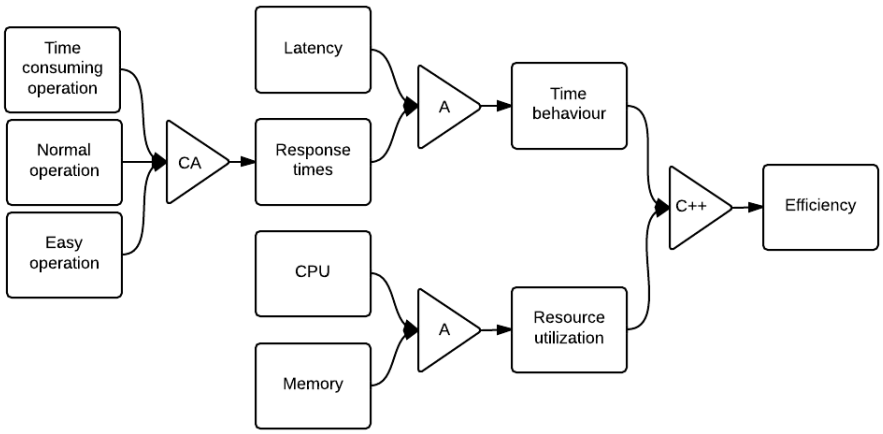


Figure 8.28: Partial logic aggregation for efficiency

figure 8.28 shows the aggregation structure of the functionality characteristic. For the HTML5 application, the partial aggregation of preferences for HTML5 was at 55%. It proved difficult to test the same test cases on the hybrid application, so the degree of satisfaction will be the same as for the HTML5 application.

The biggest factor in the efficiency aggregation were the memory use of the application, which was poor, and the latency. Both of these had really low scores (0.3 and 0.4) which impacts the aggregated preference greatly.

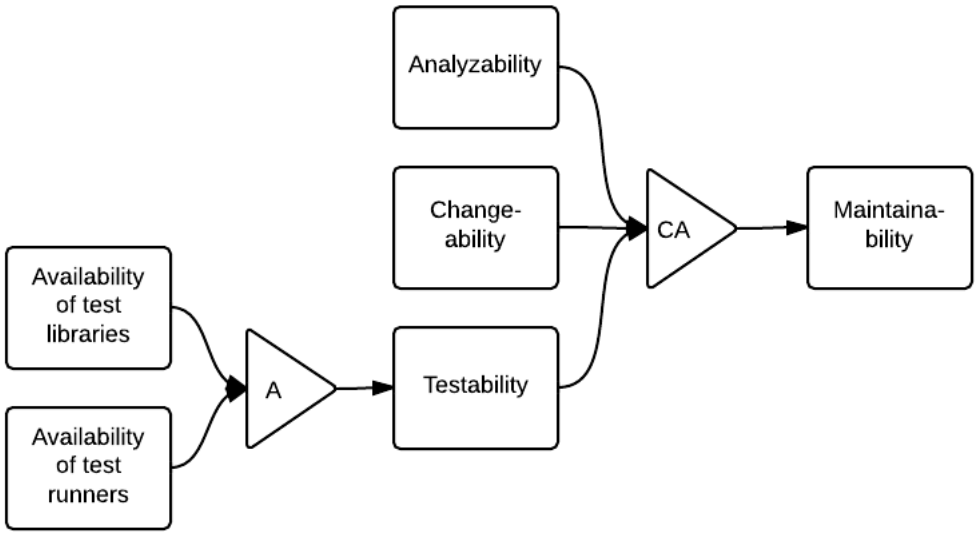


Figure 8.29: Partial logic aggregation for maintainability

Figure 8.29 shows the aggregation structure of the maintainability characteristic. This characteristic has the biggest spread in aggregated preferences. For HTML5, the partial preference was 72%, and for the hybrid application it was quite less at 56.8%.

