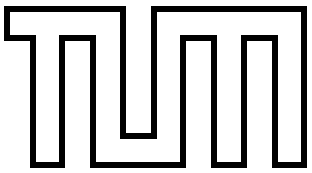


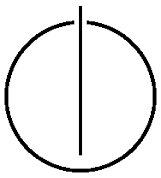
FAKULTÄT FÜR INFORMATIK

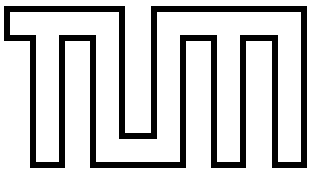
DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Masterarbeit in Informatik

Konzeptualisierung und Architektur einer Content-Management-App für HMI apps im Kontext eines E-Cars mit zentralisierter IKT Infrastruktur

Josep Mateu Clemente





FAKULTÄT FÜR INFORMATIK

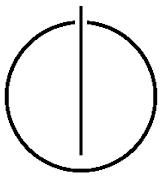
DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Masterarbeit in Informatik

Konzeptualisierung und Architektur einer Content-Management-App für HMI apps im Kontext eines E-Cars mit zentralisierter IKT Infrastruktur

Conceptualization and Architecture of a Content Management App for HMI apps in the Context of an Electric Car with a centralized ICT infrastructure

Bearbeiter:	Josep Mateu Clemente
Aufgabensteller:	Prof. Dr. Dr. h.c. Manfred Broy
Betreuer:	Thomas Koflert
Abgabedatum:	April 15, 2014



Ich versichere, dass ich diese Masterarbeit selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel verwendet habe.

I confirm that this master's thesis is my own work and I have documented all sources and material used.

München, den April 15, 2014 Josep Mateu Clemente

Acknowledgements

I would like to express my gratitude to my friends and family that stood by my side on the works of this thesis, their permanent presence and support has made possible for me to finish it. I could not have envisaged a better travel team. A special thanks go to my friend Andrew, who was all the time laughing about the close relationship I and this thesis have had and to my girlfriend Aleksandra, who has made my world change in an unbelievable way.

In addition, I would like to thank my thesis tutor who has stood firmly believing in this project even on the tough stages, even the difficulties and bad moments that arose.

Lastly I would like to thank everyone from my home University (Gerard, Eduard thank you!) and all those people that I have met on this wonderful city and that have left a footprint in me in one way or another. Furthermore, I could say that this thesis has brought me the possibility to get to know one of the most impressive cities I have ever been to, to know wonderful people and to build myself as a people. I could not foresee a better scenario to experiment the change I had while being here and studying with this University.

“Education is the key to unlock the golden door of freedom.“ (George Washington Carver)

Abstract

We understand the concept of Human Machine Interfaces for Automotive Machines from the Software Engineering perspective as a family of software designed to provide a Graphical User Interface on-board a vehicle as well as acting as a bridge between the User and the Machine. The aim of this thesis is to contribute to the Infotainment world on the concept of life expectancy, which normally dies within the few years of release. This fact is caused by two factors: lack of updates and improvements and inability to add new features to the product, which the HMI as an element nearly without consideration on the choice of an Automotive Device. The solution we present in this thesis, in terms of software, offers the possibility to update itself and extend its functionalities after the deployment has been done, also opening the door for external developers to contribute to these enhancements. There are multiple solutions already on the market regarding HMI software all of them proposing different alternatives and models. These solutions have points in common and defects that have had to be identified and merged in order to be able to create a competitive solution. We propose a solution using web-based technologies according to the HTML5 standard which allows for a Operating System-like interaction. This interaction is based on a layered concept that encapsulates each layer and allows for security controls from other components inside the Webpage. Our solution also has the ability to be deployed as a native application to a mobile device in order to be able to access to extra features not defined by the HTML5 standard. We couple this solution with a Content Management System on the Cloud which allows for Components and Updates to be installed from it. The use of Web Technologies is motivated because of them being widely spread and their independence from any underlying environment, therefore being able to be run on any device that supports HTML5 browsing. As this thesis is an architecture proposal we have not been able to provide any feasible evidence regarding its fare against the competitors. We have been able to see, though, that the current and future developments on this world are trending towards a solution similar to the one we have provided, fact that makes us believe on the future of our proposal.

Contents

List of Figures

1 Introduction

In this chapter the first approach to the problem presented will be explained, as well analysing the goals and objectives that need to be resolved in this work. The first ideas on which the rest of this thesis is going to be based on are also going to be presented here.

1.1 Context

We understand this thesis in the context of the so-called RACE project, whose objective is to create an centralised ICT architecture. This means, the project main goal is to separate driving, driver assistance and infotainment features from the control unit hardware and install them through software [Pro13]. This thesis, therefore, would be able to comply with the role of an infotainment system without implying itself in a greater scope in the RACE project. This means, that the thesis must be able to provide a solution that is compatible - but not limited to - the RACE project. This solution would therefore comply the role of the centralised HMI system by the use of a tablet display device, that would only be used to display the software, but would be deployed in a hardware environment that could coexist with other RACE systems. This thesis will also abide by the structure and Application Programming Interface scenarios envisaged by the RACE Project, therefore complying with them and their limitations.

1.2 Stakeholders

In this section we will be analysing the stakeholders for our thesis. "A stakeholder is normally defined by an entity (group or individual) who can affect or is affected by the achievement of a thesis objectives. [SFG99]". For this thesis, we, as a stakeholder, consider any entity that we want to target as having a relationship with our solution. We will be identifying the multiple stakeholders according to the stakeholder identification pattern described by Sharp, Finkelstein & Galal [SFG99].

1.2.1 Manufacturer

We define a Manufacturer as an entity that is able to provide an automotive system without HMI where out system can be deployed. Our definition of Manufacturer can be understood under the umbrella of "Developer", due the fact that they will be covering multiple roles (e.g., Quality Assurance, Product Training, Associated Software development, etc.).

1.2.2 Main User

We define the wording "Main User" the entity that will use the result of the thesis once developed. To simplify our scenarios, we will consider that this term is applicable only to the companies or people that perform a direct purchase on the developed solution. We consider that these people need to be the common users of such systems and we don't contemplate any sublet scenario.

1.2.3 Other Users

We define as "Other Users" the entities that will utilise the result of the thesis once developed, but on a passive or secondary basis. As with the previous stakeholder, we will simplify the scenarios by specifying that such entities will be the companions of the "Main User" stakeholder during the usage of the developed solution. We consider these people as occasional users of these systems.

1.2.4 Project Owners

We deem as "Project Owners" as the entity responsible for this thesis and any future associated work. They have the power to heavily influence the direction of the thesis, and they could be reduced to the writer of the thesis and, the advisor.

1.2.5 Competent Authorities

We define "Competent Authorities" as institutional entities whose main aim is to ensure that any HMI for automotive environments comply with certain regulations. We will deem this entity as fictitious due to them being different for each country, and only envisage certain general requirements. This stakeholder itself is covered under the umbrella of the "legislator" according to the pattern we're following.

1.3 Problems and Motivation

An HMI system stands for the human-machine interface in any kind of machine that allows a user to interact with an electronic or mechanic component. Typically on vehicles this system has been analogue, but with the advancement of technologies multiple proprietary systems that provide a digital user interface have arisen, therefore opening this world to extensive competition between the different companies. While trying to develop a next generation car like the project RACE, context of this thesis, we find multiple issues on this aspect. Our main intention is therefore to make a contribution by providing an alternative solution oriented on solving these issues.

1.3.1 A uniform system

Currently some systems are developed with support of only a small set of partner companies (e.g., BMW will only support Google Services [RBA13]). This forces the buyer to either own a supported device or give up specific features, thereby reducing the final real value of the product. Solutions to this problem have been proposed, some not being able to solve it due to it not being part of the expertise of the manufacturer company (e.g., Ford uses Windows Mobile [Com13a]). At this point, a number of companies tried to solve this problem, the most successful being Nokia with their Terminal Mode proposal [BBP10], but ended up with the same problem as exposed. ur solution will be considering these facts and aims to be an alternative without specific Operating Systems or contextual boundaries, ideally attracting further developers by embracing some open source ideas.

1.3.2 Multiple research branches, multiple companies

On the line on the previous section, we have observed the fact that each developed system has different user experiences, and as it can be seen on the Analysis section each of them provide different user experiences. There is also a lack of concrete standardisation (although efforts are under way, e.g. [HGM+12]) on topics regarding HMI, and therefore a different array of technologies is used on them often rendering them incompatible with other systems (though we find proposals that aim to solve this problem [IL13]). On these ground of these previous topics our solution will propose a uniform user experience based on the research and alternatives already mentioned.

1.3.3 After-selling business model

The current after selling model resides purely on customer support on components, therefore highly tampering the capability of revenues over time that a device as a car, used by most more than once per week, could provide. The previous affirmation is taken from the fact that application selling has became a huge part of revenues on the mobile world [McC13], therefore rendering us with the ability to predict the possibility that this would also happen if the context on a HMI was equivalent. From application development itself, it has proven that Indie techniques are the one marking the biggest trend by far due to the fact of it being supported on advertisement and in application purchases [Val11].

1.3.4 Open standards

To the date being, most proposed infotainment systems have been of closed development (with open source initiatives arising recently [Gen13]), due to the fact that infotainment systems are an integral part of the automotive device, and therefore reflect the Company Image (with them two being bounded, as examples have proofed [Rub13]). Some car manufacturers have chosen, though, to share the design and infrastructure between their sub-branches with successful results (e.g., General Motors [Mot13]), but majority they have different designs and usability standards between them (e.g, [Aut13]). Resuming the exposed concerns, one of the problems that we face is the idea of allowing for a solution that could be used by any business alike, but would alike customization in the very much way mobile devices are driven on some open source environments (e.g., [Fav13]).

1.4 Idea of Solution

Based on the previous section an idea of a solution can be obtained. We have identified the main problems we think an architecture for this scope should be solving, thus we consider that the subsections exposed before are part of the most considerable troubles any solution for this problem could be facing. We will be analysing the scope of the requirements at a later section. Our thesis should concentrate into finding the common points of the solutions already available on the market and group their best strengths together in order to propose an alternative, which should be able to be transformed into a prototype, to provide evidence for one of the possible paths the HMI development is going to take place into. From these points, we can decide that the thesis is going to be oriented towards designing a solution complying with open-source standards while offering the possibility to customize it (as seen in the requirements). We will also give our vision into the scope of standardisation of the HMI systems and build upon already done research (e.g., on the analysis chapter) to provide for unifying solution. Adding to this point, we also need to consider (as part of the initial problem of this thesis) the fact that the solution should be oriented towards offering Content Management facilities and being able to cope with new functionalities as they are necessary, scope that also will be tackled in the proposed solution.

1.5 Goal of the Thesis

From both points stated before a more generic idea of the thesis can be abstracted which in turn could be resumed in a few points as stated next. The main goal of this thesis is the analysis of the current state of the art and the proposal of a new architecture that aims to sum ideas from the existing ones while adding concepts for Content Management. From the analysis and requirement sections, we have concluded that the following statements require a special consideration.