The reason the hybrid application gets a low partial preference in this case is mostly because it has to target three platforms that have varying time demands related to updating and patching applications. Since this is not an issue in a web application, the web application scores much better.

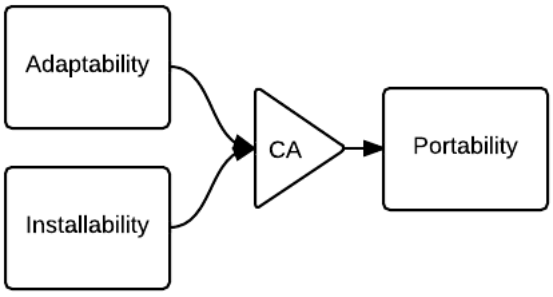


Figure 8.30: Partial logic aggregation for portability

figure 8.30 shows the aggregation structure of the portability characteristic. The portability characteristic also shows an interesting result. Here, them HTML5 application only scores 43.7%, while the hybrid application scores 100%.

Here, we have the opposite situation from the maintainability preference. Since the web application can only be deployed on iOS devices (as a bookmark), it has a lower degree of satisfaction related to portability.

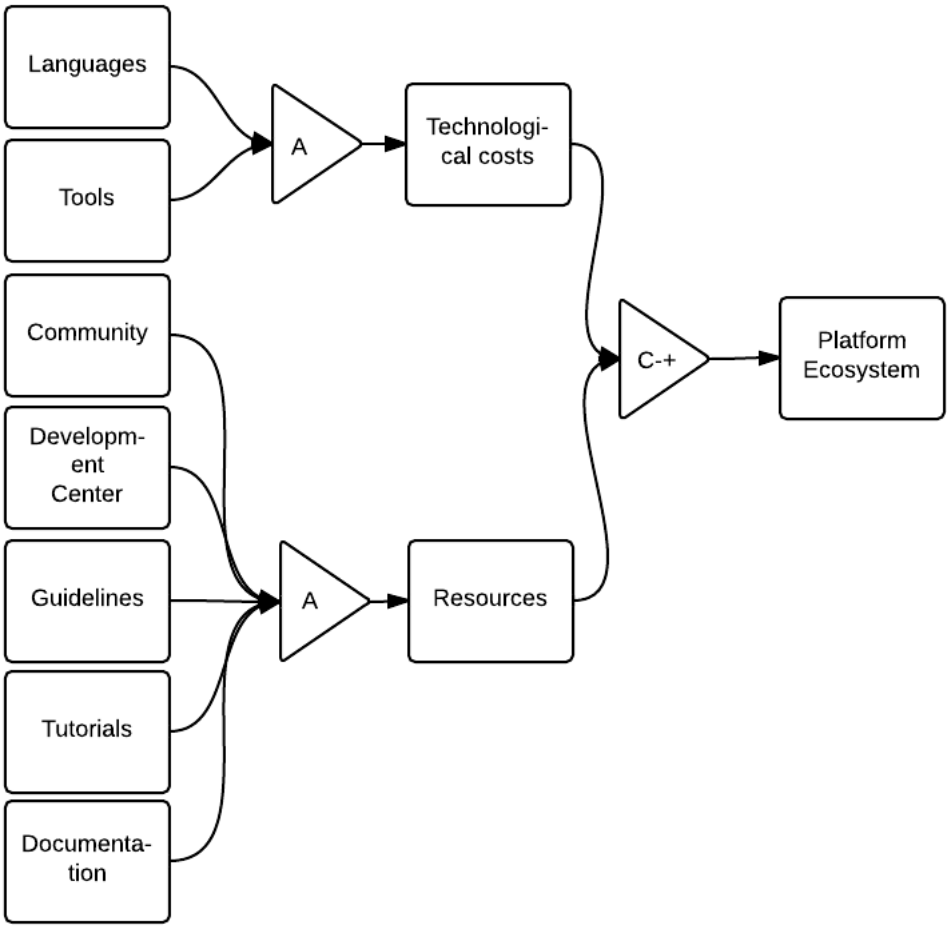


Figure 8.31: Partial logic aggregation for the platform ecosystem

Figure 8.31 shows the aggregation structure of the platform ecosystem characteristic. The partial aggregation of preferences show that for the platform ecosystem, the degree og satisfaction is fairly low.

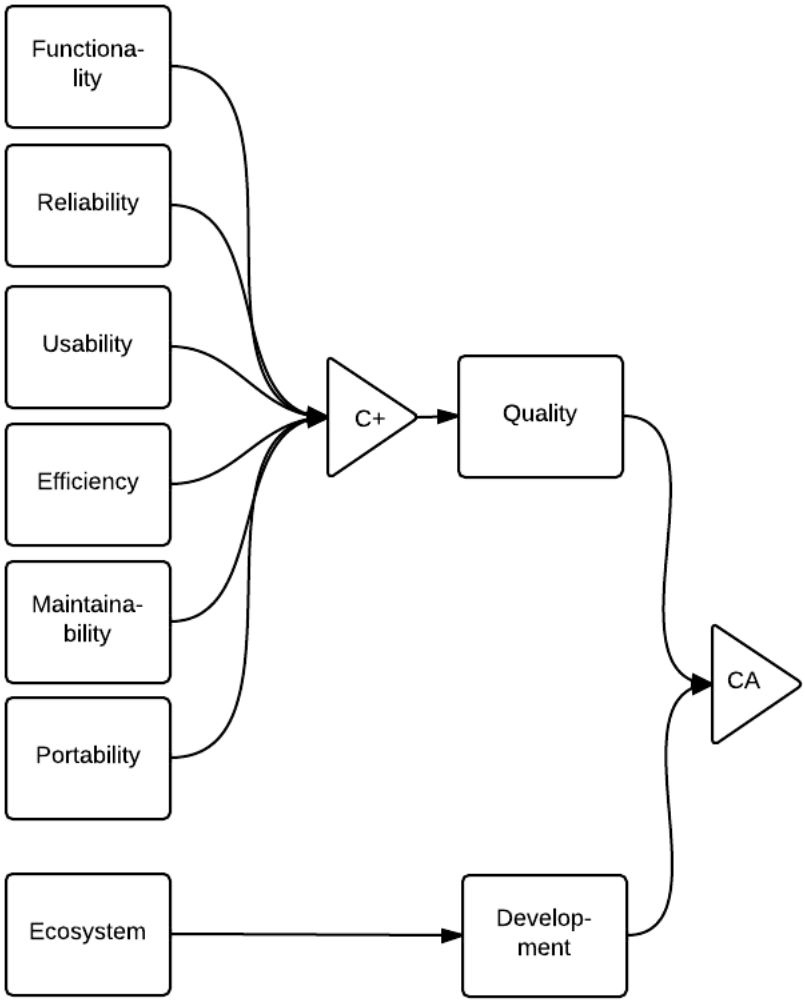


Figure 8.32: Global quality aggregation

figure 8.32 shows the aggregation structure of the total aggregation. The resulting value that is returned from this structure represents the global degree of satisfaction. In the case of this evaluation, this satisfaction represents the degree of quality related to the development of a mobile news application.

The quality characteristic of the system can be defined as *the quality of the resulting system that is achieved by using a particular framework*, while the development characteristic can be defined as *the amount of possibilities and limitations provided by the platform ecosystem.*

8.3.1 Summary

Table tab:score gives the partial and global preferences for for each of the mobile approaches. This is the last step of the aggregation, and it indicates that the mobile web applications has reached 65.3% of the quality preference, while the hybrid application reached 94%.