1.5.1 Compliance with standards

As stated previously there is a lack of completely defined standards (again, though, there are efforts under way e.g., [HGM+12]), but some first-world regions have defined some measures that such systems should have (even controversial ones e.g., [Jay13]). For this aspect, we will be considering these and other usability measures that have already been defined as part of ongoing research (e.g., [RMC13]) and that we think this thesis should take into account. Another important aspect regarding this topic are the different kind of interactions and the research done into it (e.g., [RFB+13]), which need also to be taken into special attention as requirements and indications the solution should present. On other terms, one of the goals of this thesis is to analyse the current state of the art (seen on the analysis chapter) and substantial research that has been done and provide a solution that takes this facts into account as a part of the main proposal that represents.

1.5.2 Open source environment

As exposed beforehand in the problems section, as analysed with posterity in the analysis sector and, coping with the initial problem of offering a Content Management System (CMS) on an HMI solution, we have decided that one of the goals on this thesis is to offer a solution that is able to be provided on an open-source manner, thus it becoming a requirement of the solution. As it can be seen on a later stage, we consider the concepts CMS and Open-source coupled on our solution (check the solution chapter for further clarification), and therefore we established as a goal the fact that the solution is able to be distributed on an open source manner and that is able to expand it's designed capabilities on a later stage. There have been analysis supporting both closed and open source design (e.g., [bwi09]), but empathising the connection between innovation and open source. This connection is the fact that made us choose open source (as discussed further ahead), due to the possibility to provide with substantially more content. The advantages provided by open sourcing are also clear, in the sense that any product or security flaws can be easily found by a large community of contributors (e.g., [Bou05]) and therefore easing the duration of an average bug report.

1.5.3 Cross platform support

The main aim of the product is to be compatible regardless of the hardware components, and to a greater extent be able to operate for all the users on the car regardless of specific devices being installed or not. For this objective a clearly defined and ready to use platform is needed. We have to assure that the software being develop is able to cope with a planned lifespan of a vehicle (e.g., up to 5-10 years [Inc13f]) or even with the average use on well treated vehicles (e.g., 11 years [Rec13]), in a sense we can keep a place for compatibility in the future for every new operating system being released. Furthermore, by combining this the fact of it being free of charge, and open source we can reach a wide array of not only developers but designers. These last are set to greatly improve the Look and Feel of the user, which will make for a great step in our aim to universalize our product. Ideally, this cross platform support should be able to support applications being ran natively in case they are installed (such as map applications), mainly due to the extended functionality of native applications but without renouncing to universal use of the already defined features.

1.5.4 Generic

As previously stated, one of the main problems of current system is the necessity to comply the manufacturer's standards. It is imperative that we solve this problem so that any car manufacturer can provide an Application Programming Interface capable of interacting with the designed product. To this extent the solution should be compatible with any context, such as the project RACE (e.g., [Pro13]). Even though these projects are set to go along, the product should be developed with total independence of them, by using bridge technologies that would act as an abstraction layer for them (e.g., [HG13]). Furthermore, defining a standard on car features should be avoided at all costs, as this is out of the scope of this thesis and is to be defined by the context the solution will be under with. Although this, the solution should be able to continue compliance with the other goals by resorting to independence from the environment (tackled in a further section).

1.6 Outline

In this section we have defined the main problems that have led to the creation of this thesis, as well as tried to scope the thesis to the problems and goals defined in this chapter. The main purpose of this thesis is, therefore, to analyse the existing technologies and projects that could be of our interest, as well as observing which have been the main faults of the already existing ones that have led to the start of this thesis. We will be presenting the thesis on the form of Related Work-Analysis-Solution-Discussion, with these three chapters interconnected directly between them. This means, on the first mentioned chapter, Related Work, we will be exposing the research that we have taken into account for this thesis and the current state of the art, concepts that will be studied and break down into requirements and Use Cases on the chapter of Analysis. After this, we will be proposing a Solution that is able to cope completely or partially with the proposed requirements and Use Cases, and we will be discussing the outcome of this Solution on the Chapter labelled Discussion. Furthermore there are two extra chapters labelled "Background" and "Future Work" that indicate background concepts and ideas associated with the thesis and the work that could be done after the proposals of this thesis. As an added complement, we have also provided a prototype to provide evidence for the Solution, on the results of which the Discussion is also based.

2 Background

In this chapter we will be exposing concepts needed to understand the ideas behind this work. These concepts have been taken heavily into account when developing the solution due to it's relationship with the Web Technologies.

2.1 Web Services

In this section we will be going through a short review on the concept of"Web Service", altogether with the techniques that we will be using on our solution for these. Further information on the this topic can be found widely on multiple publications that have been published since the last decade. The following concepts are based on concepts of these publications (e.g., [EAGK03]).

2.1.1 Remote Procedure Call (RPC)

One of the most used methods nowadays, it allows the remote invocations to resemble the local ones, thus reducing the complexity of the code. The only problem realisable with such a technique is the error handling, due to them not being implicitly obvious by the client. We can find two variations of RPC's, the synchronous and the asynchronous (also called of the type"fire and forget"), where the client is also known by the server.

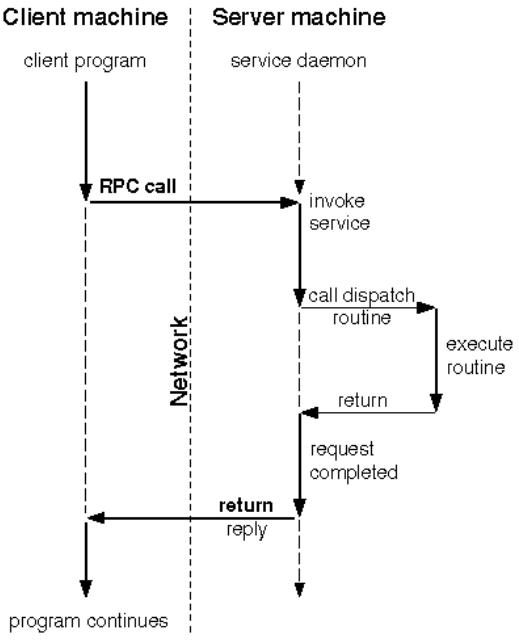


Figure 2.1: Network communication with the remote procedure call [TSG04]

2.1.2 Publish/Subscribe

The objective of this method is to separate the clients (subscribers) from the servers (publishers) through an intermediate layer that hides the participants. More said, it also allows the clients to be subscribed to the events that they desire and therefore to avoid others that otherwise would be discarded. This system, also, allows to act in a transaction basis and also independently of each participant's execution time.

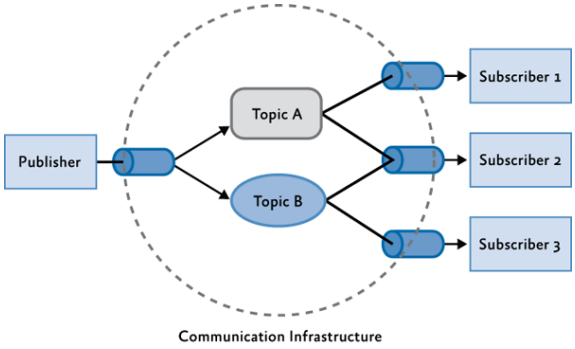


Figure 2.2: Publish/Subscribe Pattern [Mic13c]

2.2 Web Technologies

In this section we will be analysing the web technologies used through this thesis, altogether with their main functionality and the approach taken while considering them. The knowledge on this section also covers the sections relating to cloud computing, as well as providing a short insight on how the main relationship between client and server works.

2.2.1 HTML5, CSS and Javascript

HTML is the most common markup language for structuring and presenting content for the World Wide Web. Concretely the fifth version, still considered as a draft version [Con13], introduced many new features opening the door to hold a structure similar to that of an Operating System by providing the ability of using a Message Queue [EAGK03], and the ability of using such for multi-tasking [NH13]. Similarly we have seen the introduction of the corresponding standard on JavaScript, ECMA Specification 5 [Int11], that complements such scenario and allows us the usage of this specific technologies. Finally, the new standard allows us to bring the full world of 3D-graphics natively to the devices [Con11], which allows for an even greater support for application development on Web Environments.

2.2.2 PHP

A lightweight server-side web development language, it is one of the most widely used as Web Applications server-side programming language. It has gone through multiple revamps and restructures along its lifeline, but has become one of the iconic elements from the WWW deriving on the famous AMP (Apache, MySQL and, PHP) structure. The version that we consider is the 5.3 [Gro13].

2.2.3 Client-Server Web Relationship

One of the key concepts on Web interaction is the relationship between Client and Server, which normally is asynchronous and stateless [Soc99]. This concept is key to understand the architecture of any Web Environment, and should be considered through the whole thesis. Multiple solutions have been developed for this problem and progress towards a state web environment has been made, more on this will be discussed in the State of the Art analysis.

Cloud Systems

Cloud Computing is nowadays a considerable trend to take into account. By avoiding the need for a physical architecture or data-centre and outsourcing this to providers as Amazon or Google we can ensure the availability and scalability, up to a certain point, of a service. Further knowledge on this field is necessary for understanding any kind of web application that is hosted on the network [AFG+10].

2.3 Asynchronous Execution - Promise

As a key concept of asynchronous execution we find the concept of a Promise [LS88], that indicates what the user will return. It is worth noting that this concept has been long unused in the few decades in the world of code development, though its potential has increased since the advent of Web Technologies. The basis of this document revolves around this concept, and therefore has become one of the key pillars into understanding the works of this thesis.

2.4 Device Types

Though this concept is further developed on the thesis, it is mentioned often all through it and therefore is important to be able to identify and classificate the different kind of devices that might be present on a HMI architecture.

2.4.1 Main Device

We define as Main Device those devices that are installed or connected directly on the dashboard of the automotive device and therefore under the control of the pilot or the copilot. Those devices have the highest priority(or privileges, that is, are able to change directly settings of the car) on the scale of devices due to resembling to the traditional control board on the dashboard of an automotive device.

2.4.2 Trusted/Embedded Device

We define as Trusted Device those devices that are directly connected to the automotive device by the manufacturer, and therefore are surely part of it. These devices have therefore have a lesser degree to modify settings from the automotive device but are still treated as devices that should be able to access a considerable array of features from it.

2.4.3 Guest Device

We consider as guest device those devices that are connected to the system on a specific period of time, but are not bundled with it and therefore are aliens to the system. We consider that these devices should be able to manipulate a minimal amount of the automotive device settings and also access to a limited array of functionalities.

2.5 Plugin

We understand as for plugin on this thesis the concept of an application, be it standalone or not, that is encapsulated and prepared to communicate using only defined protocols. This application doesn't need to load any external module other than it's own, and while may need external communication to do it's proper function, it is designed so that it can be ran without this communication.

3 Related Work

In this chapter we will be analysing the products or components that are currently available or in process of becoming so. We will also document research that has been done and proposals that have not succeeded. The conclusions from this chapter will be the basis of the next Analysis chapter.