Quality Factor	**Weight**	**Web Application**	**Hybrid application**
Functionality	0.3	0.9247	0.9674
Reliability	0.1	0.9152	1
Usability	0.2	0.5847	0.5847
Efficiency	0.1	0.5544	0.5544
Maintainability	0.1	0.7259	0.5689
Portability	0.2	0.437	1
Quality		**0.6537**	**0.9403**

Table 8.1: Quality preference

Table 8.1 shows that the aggregated quality characteristics of the hybrid approach are much better than for the HTML5 application. The biggest differences between the two can be found in the portability of the applications.

Quality Factor	**Weight**	**Web Application**	**Hybrid application**
Development	1	0.6995	0.5968

Table 8.2: Development preference

Table 8.2 shows the development preference which is directly derived from the platform ecosystem preference. The preference shows that the degree of satisfaction related to the platform ecosystem is slightly larger when developing HTML5 applications.

Table 8.3 shows the final aggregated preferences. The final result is that an HTML5 application satisfied the evaluation criteria by a degree og 67.10%, while the hybrid application satisfies the evaluation criteria by a degree of 75.15%.

Quality Factor	**Weight**	**Web Application**	**Hybrid application**
Quality	0.6	0.6537	0.9403
Development	0.4	0.6995	0.5968
Global		**0.6710**	**0.7515**

Table 8.3: Global preference

Even though the hybrid application had a much better quality preference than the HTML5 application, the end result was more balanced because of the development preference, where the hybrid application scored worse than the HTML5 application.

9 Discussion

This chapter will discuss the creation of the HTML5 and hybrid implementations of the mobile news application, and experiences related to the chosen methodology that was used during the development of the applications. The chapter will also discuss issues and challenges that occurred during the development process, and how these were handled.

The chapter will first present a discussion on the methodology in section 9.1. Section 9.2 will describe the biggest challenges that were met during the thesis. Section 9.3 will discuss the results that were obtained after evaluating both HTML5 and a hybrid approach for mobile application development.

9.1 Methodology

The methodology chosen for the development process was an agile methodology. This made it possible to focus on implementing the functional requirements, and implement each of the functions without any big time delays in extra implementation specifics. By focusing on implementing one function at the time, the application always had some working functionality, unlike when all functionality is developed at once, and none of the requirements are working properly.

Each of the functional requirements was implemented in steps where they were tested right away. The testing was done after each of the implemented requirements instead of in the end of the entire development process to minimize the risk of ending up to change the whole implemented solution just because one part did not work properly.

The testing and evaluation after each phase was an self-activity, with little

input from other sources. This made the testing and evaluation somewhat harder, since the acceptance test were based on own personal opinions and do not necessarily reflect the opinions of others. If there had been a more thoroughly testing process with other persons that tried and used the application, the feedback would have been more accurate.

9.2 Development Challenges

Through the implementation phase, some unforeseen challenges were discovered. This section describes some of the most time consuming and important challenges from the development process.

One of the challenges that arose early in the development process was the *Same origin policy* that exists in all browser. This made it impossible to access the news recommendation system since the application was running on another host and port than the server, and all the AJAX calls from the application failed. The possibilities for fixing this were:

- **Proxy server** - Set up a proxy server and send REST calls to it, which then again calls the API with data, and the data returns the same way.
- **CORS** - CORS can be set up on the server that makes it possible to send the XMLHttpRequest across domains. This may present a security risk since one bypasses the same-origin policy.
- **JSONP** - One can use JSONP to pack a request in script tags, these must also be read on the server. This is a little bit scarier since it makes it possible to run any JavaScript code through an HTTP call.
- **Same domain** - The application can be hosted on the same domain as the server that provides the data.

The solution chosen for the same origin problem was to set up a Node.js proxy that routes all AJAX requests to the news recommendation system.

Another challenge that occurred was that JavaScript parsed data which contained hyphen on the server as operators, instead of variable names,

this was solved with locally set and rename all the variable names with hyphen to underscore instead. A good idea in later projects would be to avoid variable names that contains operators or other reserved names.

9.3 Evaluation

The evaluation results shows that both the web approach and the hybrid approach had satisfactory levels. The hybrid approach was ranked first with 75.15%, while the web approach got 67.10%. It is important to consider that parts of the evaluation was based on how well a native application would perform in the same situation, and in the case of the platform ecosystem, all evaluation criteria were derived from features of native platforms.

The characteristics that the applications scored lowest on were usability and efficiency. Of these two, the usability characteristic had the biggest impact on the final result. The Quality characteristic was computed with strong conjunction, which means that all the sub-characteristics were mandatory, but had an importance decided by weights. This means that especially for the mobile web application which had a low portability because it is not installable for most devices, the total degree of satisfaction drops considerably.

If we only consider what an application can do, HTML5 applications perform just as well as native applications in a news consumption case. The functionality and reliability characteristics had a degree of satisfaction above 90% for both implementations.

The biggest differences was in maintainability and portability. The portability in web applications scored lower than hybrid, and the hybrid scored lower in maintainability.

When considering these results, it is important to notice the degree of subjectivity in the evaluation. All the element criteria from the measurable characteristics have a big part in the determination of satisfaction for

a single characteristic, and the weights that are used are also subjective and impact all stages of the evaluation. In some cases the weights come naturally, as in the case of location that is critical for this kind of application, but it is much harder to determine what characteristics that are less important than others.

To some degree, the evaluation framework used to evaluate the non-functional requirements contained different levels of ambiguity that can be hard to handle, and make it difficult to define clear evaluation criteria. To be able to use this kind of evaluation method better, it is important to define criteria that are clearly defined and related to mobile applications and not software in general.

10 Conclusion

The goal of this project was to evaluate the use of HTML5 as development tool for mobile news recommendation systems, together with tools and methodologies used in HTML5 development.

During the project a mobile news application was created in both web and hybrid approaches. In the process of evaluation these applications, both approaches gave satisfactory scores, and the conclusion is that HTML5 is suitable for this type of application. The conclusion could have been different if it was an other type of application that was made, i.e. games and other input heavy applications.

The conclusion will also answer the research questions stated in section 1.1.

> *What are the advantages and limitations of using HTML5 as compared to native development on mobile platforms?*

The advantages and limitations of using HTML5 as compared to native development on mobile platforms were found during the technical research phase. Some of the most important factors are given in table 10.1.

HTML5	**Native**
Cross-platform	platform independent
Few development languages	Development languages for every platform
Optional IDE	IDE Constraint
Few APIs	Many APIs
No download needed	Download
Newest version accessible	Manually update
Less responsive	More responsive
Hard to find	App stores accessibility
Open browser and navigate to URL needed	One-click access
Few UI guidelines	UI guidelines
Limited access to native features	Full access to native features

Table 10.1: Advantages and limitations of using HTML5 contra native development

Table 10.1 shows that there are both advantages and limitations with the HTML5 approach compared to the native approach. And there is not one approach that differs greatly from the other.

> *What is the quality of current tools for HTML5 cross-platform service development?*

There exist a large amount of different tools that can be used in developing HTML5 cross-platform applications. Some of the most popular are taken into account in the development process, and a choice was made about who to use in the development of the mobile news application.

The big amount of tools available can make it hard to choose which to use, and they can be used for different things. It is important that developers perform a thorough research before they choose tools, and find tools that are most suitable for their implementation.

Since the mobile news application was a small client-side application, the development tools choice fell on write much of the code self. In a bigger project, it may be smarter to choose a bigger tool like jQuery Mobile.

What methodological approaches are suitable for HTML5 app development?.

In this thesis, an agile methodology was chosen. There does not exist that much information about methodological approaches in mobile development, but for larger enterprise solutions, a methodology that takes the mobile paradigm into account is important. Especially when considering the user experience of the application, and how this works across multiple devices.

For this thesis, the chosen methodology was a suitable HTML5 application development process.