3.1 State of the Art

This section will provide an insight on the software currently available on the market that could be taken into account as references or possible incorporations for the proposal of a solution or prototype.

3.1.1 Native Phone Framework

In order to be able to access the full set of features from a device we require a framework that allows us to extend our application, thus bringing the generic web application into a native one. To that end, the frameworks listed in this subsection should support the conversion from the HTML5 and JavaScript to each deployment device's native system (on an initial basis, and as a non-negotiable requirement, Android).

Apache Cordova

This software is the base of others that will be explained in subsequent subsections. It allows to access native API's from a Web Application using JavaScript calls, thus removing the necessity to program directly using the native environment programming language [Fou13a]. This software does not provide a UI interface and thus one is recommended, such as jQuery Mobile or Dojo Mobile. This framework is licensed under Apache License 2.0 [Fou04] and thus could be used without paying any license fees nor modifying the distribution license.

PhoneGap

Since October 2011 licensed with Apache License 2.0 [Pho13], as its code is now part of the Apache Cordova Project, it is free of use and doesn't require license fees nor having to modify the distribution license. This Framework allows to deploy applications to a huger variety of Operating Systems than the other analysed frameworks, accessing its native SDK methods via JavaScript and packaging the application as Native bits (in a similar fashion to Java compiling e.g., [HGmWH96]), thus dramatically increasing the Applications speed due to the use of built-in JavaScript and HTML rendering engines. To build applications for the Apache Cordova Framework, PhoneGap should be used as it provides ready-to-deploy templates. The following subsections will analyse the available UI Frameworks that could be used.

Appcelerator Titanium Mobile Development Environment

This is one of the most used Mobile Frameworks, more than 30000 Applications have been shipped to Marketplaces using it, and provides with an extensive API set that allows users to manipulate the interface in a platform-independent way. Its flagship features consist of providing a Marketplace where developers can download other compatible APIs (such as the AT&T API [ATT13]) and the fact that supports desktop, mobile and tablet environment [App13a]. The main drawback consists on that the code is being interpreted by the program instead of being "compiled", that means that the code execution is slower and takes more resources than a native application. It is licensed on a free or a paid version, which differs on the services in the cloud. That is, it is included services pro month on the free version but in a limited scale, that may be expanded (including professional support) by purchasing a license.

Kurogo Mobile Platform

Coming from the MIT Mobile Web Framework [Com13b], this project aims to provide a rich to use mobile web feature set with a full set of abstracted features that allows users to even use Ruby to program for the Framework. This Framework is used by prestigious universities like the Harvard University or the Vermont University, only stating the main disadvantage that resides on the fact that to develop for Native Applications a license fee is required [Inc13g]. This is because the Platform is licensed under LGPL [Sys07], and for Apple Store applications this license is not applicable.

QT Framework

Following the release of Qt 5.1 Alpha, native Application Support for Android and iOS has been added. Qt is a renowned framework that has a wide variety of features emphasising on advanced OpenGL integration. The principal problem resides on the fact that the HTML and JavaScript code wouldn't be optimised and would run under the Qt's Web Browser or the Operating System's one [Oyj13]. The product is licensed under LGPL or under a commercial license.

MoSync

This product consists of two sub-products, MoSync and MoReload. The first one provides a hybrid combination of C++ and HTML5/JavaScript in order to develop the interface, while the second allows for live test of the applications on Mobile Devices [AB13]. This product is licensed under LGPL but also has an annual free license (which must be renewed, free-of-charge every year).

Corona

This framework uses house-build HTML and JavaScript engines and only provides a limited amount of features to access the native interfaces on its Free Edition. This Framework was released on 2008 and hasn't had a huge impact in the Native Application Frameworks world [Inc13c], mainly due to its constant license changes and to the fact that nowadays only supports 4 Mobile Operating Systems: iOS, Android, Kindle and nook [LLC13]. Corona offers for free a service called "Corona Cloud", which allows for free and unlimited User Account management, leader boards and achievements. That being said, all the other features are limited on its free package and should be upgraded to more expensive packages (multiplayer, push notifications and cloud sync between other options are also available).

Other Alternatives

There are other alternatives such as *RhoMobile* [MS13], which was set on an unknown status after the purchase by Motorola, or *Unity 3D* [Tec13b] which is a huge suite for 3D Game development, but it is not designed to work as a Native Phone Framework. A comprehensive, community maintained list of alternatives is available on the net [Wik13].

3.1.2 Web User Interface Mobile Frameworks

In this section we will be analysing the multiple Mobile Web User Interface Frameworks available on the market that have support for Touch Devices. These frameworks have also had a significant impact on the market [Ltd13].

JQuery Mobile

This framework is the Mobile library from the jQuery Framework (the most used according to cited statistics). It is integrated with the jQuery project thus offering also Desktop environment support. It also provides for an ease-of-use Skin creation and management and a module selection, to reduce the amount of JavaScript code by selecting the used features [jF13]. The only comparable UI Framework would be Dojo Mobile, but still jQuery has a broader coverage of browsers. It is licensed under the MIT License [Ini13], which basically allows for free use as long as the copyright header is held intact.

Dojo Mobile

A differential feature from this framework is the design, as it is designed as a loosely-based plugin environment that allows developers to reduce the amount of JavaScript code needed to be loaded into the web browser. The main advantage also is the huge coverage of browsers, including both Desktop and Mobile flavours and also adding native look and feel versions for mobile environments [Fou13c]. It should also be noted the huge number of important companies behind this framework (such as VMWare or IBM [Fou13b]), which provides it with a stable and long-term development team and support. This framework is licensed under either the modified BSD [otUoC13] or the Academic Free License [Ros05], which basically allows developers to use it free of charge and without any need to modify the final product license.

Sencha Touch

The free version of Sencha that allows to deploy for touch-enabled devices. It uses a custom built JavaScript UI interface which aims to provide native looks on each device, but for embedded or desktop applications deployment requires a license fee [Inc13h]. One huge advantage of Sencha is the ability to interact with the AT&T Platform API SDK, which should simplify the development in case this platform was required.

Google Web Toolkit

This full-fledged toolkit allows developers to use Java in order to develop for JavaScript, as it includes a compiler that automatically translates the Java code into usable JavaScript, thus augmenting the level of abstraction and reducing the time required to develop. This software includes also a Mobile version, with UI specifically developed to fulfil the mobile browsing requirements, as well as a wrapper around PhoneGap to provide for an easy-to-use interface that bridges between the page and the framework [Inc13e].

3.2 Communication Protocols

In this section we will analyse the current standardised or draft protocols available for communication between the web client and the server. The extra data and flaws of these protocols will be shown, as well as its ability to provide updates without the other part's initiation.

3.2.1 XMLHttpRequest and Comet Long Polling

The goal of this technique is to allow the server to "push" information to the client, thus the need for a real-time connection between the two. To do so, the client has to ask the server for information, because there's no full-duplex connection (and thus, the client can only receive data as a response from the server) [Con12].

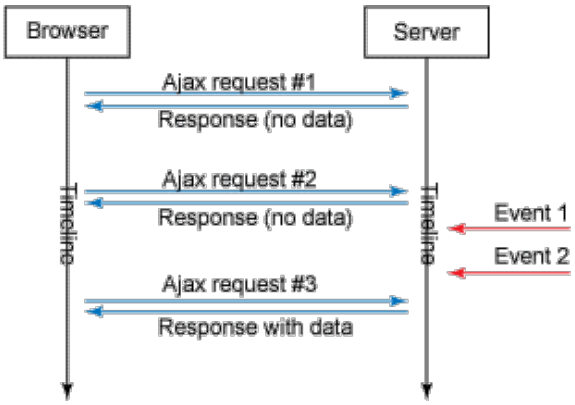


Figure 3.1: Reverse Ajax with HTTP polling [Car11]

The problem of this method resides on the amount off bandwidth lost, to overcome that disadvantage we may think of just sending the response when there's actually an answer. This fact, but, leads to the problem of when shall we decide to send the request? What if we need a constant flow of critical information? This is when Comet [Ido] comes in, as we can see on the following drawing.

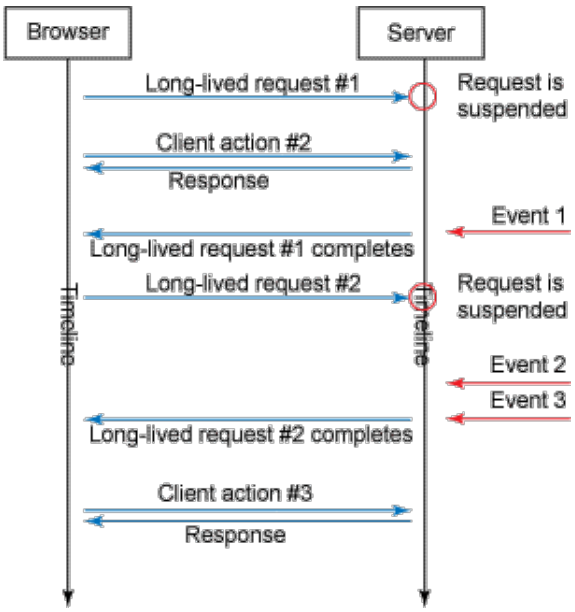


Figure 3.2: Reverse Ajax with Comet [Car11]

By just keeping the request open (much like streaming) we can end it when we do actually have the data, and initiate a request as soon as we receive the answer. This solution pretty much solves any underlying problems, only leaving the ones regarding the amount of data lost on doing requests and to the fact that we need a method to know if a connection has been lost, apart from the one of regularly polling the server, and that's precisely where WebSockets fit.

3.2.2 Web Sockets

Web Sockets is a novel technology meant to be the next big step forward in changing the actual HTML model [Kum12], in a sense that allows for constant communication between the browser and the server without resorting to huge payload techniques. An extended analysis is required because at the moment of writing this document [HG13] it is still considered as a DRAFT status by the W3C and thus each browser may use his own interpretation and implementation of the soon-to-be standard.

Event Driven

The main novelty that WebSockets provide to the world of Web browsing is the capability to provide a full-duplex (two-way) communication between the user and the server, without the first one having to install complex plug-ins and without actually knowing the underlying technology. As seen by the anterior image, we have a huge boost on the interaction between the older

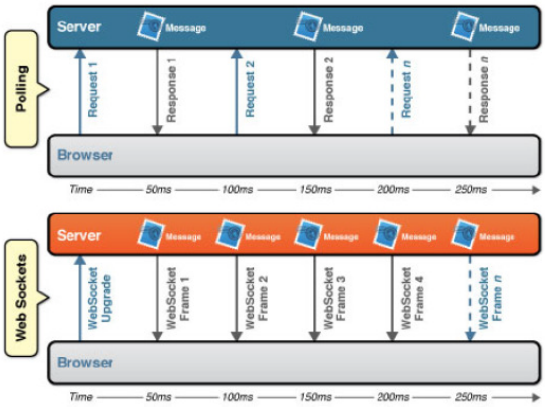


Figure 3.3: Latency comparison between the polling and WebSocket applications [LC13]

techniques (in this case, Long-term polling) and the new Web Sockets. Also, by using WebSockets (as it is a two way communication), we can exactly know when the connection has been lost and therefore inform the user or take the appropriate steps. This new functionality allows us to actually implement a full Event-driven architecture, as it is shown in [Fur10].

Security

As a new, non-fully standardised, protocol WebSockets provide a huge amount of vulnerability-prone features that should be analysed and considered before any direct use [Lai13]. We can certainly affirm, that any WebSocket connection made using the standard TLS [Mic03] encryption will be as secure as the current HTTP-based alternatives (due to the handshaking and posterior encapsulating being held on the same fashion).

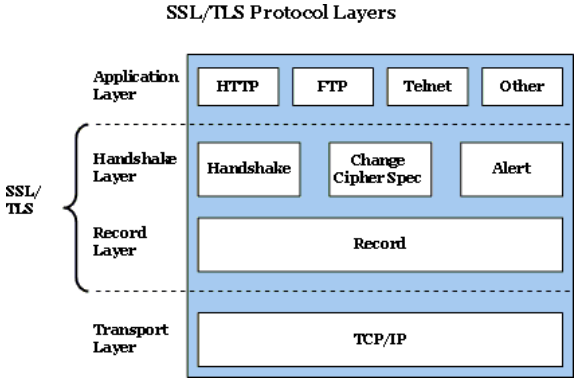


Figure 3.4: SSL/TLS Protocol Layers

The main problem associated with WebSockets may arise from its very own usage. An extensive analysis on the pros and cons of WebSockets can be found on [Erk12] upon which this section is based on. However, as far as this thesis is concerned on none of the security inconvenient should be a problem due to the fact that we're assured verified-origin content. The only notable problems that may arise are due to the possible security implementations that the browsers can implement (especially when regarding to white-list services as noted by the anterior reference).

Extendibility

In order to avoid multiple parallel developments or considering out-of-scope browsers that do not comply with the draft standard version being used [MozPM], there have been extensive developments in WebSockets emulation for non-compatible browsers [LLC12]. An example of such libraries is SockJS [Soc13], which is free to use and provide ready-to-deploy Client and Server complement that simulate using the already known and commented HTTP protocols the WebSocket interaction. Using this interfaces we could easily speed up the development by easing the actual deployed code and bringing the level of supported browsers to the levels of Internet Explorer version 5.5 which by now should represent 100% of the used mobile browsers.

WAMP – WebSocket Application Messaging Protocol

As a result of the WebSocket protocol a number of registered subprotocols appeared [MozPM]. Of those, one that is immediately interesting for our needs is meant to be WAMP. This protocol provides us exactly with the necessary structure to avoid a low-level design using WebSockets. By using the already-known standard patterns of RPC and Publish & Subscribe we can simplify the work regarding to it (mainly due to the fact that our design relies on the latest property in order to get updated information). As an added advantage, by using the technique of RPC we can simplify the communication with the server by effectively transferring all the transmissions to a WebSocket-based transmission. There are immediate inherent benefits with this approach, such as that WebDevelopers can totally abstract (and can be forced to do so) from Client – Web Server communication, only having access to a handful of methods that will provide it with data associated to it. To that extent, all plugins calls can be evaluated client-side and denied in case a plugin lacks the required privileges to access them (for example, due to a buggy plugin implementation).

3.2.3 Client-based Security

It's an concern to avoid user modification of this service due to the criticality of the underlying systems. Due to the system being based on JavaScript, as it is the very base technology of web browsing, it arises the requirement that some sort of control over JavaScript tampering should be implemented. However, that requirement is currently not possible to be complied up with because the current implementation of browser interpretation of JavaScript implies that the user can change which code is executed in their computer, and furthermore can change and execute custom code if deemed so. However, a number of options exist in order to complicate and make it a burdensome task to actually understand and modify the code in order to perform custom actions and also a number of recommendations in order to be able to limit the harm that an unauthorised user may do in the system.

- Authenticate users against a trusted-user database with assigned privileges and role levels.
- Transmit only the information that a user is available to use, therefore implying that a role changing event may represent a full reload of the user interface. This has advantages in the immediate transfer of data but may actually cause a detriment in the functionality of caching features.
- Obfuscate the JavaScript code in order to difficult user interpretation of it. There are professional tools that can do this task for us (and also the other way around), using of such would require a user to have a professional tool which should avoid the vast majority of users to even try to tamper with it. Even thought, this methods may not be enough as already demonstrated [LCDna].
- Watch for Document Changing Events (DOMEvents) and execute code when one of these occur, such as reloading the page with a totally different JavaScript (obfuscated) code or by banning the user from using the service and notifying the driver.

- Use blackbox simulating environments in order to check for possible loopholes or vulnerabilities that the developers might have not taken into account [BHS+10].
- Use immutable objects, as defined by the ECMAScript 5 standard [Int11], in order to difficult object property modification.

Ultimately all this suggestions may prove useless against a well-armed and experience system breaker, but should provide enough time or warnings for the driver to know it and to take the opportune steps. Also, by taking actions on the server-side we may effectively limit the extent of damage able to be done on the system.

3.3 State of Practise

In this section we will be analysing the Infotainment and HMI solutions already existing on the market. We will cover the ones designed and developed by major manufacturers as well as research and proposals that have not been able to get a footstep on the market.

3.3.1 Next-Generation HMI Evolution

In this section we will be analysing the different approaches proposed over the last decade regarding HMI systems development. It is key that we analyse the market in this sense and provide a general overview of the proposals as well as the their main contributions and innovations.

Nokia

Nokia started to promote an unsuccessful mobile-based HMI system back in 2010. The system they proposed had a wide range of innovating features that were not present in any system available till then, therefore becoming the first next-generation HMI system. It's important to note that their proposal, labelled "Terminal Mode", already had into consideration two key features: Manufacturer and Application Independent (thought this one ultimately was not complied with). They aimed to promote a product that would not be bound to any system already existing and to make it a basis for standardisation on the world of HMI systems. The key fact of their approach is bound to the fact that the user needed a Nokia device to ultimately use this system [BBP10], excluding therefore all the users without any device as such.

General Motors

General Motors used to have its UX design for Infotainment systems externalised to other developers. This trend changed when they decided to take a direct action on them due to the lack of brand customisation of these systems, and ultimately proved the right approach for them. Lessons that have to be learned from GM's approach is the fact that they came into the conclusion that the user's life should be shared within a car, that is, there should not be a drastic change on the user behaviour while he is driving and while he is on the real life, therefore the automotive vehicle should provide the user with a similar experience than that of his real life. [GHHW10]

An example of Unified System General Motors introduced a huge concept into the industry by defining a unified User Experience interface for all its brands, that is, all the automotive Vehicles from General Motors have a common HMI system and usability-like features. This is a key concept for the future of the industry, and one of the justifying facts for this thesis, as it shows that providing the same base system for multiple manufacturers doesn't make them unable to compete with each other. [Mot13]

BMW

In the same fashion of General Motors, BMW introduced some of the current standards back in the beginning of the century. Those include, for instance, the need to support multiple monitors or the fact that user controls should either be integrated into the automotive device or be on a set of easy-access for the user. Further envisaging into this approach, BMW has specialised into the

Figure 3.5: General Motors is an example in providing a Unified System for all it's branches [Che13]

idea behind providing a non-distracting user experience, that drastically reduces distraction of the user using in-car systems, even promoting for the fact that any user response should take less than two seconds. [NDEK09]

iDrive 4.2 and BMW Online With the advent of the current generation of mobile devices and operating systems, BMW has taken the approach to provide parallel services to either couple with those already existing on the market or co-exist safely by allowing the possibility to use the mobile services instead of the own of BMW. It is key noting the planned release of the new system "BMW Online" that would allow users to be connected with the world while the vehicle is not on drive mode, or even the fact that services such as "Google Maps" or "Apple Siri" are now able to be used within the car. [RBA13]

Audi

Traditionally aiming for an exclusive set of user experience and features, Audi has been a leading innovative business on in-car user experience. Their latest proposal therefore aims to make their vehicle social and integrated with the most common services defined by this approach such as Facebook or Twitter. Furthermore, they have totally replaced their in-house built systems by those provided by Google therefore integrating into their system the features regarding Voice-based navigation, wireless connectivity and mobile device integration. One of the lack of their systems is, though, the inability of the user to change Services Provider, being bound to those already defined. [Aud13]

Ford and Microsoft

Ford was one of the first major manufacturers to decide a total outsourcing of its services to a major software and mobile company, Microsoft. Microsoft had vast experience on the development of mobile devices for Enterprise Users and therefore appeared as a clear approach. Fruit of this agreement, Ford has been able to provide a software that had integrated phone within and that allowed the Users to use a vast number of Mobile Features (Web 1.0) while assuring itself a constant update on this rapid-development world. [Com13a]

Sued over HMI system This agreement hasn't been ultimately successful and has led to a huge downgrade on Ford rating on HMI devices. Distractions due to the system being generically mobile with a custom configuration led to multiple errors, freezing and slow response environments, and distractions that strained to a maximum extent Ford resources on this sense, ultimately leading to a joint customer complain against its defective proposal. [Rub13]

Volvo

Being one of the safest car manufacturers, Volvo traditionally hasn't provided a huge interest on enhancing and setting as a proper Infotainment system provider. [Vol13] The fact that their systems lack in such a sense features that have been long established in the industry have lead the manufacturer to couple with other developers in order to provide for a new system that tries to close the gap with other manufacturers. [McG13] The problem that has arisen with this manufacturer is the high specialisation level of their systems while not providing for a proper infotainment environment, therefore making the driving environment more dangerous and drastically downgrading the User Experience on their vehicles, ultimately leading to negative overview of their general system.

Mazda

This manufacturer has developed its infotainment system with a key objective: The ability to revamp it when necessary, that is, when next-generation HMI systems come into effect. They have provided a system itself complying with the current generation HMI but with the ability of auto-update and, literally, "Also, software is updatable, just like smartphones, through USB ports, [the] whole software can be rewritten". They have cited their lack of vision for the development of tablets and smartphones, so they took a most conservative approach while proposing this HMI that allows itself to be completely changed and expanded both in software and in hardware considerations, therefore expanding the useful lifetime of their HMI system. [Fal13]

Others

In this section we will be analysing drastic proposals on the industry of HMI systems that ultimately have resulted n failure and lack of impact, but altogether have provided for new ideas that have been adopted in other products.

i-PASSION Labelled as i-PASSION is a new and revolutionary system that aims for total user integration with real life bounding features, such as household and child management. The main aim of this proposal is to provide the user with a complete control on all the features that might distract him, such as family status, together with healthcare on the user, such as controlling the pulse. This innovative system also promoted for the need to provide for user experience also on the rest of passengers and integrating common user devices, as laptops and tablets, into the system by bounding the two devices and allowing for a range asset of tasks. [Jeo10] In resume, the most ground-breaking idea that is provided by this design revolves around driver healthcare, a fact that has not been taken into account on other systems and that ultimately derivative in the wide range of options that are proposed on this design.