11 Further Work

In this thesis, there has been performed a literature review and a prototype that demonstrates the use of HTML5 for mobile news recommendation systems. The mobile news application works for the given use case, but some functionality and further improvements had to wait because of the time limit. These improvements could increase the application usefulness:

- **Implement missing functionality** - Some of the functionality that were describes as requirements in this theses were not implemented. These are basic functions that the application should have.
- **User Profile** - Users could have their own user profile and their settings saved, so that they don't need to change settings every time they access the application.
- **Recommendations** - Recommendation based on related news and read ratio by others. This function could be added so that the users could get more recommendation based news.
- **More platforms** - The hybrid implementation could cover more platforms (iOS and Windows Phone). The hybrid approach is only implement for the Android platform, this could be extended, so that more platforms could be reached.
- **Search** - Implement Search functionality. A search functionality would give the user the possibility to search for specific news, and get matching news articles in return.

Another interesting possibility is to arrange focus group who performs in depth testing on the quality of the application. This would result in deeper understanding of the system, and making it easier to see which functional and non-functional requirements that is of importance.

Bibliography

[1] PhoneGap. Apps created with phonegap. http://phonegap.com/app. Accessed: 2013-05-29.

[2] Anthony Wasserman I. Software engineering issues for mobile application development. pages 397–400, 2010.

[3] Lionbridge. Mobile web apps vs. mobile native apps: How to make the right choice?, 2012.

[4] Jeff Wisniewski. Mobile that works for your library. *Online*, 35(1):54–57, 2011.

[5] Android developers. http://developer.android.com/. Accessed: 2013-05-29.

[6] ios developers. https://developer.apple.com. Accessed: 2013-05-29.

[7] Windows phone dev center. http://dev.windowsphone.com/. Accessed: 2013-05-29.

[8] Gustavo Hartman, Geoff Stead, and Asi DeGani. Cross-platform mobile development, March 2011.

[9] The Apache Software Foundation. Apache cordova. http://cordova.apache.org/. Accessed: 2013-05-20.

[10] Andre Charland and Brian Leroux. Mobile application development: web vs. native. *Commun. ACM*, 54(5):49–53, May 2011.

[11] mozilla. Firefox marketplace. https://marketplace.firefox.com/. Accessed: 2013-05-29.

[12] Matthew B. Hoy. Html5: A new standard for the web. *Medical Reference Services Quarterly*, 30(1):50–55, 2011. PMID: 21271452.

[13] Mark Doernhoefer. Surfing the net for software engineering notes. *ACM SIGSOFT Software Engineering Notes*, 31(4):16–24, 2006.

[14] Håkon Wium Lie and Bert Bos. Cascading style sheets, level 1. April 2008.

[15] Håkon Wium Lie and Janne Saarela. Multipurpose web publishing, using html, xml, and css. *Communications of the ACM*, 42(10):95–101, October 1999.

[16] W3C. Cascading style sheets (css) snapshot 2010. http://www.w3.org/TR/CSS/. Accessed: 2013-05-29.

[17] Tony Wasserman. Software engineering issues for mobile application development. *FoSER 2010*, 2010.

[18] Peter Lubbers, Brian Albers, and Frank Salim. The future of html5. In *Pro HTML5 Programming*, pages 313–321. Springer, 2011.

[19] Karen Henricksen, Jadwiga Indulska, and Andry Rakotonirainy. Modeling context information in pervasive computing systems. In *Pervasive Computing*, pages 167–180. Springer, 2002.

[20] Karen Henricksen and Jadwiga Indulska. Developing context-aware pervasive computing applications: Models and approach. *Pervasive and Mobile Computing*, 2(1):37 – 64, 2006.

[21] H.J. Lee and Sung Joo Park. Moners: A news recommender for the mobile web. *Expert System with Applications*, 32:143–150, 2007.

[22] Ntnu smartmedia. http://smartmedia.idi.ntnu.no/. Accessed: 2013-05-16.