CA-Fi This proposal revolves around taking an already existing Mobile System, Android, and embedding it into a structure capable of being adapted into almost any vehicle. It provides for the vast majority of Infotainment features and, as it is based on a Mobile OS, it provides access to a full set of applications as well as security updates. The main problem with this approach is the fact that the system itself is not adapted to the conditions that arise from driving, as well as providing unverified access to applications that ultimately might harm the user by provoking active distraction and, ultimately, driver accident. [IL13]

Figure 3.6: Lexus new proposal also is based on Android and has it's own marketplace [Par13]

VNC Automotive This approach deals with the coupling of a HMI device with a mobile one by providing the ability to remotely visualize and control a mobile device from the own Infotainment infrastructure. One of the key advantages of this approach is the fact that all upgrade features and application handling are phased out due to them being an integral part of the services provided by the mobile device. It also claims to provide safety control due to the software provided by OEM manufacturers ability to control the applications being used, but ultimately this might lead to a failure on identifying those or erratic behaviour, which ultimately might as well cause driver distraction. [Rea13]

3.3.2 Research that can be reused

In this section we will be analysing possible research already done in the context of Automotive Vehicles and that can be useful while developing certain aspects of this thesis. Concretely, we will be analysing proposed enhancements in the sectors of communication, interaction nd content displaying as well as different approaches based on the multiple backgrounds of the users.

Entertainment

As a key part of any infotainment system its the entertainment part. This part, subtle at the very beginning and restricted to simple functional access and abilities, has progressed to require a fully interactive system that allows the user to access and use the most common applications and experiences available to mobile devices. Too this extent, we have been able to identify two basic usage environments regarding with entertainment, the analysis of those will follow.

Classical As a concept of infotainment usage we can find usages as navigation, video display, call handling or the most typical associated with music play. This kind of applications have been long developed and tested, and the final outcome has been that of what we have available nowadays, applications with large inputs and ease of use interfaces whose functionality is extremely simple in order to avoid any kind of driver distraction. Although proven, with the advent of the new technologies and the concepts of touch screens and voice interaction, those models have become outdated and in need for revision. Recent studies [LK12] cite the advantages of concentrating the interaction on certain movements as swiping, not only on the classical applications but on an overall usage for the system.

Gaming The concept of entertaining itself is a mean to indicate distraction upon certain tasks. In this case, there's the increasing need to provide the users of the vehicle with a more pleasing experience without ever ignoring the fact that the safety of the vehicles is of utmost importance. Taking this concept in mind, we can find research [TGH+11] that has analysed and indicated multiple interaction possibilities between the users of the automotive vehicle without incurring into a high level of distraction and therefore keeping the safety standards to a recommended level.

Content Displaying

In the current state of the art on Content Interaction we can find multiple approaches, all of them having their positive and negative aspects, that define the vast majority of the user tasks while being on an Automotive Vehicle. For this extent, and due to the fact that all of them are interconnected and affect directly the user experience on such environments it is necessary to take into account the multiple research on this direction. Follows an analysis on the current methods available and an insight on techniques and best practices while displaying content and providing information to the user in order to simplify the load of unnecessary information that ultimately distracts the user from its main task, driving.

Current Input Methods There has been continuous development in the automotive industry since the arise of the Personal Navigation Devices at the beginning of the millennia, and ultimately that has lead to the integration of multiple technologies that these devices provided, such as touch and voice integration into the own vehicle. Although now having the possibility to use

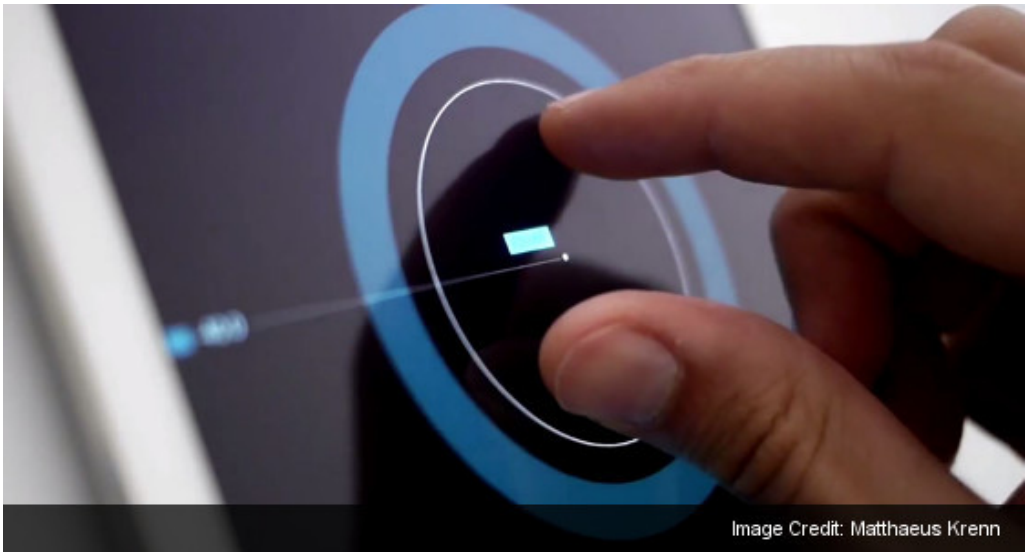


Figure 3.7: Touch Input methods are gaining increased sensibility [Kre13]

these new interaction capabilities, recent studies [dMHW+09] demonstrate that only a well integrated and shared interaction set will allow the users to obtain the maximum performance out of the system. It is important to note that integration with the market is necessary, and therefore the need to keep backwards compatibility with already existing systems, such as the physical buttons, is a need still.

Information Display Nowadays the users are subject to a constant bombardment of information which can detriment in their concentration during critical periods. This question is specially sensible when talking about the concentration capacity of a user during driving, a lack of which may lead to drastic outcomes, and therefore prioritises on most of the available researches. One of the approaches used in order to solve this problem is by summarising and awaiting the perfect moment in order to transmit the user the information. Research [RLH11] shows that by using specific algorithms is possible to transmit not so important information when the load of driver's tasks is low and therefore it is able to handle this information.

User Information Absorption Coupled with the aforementioned elements, there is another one that influences heavily the way the user gets the information being displayed. There is a full research [PSDK11] world behind this and is centred on the different locations and positioning of the multiple interaction elements, to allow for a natural flow when interacting with them and an intuitive response from the user. It is worth noting that this area is particularly interesting when considering the fact that the number of input devices is greater than a single centralised one.

Social Approach

The current world seen from an information perspective is always on a changing state, having evolved from the previous local interaction levels to a global scope. This is what we understand for Social Web and Internet overall, and therefore is important to understand the concepts and ideas behind it, as well as to adapt to it and provide support for innovation in this sense. In this section we will therefore analyse investigation done in this direction, that provides feedback on the different behaviours and need of users according to their age, experience levels in the informative environment or global communication needs.

Socializing the Automotive Environment User necessities have evolved since the last decade, from a user that required a simple functional product to one that expects to have all the information he could want at his fingertips. It is important for any product developed in this sense to be able to provide the user with this kind of information on a integrated basis, so that the user conserves the feeling of his world going at the pace he desires. This concept is further expanded

by taking into account the different information that could derive from urban areas, such as city information and news or traffic information.[SRF12] Allowing the user to always be informed on the latest events on his world will make him want to stay using the devices that provide that level of information, and therefore will ease and facilitate the use of the devices and interfaces.

Avoiding Cultural Shocks One of the key approaches of social computing and social interfaces is the fact that they are adapted to every culture by providing the information in the appropriate content and basis, as well as prioritizing it according to the cultural preferences. Ultimately, each user is different and therefore will want the environment to adapt to its own basis, but they are influenced heavily by their culture and ways of doing. Research [JRL12] shows that such affirmations are correct and that they heavily influence on the user experience even on the automotive environment with evidence showing that allowing each device and interface to be adapt to their general cultural preferences on a first glance eases the tension the user might have while using it.

Adapting to multiple Age ranges One of the major research fields on the causes of traffic accidents has to do with the variability of age on the different users and their approaches when tackling driving situations. It is important for any interface designed for all the segments to take into account these differences and provide custom feedback and responses to these segments. By being able to provide a different interaction to multiple ages we can ensure the willingness of the older sectors to obtain new devices, thus eliminating the concept of technology difficulty while also keeping the interest and usage from the younger sectors. [RS09b] Research ultimately shows that by acting as a bridge of these two sectors an interface for automotive devices would ultimately be highly successful.

3.4 Standards

In this section we will be evaluating the research done while relating to the different conditions that can arise within an automotive device. In this sense, we will be analysing the different work done in the concepts of usability and security, around whose the main aim for user satisfaction is centred about. Concretely, we will be analysing which concepts should the solution presented on this thesis take into account in order to avoid falling into design failures as the ones indicated on the previous section.

3.4.1 Usability Standards

In this section we will be exposing the research already done in the fields of usability on Automotive Environments. Concretely, we will be analysing which factors might lead to driving distraction and therefore should be taken into account on the design of any solution, as well as different interaction concepts that should be present on it. Altogether with these factors, we will be giving emphasis into simplifying the concept of driving and the interactions needed on the environment by applying the idea of a “smart“ vehicle.

Driver Distraction

In this section we will be tackling the concept of driver distraction feedback, one of the first approaches in order to keep the user notified of the distractions that he is submitted to and therefore allowing him to act upon these. These distractions might come in the basis of having the driver subject to different events from the global environment and the ones having to do with the different feedback that we will be providing. Ultimately social interaction inside the car is a subject that has already been tackled beforehand in a previous section and therefore won’t be analysed on a second basis.

Interactive Interfaces Having the user using a connected automotive environment might itself be counter-productive when talking about driver distraction and therefore its ability to respond to these events. It is important to keep a balance between the user concentration and its desire to keep connected and obtain the latest information that he is requesting. Therefore, research [LDS12] has proposed new and innovative methods to calculate such events and therefore allows design to be adapted to the negative facts that might arise from these calculations and analysis.

Driver Performance The first commitment any analysis comes into when tackling user distraction is the fact that a report is needed. The principal assumption is that the user will not be aware of its own errors and distractions and therefore reminders are needed in order for further corrections to be applied. In this sense, performance feedback might be a viable option when regarding feedback report, and research [RJL12] shows that this feedback doesn't impact negatively on the user experience except on the cases where it adds up as being an additional distraction. Therefore, this research coincides with the previous analysis showing that the interfaces need to adapt to every user needs and requirements.

Interaction

In this section we will be analysing the usability requirements related to different interaction procedures. Concretely, we will be analysing the user behaviour when changing the underlying technologies and the fact that the user might or not be concious of that. Concretely, we will be analysing the need for user to keep the interfaces adapted to what he is already used to, in order to avoid unexpected user responses. We will also be analysing upon the fact that no real standard exists when relating to user interaction on HMI devices, therefore implying that following guidelines leads to a divergent number of already existing interfaces that might influence and impact into what the user considers as a traditional interface.

Adapting to different Automotive Vehicles The number of underlying technologies used in the automotive world is greater each year, with new alternatives appearing with their benefits and disadvantages. Ultimately, it is upon the user to choose which kind of technology he desires to use, but more often than not they feel guided by the opinion of external reviews due to their lack of knowledge. This factor, while not being itself critical, can be heavily influenced by having a different interaction interface, which might lead him having insecure behaviour as research has proven. [SAA+11] This research has shown that is important to minimize the foreign sensation of a user while using a new underlying technology by providing an already known and familiar design that can guide the user through this technology transition.

Heterogeneous Interaction Design Due to the lack of standardisation in the current world of User Interaction design for HMI devices, each manufacturer takes its own research and background while designing it. Therefore, each user might have a different idea of a traditional interface and therefore might consider other devices too complicated too use or understand. While this issue has settled in the most used components, it is still a hot issue on most innovative environments, where no major opinion has been settled and therefore multiple ideas are being proposed and developed. Research and standardisation efforts are under way [RFB+13], but in the meantime user comprehension and adaptation must be noted as the preventive solution against any kind of user fears by providing transition help with the current designs.

Interface

In this section we will be considering the different research regarding user transitioning and adaptation from the devices they are currently using, be it PND, PDA, Tablet or Mobile Devices, to an infotainment based environment. For this aspect, multiple proof has been established demonstrating the different elements that need to be taken into consideration on environments for automotive devices that have to provide a user experience similar to the other devices.

Figure 3.8: Tesla Motors Infotainment on Electrical cars proposal breaks with current models [iN13]

Content Adaptation Due to the increasing urbanisation and crowding of the driving environment drivers have been treating more often than not with congestion environments in traffics jams or even red lights. On these spaces, drivers tend to use their mobile devices in order to keep connected with the world as research suggests. [AKS+10] It is important that the future of the automotive design to grow on the direction to cooperate with these devices in order to allow the user to access the information he needs, but in a controlled manner thus ensuring always the safety of the own user by avoiding the possible distractions that might arise.

Application Transfer Transferring from a mobile environment to an automotive one is not as easy as it sounds, due to the users being used to the mobile interactions and way of doing. It is important, therefore, for the automotive designs to learn from those and be able to provide the same features but in a controlled manner and space, so that they're adapted for the requirements that might arise on a car environment as research has empathised. [Son10]

Figure 3.9: Kia future vision emulates a media centre [New12]

3.4.2 Security Standards

In this section we will be analysing a shared content with the previous section regarding the concept of "Security" while on an automotive environment. Concretely, we will be observing the factors that might lead to a car accident and driver distraction from a security oriented point of view. This means, we will be giving a major emphasis on the kind of interactions disallowed during driving and the major reasoning behind their selection.

Infotainment Distractions

In this section we will be considering the multiple distractions that can arise from the use of an infotainment system, and the security risks that these distractions can cause. Furthermore, we will be analysing research showing the different security considerations between multiple countries and the need to keep track the different age of the users or impairments. We will also feature research showing a practical analysis on the interface distractions from current devices.

Standardisation per Country With the recent developments on the HMI industry, a full new world of standardisation has come suddenly into high pressure to be agreed upon. This is due to the proliferation of multiple devices and interfaces that have not the proper research and testing behind them and therefore might pose a risk to the security of the system. These standardisation processes, such as of Germany as noted on research [HGM+12], have to be keep on track for any proposal due to them having the possibility to be enforced on the very near future.

Age considerations One of the growing security risks on the road environments is that of the growing age of the driving population. This factor is of high concern due to these users not being conscious in a direct manner of their loose of features that might lead to slower response times or different impairments. Research shows, though, that by providing a system that constantly adapts to these impairments security issues can be drastically reduced. [RS09a]

Structural Distractions We consider into this category all these elements that will cause a structural distraction on the users, such as those from embedded devices or external ones but used directly by the driver. We consider into the second category devices such as PNDs or PDAs, the regulation of which is spare and therefore provide no liable standard for them. Even though most of them keeping the same design and functionality, they affect heavily on driver's reactions and behaviour and therefore lessons from this must be considered as research suggest.[KPeM+09]

Vehicle Distribution

In this section we will be analysing the outcome of multiple research on the ideas behind the location of the different devices inside a car and their distribution. Furthermore, we will gain an insight on how differently designed interfaces affect those people that have a higher degree concentrating and deducting the tasks and also will analyse the outcome of suppressing both of them and providing the user with an embedded experience using augmented reality.

Distribution on Car Device and element distribution inside a car has been since long time a standing part of the car design, that is, an immovable part that has only changed and design. With the advent of the new technologies and requirements for the environment of a car, research has been carried out[MWG+11] that has shown how much the different element and devices distribution on a car might impact heavily on the security and response of the user.

Interfaces impact Developing a bad interface can lead to disastrous effects on the user experience while driving and may cause major security issues. Therefore is important to take into account the level of concentration a user needs in order to successfully use the interface while driving, research suggests. [MCS+11] By doing successfully tested and calibrated interfaces these kind of errors could be avoided, while also providing for a better user experience as suggested by the aforementioned citation.

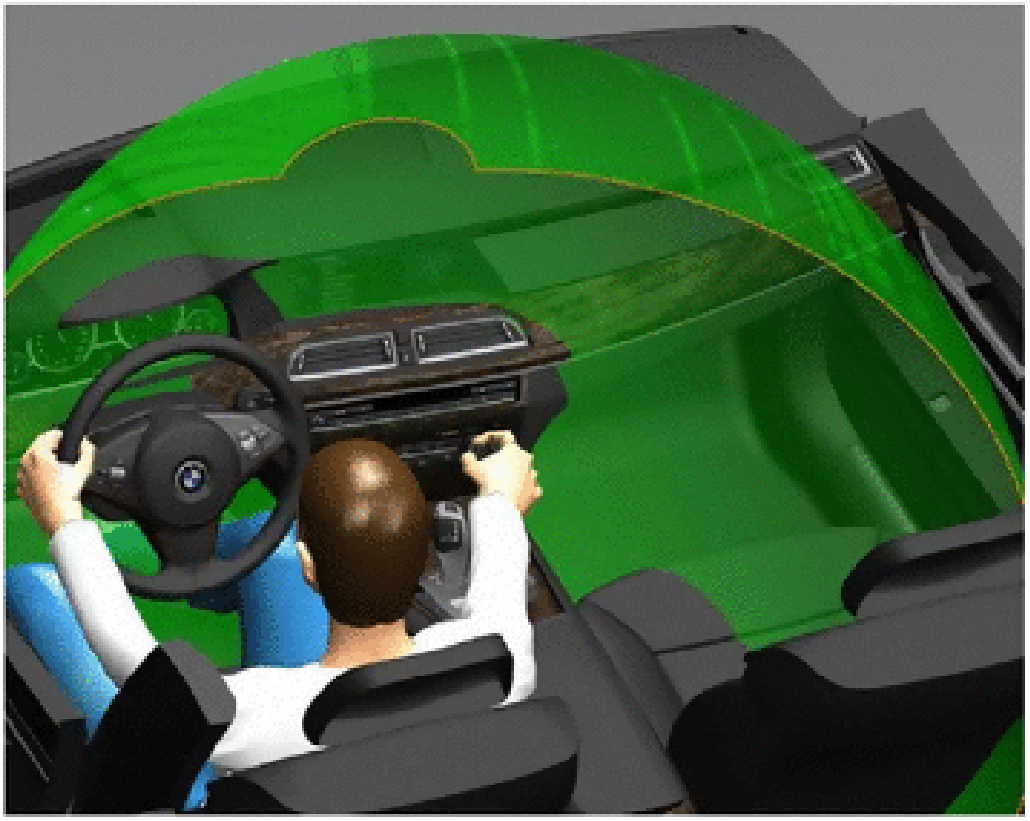


Figure 3.10: Element placement in an Automotive Vehicle is of extreme importance [RBA13]

Augmented Reality Interface One of the alternatives that have been considered and to a certain trend implemented, for instance, on rear cameras, is the use of Augmented Reality in order to quell the necessity for devices to provide for information to the user. Research on this field leads to the conclusion that using such kind of environment might even be beneficial to the user and reduce the distractions on the user by allowing a wider place-holder where to display the information.[FBH+11]

External Factors

In this section we will be analysing the major factors that lead to security disruptions, or distractions and to a major extent to driving accidents. It's worth nothing that we will be analysing research on the three main fields regarding this aspect, all of them revolving around the way the transmission of information from the user to the machine is transmitted and the way notifications are transmitted back to the user.

High demanding Interactions On the way to add extra technology support and features to an Automotive Environment we fall into more complex environments which might themselves cause major catastrophes. Research on this field[TSZ+11] suggest that a badly designed interface is likely to cause distractions and accidents on situations where a high level of concentration and fast-paced action is on need.

Input and security The way that the input concept for Automotive Devices has evolved is pretty much similar at the one that the HMI devices have evolved. This fault itself influences heavily on the fact that there's no standard when regarding to data inputs into systems. Research, though, provides us with an insight on which kind of elements[KSL13] are outdated and should not be considered as input methods because of their grave distraction levels.

Traditional methods Sticking with the traditional way of acting while driving might not be the safest approach. In this case we are referring to the fact of providing the user with a manual input method against a computer-assisted one. Research[WCN+13] shows that software that is providing direct feedback to the user and requires minimum effort to use is a better way to reduce risk on driving.

Crisis Management

In this section we will be analysing different approaches while handling the possible crisis scenarios prior to an accident. These must be tackled with extreme care and caution, because te most critical part of the system arises on these scenarios and ultimately is what assesses whether a system is appropriate for usage. Concretely, we will be observing the two very important steps needed in order to minimize the feasible effects that any accident might have upon the driver by allowing him to concentrate directly on resolving the situation.

Assessing Driver Workload One of the elements that any system should be tackling nowadays is the control on the driver's status, the so called healthcare on the driver. In order to determine which actions can be done taking into account the level of user distraction, or the corrective actions needed in order to improve it, is necessary to evaluate and quantify the level of user concentration as suggested by research into the topic.[SPB+13]

Reacting to critical situations Another of the critical steps on any automotive system is the fact that it should be able to properly act when a critical situation is being handled by helping the user in all steps possible to solve the situation or, in case that is not feasible, provide aftercare. Research indicates that a considerable danger could be providing direct feedback to the user while a situation of such is under way. [HJC10]

Environment adaptation Adapting the user environment in the case of a crisis means that the own automotive vehicle is able to modify subtle elements, such as the temperature, humidity level or control touch experience in order to ease the use of it by the user. Research in this sense show that such elements, also called driver health care, might allow to reduce the level of stress a user can have while commanding the environment.[Jeo10]

3.5 Resume

In this chapter we have overseen an overview through some products that have already been published and that we can use as evidence for their features being possible. We will use this on the next sections of Analysis and Solution, as they will become the base we will be using for proposing our solution. In addition, we have also presented research into the Infotainment fields that will contribute to our solution proposal. The experience shown on this chapter is therefore crucial to consider requirements on the proposal. Overall, we have extracted concepts such as the controls that are going to be used by the user and constraints such as keeping the traditional interfaces on the vehicle due to it being safer for the user. It has come into our sight the necessity to empathise on Security on any proposal done on this sector, all of which will also have their impact on the requirements. While there are interesting proposals, such as an Android System as a car interface, we consider that our problem is still not solved but a combination of the present solutions on the market (for example, we get the multi-device environment from BMWi and the Tablet Display from Terminal Mode of Nokia).