[23] Mozhgan Tavakolifard, Jon Atle Gulla, Kevin C. Almeroth, Jon Espen Ingvaldsen, Gaute Nygreen, and Erik Berg. Tailored news in the palm of your hand: A multi-perspective transparent approach to news recommendation. 2013.

[24] Apache solr. http://vm-6120.idi.ntnu.no:8080/solr. Accessed: 2013-05-31.

[25] The Eclipse Foundation. Eclipse. http://www.eclipse.org/. Accessed: 2013-05-31.

[26] W3C. Same origin policy. http://www.w3.org/Security/wiki/Same_Origin_Policy. Accessed: 2013-05-27.

[27] W3C. Xmlhttprequest. http://www.w3.org/TR/XMLHttpRequest/. Accessed: 2013-05-20.

[28] Inc Joyent. node.js. http://nodejs.org/. Accessed: 2013-05-20.

[29] Appspresso. http://appspresso.com/. Accessed: 2013-05-27.

[30] Application Craft Ltd. Applicaton craft. http://www.applicationcraft.com/. Accessed: 2013-05-16.

[31] The jQuery Foundation. http://jquerymobile.com/. Accessed: 2013-05-27.

[32] Sencha Inc. Sencha touch. http://www.sencha.com/products/touch. Accessed: 2013-05-27.

[33] Martin Fowler. *UML distilled*. Addison-Wesley Professional, 2004.

[34] P Botella, X Burgués, JP Carvallo, X Franch, G Grau, J Marco, and C Quer. Iso/iec 9126 in practice: what do we need to know?

[35] The International Organization for Standardization (ISO). Iso standard 9126: Software engineering product quality, part 1: Quality model, 2001.

[36] The webkit open source project. http://www.webkit.org/. Accessed: 2013-05-31.

[37] Opera Software Asa. Opera mobile emulator. http://www.opera.com/no/developer/mobile-emulator. Accessed: 2013-03-20.

[38] DocumentCloud Inc. Jeremy Ashkenas. Backbone.js. http://backbonejs.org/. Accessed: 2013-05-02.

[39] Tilde Inc. Ember. http://emberjs.com/. Accessed: 2013-05-27.

[40] Knockoutjs.com. Knockout. http://knockoutjs.com/. Accessed: 2013-05-27.

[41] Google. Angularjs. http://angularjs.org/. Accessed: 2013-05-27.

[42] Derick Bailey. backbone.marionette. https://github.com/marionettejs/backbone.marionette. Accessed: 2013-05-02.

[43] Amazium. http://www.amazium.co.uk/. Accessed: 2013-05-02.

[44] Jozo J. Dujmović and Hajime Nagashima. {LSP} method and its use for evaluation of java {IDEs}. *International Journal of Approximate Reasoning*, 41(1):3 – 22, 2006.

[45] W3C. Geolocation api specification. http://dev.w3.org/geo/api/spec-source.html. Accessed: 2013-05-29.

A Use Cases

A.1 Opening a List of News Articles

Identifier	UC1
Description	A user opens the web application, and a list of news articles are shown.
Actors	User (primary)
Assumptions	The application must be accessible through an URL or cross-platform application.
Steps	1. The user opens the application. 1.1. The user opens the application by typing in an URL. 1.2. The user opens the application by selection the application. 2. A list of news articles are shown to the user.
Variations	None
Non-functional	• **Performance** - The application should have a maximum delay. • **Robustness** - The application should fail gracefully if there is no connectivity.

A.2 Opening a Single News Article

Identifier	UC2
Description	A user opens a news article from the list of articles.
Actors	User (primary)
Assumptions	There exists a list of news articles available to the user.
Steps	1. The user navigates the list of available news articles. 2. The user selects a news article. 3. The news article is opened.
Variations	None
Non-functional	• **Performance** - The time to fetch the news article should happen without delay. • **Robustness** - Failures should be dealt with gracefully. • **Quality** - The news article should be easy to select.