4 Analysis

In this chapter we will be analysing the work on the previous chapter, Related Work, and deciding the requirements and Use Cases that our proposal has to support upon the Problems and Goals exposed on the Introduction Chapter. We will explain the requirements that are considered to be needed for any successful implementation of the work as well as the cases that are most feasibly in need to be considered. In order to provide for this, we have based our analysis also on products, proposals and research already available, as explained on the chapter of Related Work, as well as focusing on following the background directives defined in the Introduction chapter. In this chapter we will be exposing the main topic of this thesis, the analysis of the requirements needed by a solution to fulfil the goals and problems exposed on the introduction chapter.

4.1 Requisite Analysis

In this section we expose the requisites that have been elicited for this thesis. The vast majority of them are functional requirements, due to it being constricted by the different environmental variables that we will see along this thesis.

4.1.1 Constraints

In this section we will be developing upon the concept of Constraints for any feasible solution that could be proposed on this thesis. An accepted definition for Constraint could be "those that define any restrictions imposed on the choices that the supplier can make when designing and developing the software. [Wes13]", therefore they are requirements needed to be complied with in order to suffice the conditions of the initial problem.

Requirement: Native device

Description: The system must be able to access device-specific functions without needing to be modified substantially, that is, only adding the corresponding extra module.
Justification: It is specified as one of the main problems and goals of for this thesis, therefore becomes a core requirement.
Satisfaction Criteria: The user can install an application on their system that provides the same features as the web version, as well as extended ones.
Satisfaction of the Client: 5
Dissatisfaction of the Client: 5
Priority: Medium
Conflicts: —
Additional Notes: —

Requirement: Push Support

Description: The system must transmit important data to the clients without requesting for it.
Justification: While closely related to an Event-based system, this one differs in that it makes explicitly the need for a non-user initiated communication.
Satisfaction Criteria: The client can receive information from the server without asking for it.
Satisfaction of the Client: 1
Dissatisfaction of the Client: 5
Priority: Medium
Conflicts: —
Additional Notes: —

Requirement: Genericness

Description: The system should be able to run in modern devices such as smartphones, tablets or net-/notebooks as well as being integrated on the car.
Justification: As one of the key goals of the thesis, this requirement is essential if any proposed architecture is supposed to satisfy our stakeholders needs.
Satisfaction Criteria: The user can use the application on his modern handheld device.
Satisfaction of the Client: 1
Dissatisfaction of the Client: 5
Priority: Medium
Conflicts: —
Additional Notes: —

4.1.2 Functional

In this section we will be specifying the core functional requirements that we deem as essentials for te development of a feasible solution. These requirements form the core of any solution or proposal and are themselves defined as "those that define what the software has to do in order for the users to accomplish their objectives" [Wes13], therefore indicating the main functionalities that any compliant solution has to have at least.

Requirement: Diagnostics

Description: The system must be able to identify problems itself is having and propose steps for the user to take in order to solve this problem.
Justification: The user needs to be able to get the problems with the system solved without the need of a technician, at least the most common ones.
Satisfaction Criteria: The user is able to run a diagnostics tool whenever he finds a problem, and the system automatically runs it when he detects any issue that requires user interaction.
Satisfaction of the Client: 5
Dissatisfaction of the Client: 5
Priority: High
Conflicts: —
Additional Notes: —

Requirement: Component Installation and Update

Description: The system must be able to add new extensions on the fly, that is, while deployed. These extensions may be in working in different contexts and with multiple aims. Also the system must be able to update them whenever new versions become available.
Justification: In order to provide an extensibility context to the architecture, we define 'components' as the key extensibility feature that will allow for expanding the system features while deployed.
Satisfaction Criteria: The user can expand the features that the system provides on the fly, without having to have the whole architecture redeployed.
Satisfaction of the Client: 3
Dissatisfaction of the Client: 5
Priority: Medium
Conflicts: —
Additional Notes: —

Requirement: Component Trusting

Description: Each installed component must go through a series of tests before being able to be deployed, that is, it must have the seal of approval of a trusted company before being able to be

installed.
Justification: In order to avoid unauthorised components (either because the author didn't authorise or because it didn't go through the necessary tests) to be installed.
Satisfaction Criteria: User can't add a non-trusted or non-authorised component to the system.
Satisfaction of the Client: 1
Dissatisfaction of the Client: 5
Priority: Medium
Conflicts: —
Additional Notes: —

Requirement: System Boot Protection

Description: The system must ensure its very own integrity, and the integrity and allowance of the components installed, in each start.
Justification: In order to compliment requirement 6 and to ensure that malware can't replace the base system with unwanted finalities.
Satisfaction Criteria: The system and its components are checked against compromise, disabling them in case of being so.
Satisfaction of the Client: 1
Dissatisfaction of the Client: 5
Priority: Medium
Conflicts: —
Additional Notes: —

Requirement: System Restore

Description: The system must be able to be restored to a previous status in case of critical unwanted modification.
Justification: In order to provide security against external attacks, a system check and restore if failed must be run on every boot (start). The system should try to restore the most recent version of itself.
Satisfaction Criteria: On a unwanted user or external modification and architecture start, the architecture must detect the changes and revert to a trusted version, even if older than the currently installed version.
Satisfaction of the Client: 4
Dissatisfaction of the Client: 2
Priority: Medium
Conflicts: —
Additional Notes: —

Requirement: System Update

Description: The system must be able to autonomously update itself when a new version is available.
Justification: In order to provide for the latest security updates and system enhancements, a self update mechanism must be provided.
Satisfaction Criteria: The system updates itself when a trusted update is present, in a way defined by the very own architecture.
Satisfaction of the Client: 2
Dissatisfaction of the Client: 4
Priority: Medium
Conflicts: —
Additional Notes: —

Requirement: License Tracking

Description: The system must install updates and new components whenever they available, as well as take care of any kind of license limitations of the installed components.
Justification: It is necessary in order to provide an autonomous component license handling system that will handle expired components and notify the user of the situation.
Satisfaction Criteria: A component is able to set an expiry date, from which the component won't work unless a license change is made.
Satisfaction of the Client: 1
Dissatisfaction of the Client: 4
Priority: Medium
Conflicts: —
Additional Notes: —

Requirement: Silent System Actions

Description: System actions that don't require user interaction should be done in the background and only notified, if so, when completed or on failure.
Justification: It is necessary to provide a cover ground for operations that could potentially increase the user feel on the application, such as component updates.
Satisfaction Criteria: Components and system can be updated on the fly and without the user having to do a direct action over it.
Satisfaction of the Client: 5
Dissatisfaction of the Client: 2
Priority: Medium
Conflicts: —
Additional Notes: —

Requirement: Plugin Support

Description: The system must provide support for 'plugins', which are extensions that act like an full fledged embedded application.
Justification: This would represent the main extensibility feature of the whole architecture, with a practically unlimited ground of possibilities.
Satisfaction Criteria: The system is able to execute plugins no matter what they are designed for, and is able to keep them under certain restricted security constraints.
Satisfaction of the Client: 1
Dissatisfaction of the Client: 5
Priority: Medium
Conflicts: —
Additional Notes: —

Requirement: Layout Support

Description: A layout component is responsible of showing the plugins arranged in multiple ways.
Justification: So that the user can organise his plugins, which need to have different display modes depending on the size, and it can customise based on the selected layout.
Satisfaction Criteria: The user can download and change the layout, which impact in the direct look and feel of the displayed architecture.
Satisfaction of the Client: 1
Dissatisfaction of the Client: 5
Priority: Medium
Conflicts: —
Additional Notes: —

Requirement: Language Support

Description: A multiple, expandable, language system must be provided so that the own architecture is translated and also offers support to the other components in this sense.
Justification: It is necessary in order to reach the greatest amount of regions and users without having to redesign or modify substantially the architecture or otherwise risk loosing clients.
Satisfaction Criteria: The user can add language packs to the deployed architecture that enhance and improve the user experience by translating the user interface to the selected language.
Satisfaction of the Client: 1
Dissatisfaction of the Client: 5
Priority: Medium
Conflicts: —
Additional Notes: —

Requirement: Style Support

Description: A component that changes directly the user interface, and therefore impacts directly in the user experience.
Justification: In order to provide for diversifying and enhancing user experience, thus avoiding the need for major revamps of the product even after long term deployment. Also, in order to provide a similar look and feel across all the components, by enabling them to adapt to the selected style.
Satisfaction Criteria: The user is able to install a new style, revamping the user look and feel.
Satisfaction of the Client: 1
Dissatisfaction of the Client: 5
Priority: Medium
Conflicts: —
Additional Notes: —

Requirement: Embedded devices detection

Description: The differentiation between a device installed on the ground and a device that is not, therefore bringing the possibility of offering different user experiences to each of them.
Justification: In order to prevent external devices from changing key features in the architecture's environment, therefore also avoiding a key security risk.
Satisfaction Criteria: An external device is identified as such and an installed device is also identified with its key differentiation.
Satisfaction of the Client: 1
Dissatisfaction of the Client: 5
Priority: Medium
Conflicts: —
Additional Notes: —

Requirement: Data Storage

Description: Saving the data to different locations on depending the certain environmental parameters, such as, but not restricted to, the type of device currently being used. Certain restrictions may apply to the components using this requirement.
Justification: In order to provide the architecture to keep the data from an execution to the next execution, for tracking or for other purposes that the components might deem to.
Satisfaction Criteria: The system must be able to save data and restore it on when the necessity arises.
Satisfaction of the Client: 1
Dissatisfaction of the Client: 5
Priority: Medium

Conflicts: —
Additional Notes: —

Requirement: Plugin execution side-by-side

Description: Running more than one plugin at the same time, while providing the same experience as of a normal plugin execution.
Justification: Enhances the user experience by allowing user multitasking, must be handled by the layout (to which in turn this requirement may apply and therefore be shown along the running plugins).
Satisfaction Criteria: The user can run more than one plugin at the same time and they offer no interference between them, furthermore they behave in the same way as if they were ran in a single basis.
Satisfaction of the Client: 5
Dissatisfaction of the Client: 1
Priority: Medium
Conflicts: —
Additional Notes: —

Requirement: Online profile support

Description: Being able to save the preferences and settings on the cloud, so that they can be autonomously used in another device without the need to reconfigure it.
Justification: Expanding user experience by storing the data in a place that is always accessible, the cloud, therefore aiding in the device transfer task, even if it is only temporary.
Satisfaction Criteria: The user can successfully log in on another device and have a similar user experience that on his own device and environment.
Satisfaction of the Client: 5
Dissatisfaction of the Client: 1
Priority: Medium
Conflicts: —
Additional Notes: —

Requirement: Purchase support

Description: An online store that allows the purchase of new features and extensions or components from any location.
Justification: In order to provide for a so called 'appstore' that represents the key pillar in most after-selling business models, therefore complying with one of the goals of the thesis.
Satisfaction Criteria: The user can access a online store and perform a purasche on a desired component or feature using an online payment method.
Satisfaction of the Client: 1
Dissatisfaction of the Client: 5
Priority: Medium
Conflicts: —
Additional Notes: —

Requirement: Settings backup and restoration

Description: Allowance to save and restore the settings on a local or online basis.
Justification: The user experience can be tampered by system crashes and posterior restoring, so an independent procedure that allows the user to recover its customizations must be provided.
Satisfaction Criteria: The user can save a backup of its settings and restore them anytime afterwards, even online (on the cloud).