A.3 Opening a Single News Article in a Map

Identifier	UC3
Description	A user opens a news article in a map.
Actors	User (primary)
Assumptions	The user has opened a news article from the list of articles.
Steps	1. The user opens the map for the desired news article. 2. A map with the news article is shown to the user.
Variations	None
Non-functional	• **Performance** - The application should response with low response time. • **Robustness** - Failures should be handled correct by the application and give the user proper feedback. • **Quality** - If the article does not have any location, it should not be possible to choose it. The map option should be intuitive.

A.4 Opening News Articles in a Map

Identifier	UC4
Description	A user opens the map with news articles.
Actors	User (primary), Application
Assumptions	The application has preloaded all news articles. The application has access to the users location. In the settings menu, the map option is chosen.
Steps	1. The user selects the map option. 2. The application checks which articles to show in the map. 2.1. The application retrieves the show all news setting. 2.2. The application retrieves the local news setting. 3. A map with markers with the location of news articles are shown to the user.
Variations	None
Non-functional	• **Performance** - The application should have a small response time. • **Robustness** - The application should handle failures in a proper way. • **Quality** - The map should be easy accessible.

A.5 Filter News Articles

Identifier	UC5
Description	A user opens the settings menu, and selects a settings option.
Actors	User (primary)
Assumptions	The application has preloaded all news articles. The user is currently interacting with the view of all news articles.
Steps	1. The user opens the settings menu. 2. The user selects a option. 3. The application sets the option. 4. The user saves the settings. 5. The news articles are reloaded.
Variations	None
Non-functional	• **Performance** - The application should respond fast when the user change the settings. • **Robustness** - It should not be possible for the user to choose settings that is not possible. • **Quality** - The settings option should be easy to recognize.

B Element Preference Calculations

B.1 Functionality

Characteristic	Weight	Score web	Score hybrid
Suitability - 0.6			
Storage	0.1	1	1
Connectivity	0.1	2/3	1
Geolocation - 0.6			
GPS	0.6	1	1
Network	0.4	1	1
User interface - 0.2			
Back button	0.4	2/3	2/3
Resolution handling	0.5	1	1
Orientation handling	0.1	1	1
Interoperability - 0.3			
Data formats	0.3	1	1
Communication protocols	0.7	1	1
Security - 0.1			
Privacy	1	1	1

B.2 Reliability

Characteristic	Weight	Score web	Score hybrid
Maturity - 0.5			
Offline capability	1	5/6	1
Recoverability - 0.5			
Storage	1	1	1

B.3 Usability

Characteristic	Weight	Score web	Score hybrid
Understandability			
Navigation	0.4	5/6	5/6
User interface	0.6	3/6	3/6

B.4 Efficiency

Characteristic	Weight	Score web	Score hybrid
Time behaviour - 0.5			
Latency	0.5	0.4	0.4
Response time - 0.5			
Time consuming operation	0.4	0.40	0.40
Normal operation	0.3	1	1
Easy operation	0.3	1	1
Resource utilization - 0.5			
CPU	0.5	0.98	0.98
Memory	0.5	0.3	0.3

B.5 Maintainability

Characteristic	Weight	Score web	Score hybrid
Analyzability - 0.4			
Diagnostic tools	1	0.5	0.5
Changeability - 0.2			
Updates within reasonable time	1	1	0
Testability - 0.4			
Test libraries	0.5	1	1
Test runners	0.5	1	1

B.6 Portability

Characteristic	Weight	Score web	Score hybrid
Adaptability - 0.7			
Effort	1	0.5	1
Installability - 0.3			
Number of platforms	1	1/3	1

B.7 Platform Ecosystem

Characteristic	Weight	Score web	Score hybrid
Technical costs - 0.7			
Languages	0.4	0.5	0.5
Tools	0.6	1	0.7
Resources - 0.3			
Community	0.3	1	3/5
Development Center	0.1	1	1
Guidelines	0.2	1/3	1/3
Tutorials	0.3	0	0.5
Documentation	0.1	0.5	0.5