Satisfaction of the Client: 3
Dissatisfaction of the Client: 5
Priority: Medium
Conflicts: —
Additional Notes: —

Requirement: Online updates and component providing

Description: Online downloading of new content that has either been purchased or is allowed by the license requirements that constrain the user.
Justification: In order to provide a key support to new component releases, this feature would greatly enhance the user experience by allowing the whole architecture to be kept up to date as long as there is a present internet connection.
Satisfaction Criteria: The system can download content from the cloud in order to update or enhance itself.
Satisfaction of the Client: 2
Dissatisfaction of the Client: 4
Priority: Medium
Conflicts: —
Additional Notes: —

4.1.3 Usability

In this section we will be analysing the diverse usability requirements that, at lest, any viable solution should have. In concrete, we can easily define as usability requirement as those that determine the acceptability of a certain product [RMC13], therefore easily obtaining the rank of the most important non-functional requirements to be fulfilled by any studied solution.

Requirement: Plugin Preview

Description: The system must be able to show critical information of the plugin without them being actively running.
Justification: In order to provide a seamless experience of the plugins being 'gadgets' on the system.
Satisfaction Criteria: The user is capable of knowing information regarding a plugin output without having to run the plugin.
Satisfaction of the Client: 1
Dissatisfaction of the Client: 5
Priority: Medium
Conflicts: —
Additional Notes: —

Requirement: Quick Installation

Description: The system must proceed extremely fast with updates.
Justification: The user experience must, by all means, not be tampered by lengthy update procedures and waiting times. Therefore, a quick updating method must be provided.
Satisfaction Criteria: The user can nearly not feel the installation of the updates apart from the system notifications, if available. That is, they last less than 5 seconds.
Satisfaction of the Client: 5
Dissatisfaction of the Client: 2
Priority: Medium
Conflicts: —
Additional Notes: —

Requirement: Execution restoration

Description: The ability to restore execution to the point where it was before system exit, with the components used and the same user interface.
Justification: Enhance the user experience by providing a work on-the-fly suspension and restoration, therefore avoiding the need the cumbersome task of restoring the components, if required.
Satisfaction Criteria: The user can leave the use of an architecture and resume it again, with the user interface being exactly the same.
Satisfaction of the Client: 5
Dissatisfaction of the Client: 1
Priority: Medium
Conflicts: —
Additional Notes: —

4.2 Use Cases

In this section we will present the set of basic use cases that we will be evaluating along the design of the thesis. Altogether with the requirements, this is of importance in the way to define a new architecture.

4.2.1 Use Case Diagram

As follows is the Use Case Diagram depicting the relationship between the different kind of users and the use cases that are available to them.

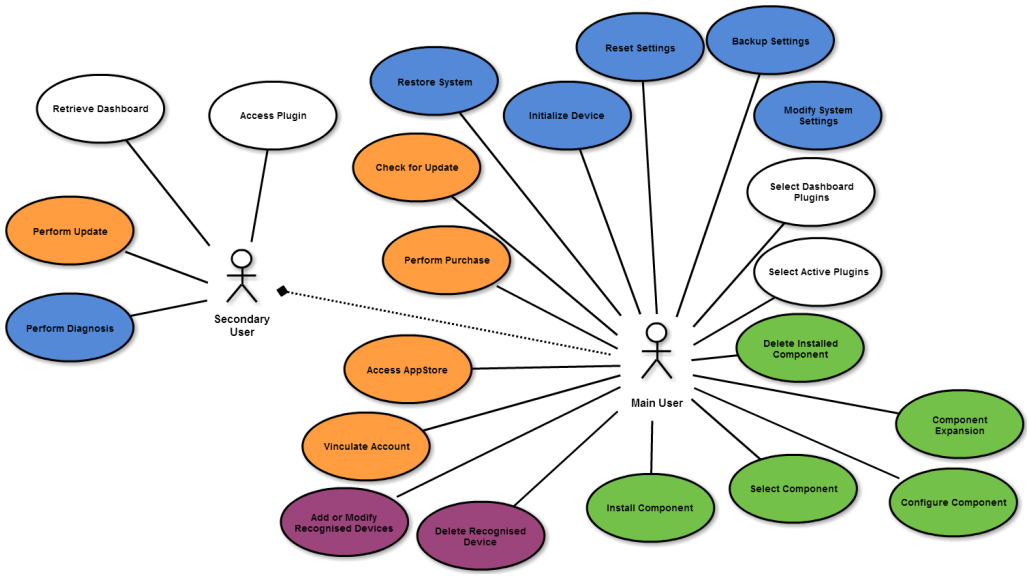


Figure 4.1: Use Case Diagram

4.2.2 Initialise Device

Main Actor: User
Precondition: The system is in a uninitialised status.
Trigger: The user tries to access the system.
Main Scenario:

- 1. The user request access to the system using a main device (a device attached to the dashboard).
- 2. The system detects that is in a uninitialised status and provides for the initialisation screen.
- 3. The user follows the instructions presented by the system and provides the adequate configuration.
- 4. The system is set to a initialised status with the configuration provided by the user. Afterwards, proceeds with the Use Case "Restore Device".

Alternative Cases:

- 4a. The configuration provided by the user is invalid and the use case returns to step 2.
- 4b./2a. The system detects that a critical component of it can't be configured, and informs the user. Afterwards it starts immediately the Use Case "Restore System".
- 4c. The system detects that a non-critical component of it can't be configured. It informs the user and proceeds with the Use Case "Retrieve Dashboard".

Satisfaction of the Client: 1
Dissatisfaction of the Client: 5
Dependencies: Restore System, Retrieve Dashboard

Conflicts: —
Additional Notes: —
Author: Josep
Validation Scenarios:

- Use 1. The user uses a main device to access the system. The system detects that it has still not been initialised and provides a screen that the user fills in with the settings it needs to be configured. The user accepts and the system proceeds to save the settings and initialised itself.
- Use 2. The user uses a main device to access the system. The system detects that it has still not been initialised and provides a screen that the user fills in with the settings it needs to be configured. The user accepts and the system detects that the settings are not correctly set, the system indicates so and the user correctly introduces the settings. Following the system saves the settings and configures itself.
- Use 3. The user uses a main device to access the system. The system detects that it has still not been initialised and provides a screen that the user fills in with the settings it needs to be configured. The system tries to configure itself but detects that a critical error has occurred with one of the components. It informs the user and proceeds with restoring itself.
- Use 4. The user uses a main device to access the system. The system detects that it has still not been initialised and that a critical component is missing or damaged. It informs the user and proceeds with restoring itself.

History: 2013
Modified: 2013

4.2.3 Retrieve Dashboard

Main Actor: User
Precondition: The system has been initialized.
Trigger: The user wants to access the system.
Main Scenario:

- 1. The user connects a device to the wireless system, or uses one that is embedded in the car.
- 2. The system recognises the user and provides for the adequate screen.

Alternative Cases:

- 2a. The system is not available and the system provides for a default screen.
- 2b. The system can't provide a content for the current user or device, the system provides information as well as registers the attempt.

Satisfaction of the Client: 1
Dissatisfaction of the Client: 5
Dependencies: —
Conflicts: —
Additional Notes: —
Author: Josep
Validation Scenarios:

- Use 1. The user uses a device to access the system. The system is available and identifies the type of device and user and provides a dashboard adapted to it.
- Use 2. The user uses a device to access the system. The system is unavailable and therefore a default information screen is displayed, urging the user to take consequent steps.
- Use 3. The user uses a device to access the system. The system detects and invalid user or device, such as those of a tampering attempt for instance, and provides information regarding this event. The system also records the data of this incident.

History: 2013
Modified: 2013

4.2.4 Restore System

Main Actor: User
Precondition: The system will always have a base restore version.
Trigger: The system detects that a System Restore has been requested.
Main Scenario:

- 1. The system detects that a System Restore procedure has been requested.
- 2. The system proceeds to check for the latest version available and downloads it.
- 3. The system shows the user a list of available versions to be restored to.
- 4. The user selects the desired version.
- 5. The system proceeds to check for the integrity of the selected version.
- 6. The system restores itself and restarts.

Alternative Cases:

- 1a. The user requests for a System Restore procedure to be done, the System requests the procedure to be done and proceeds to step 1.
- 2a. The system can't find the latest version available and proceeds to step 3.
- 2b. The system detects that it already downloaded the latest version available and proceeds to step 3.
- 5a. The system detects that the selected version is not valid: informs the user, proceeds to delete it and returns to step 2.
- 6a. The system finds a critical error while restoring, informing the user that he needs to restart the restore system procedure. The system reinitialises itself and returns to step 1.

Satisfaction of the Client: 1
Dissatisfaction of the Client: 5
Dependencies: —
Conflicts: —
Additional Notes: —
Author: Josep
Validation Scenarios:

- Use 1. The system detects at startup that a system restore has been scheduled and enters into service mode. The system checks for the last available version and downloads it. Once done, it shows the user a list of available versions capable of being restored to, whilst the user chooses one of them. After that, the system proceeds to check the selected version and afterwards starts with the system restore. After restoring, exits service mode and proceeds with the corresponding use case.
- Use 2. The user goes to the settings menu of a working system and requests the system to be restored. The system enters in service mode and checks for the last available version. The system downloads the latest available version from internet and shows the user the list of versions that it can restore to. The user selects a version and the system checks that the version is correct, in integrity terms, and proceeds to restore it. Once done the system exits service mode and presents the user with the corresponding use case.
- Use 3. The user goes to the settings menu of a working system and requests the system to be restored. The system enters in service mode and checks for the last available version. The system can't connect or check, so and shows the user the list of versions present on the system that it can restore to. The user selects a version and the system checks that the version is correct, in integrity terms, and proceeds to restore it. Once done the system exits service mode and presents the user with the corresponding use case.
- Use 4. The user goes to the settings menu of a working system and requests the system to be restored. The system enters in service mode and checks for the last available version. The system downloads the latest available version from internet and shows the user the list of versions that it can restore to. The user selects a version and the system checks that the version is correct, in integrity terms, but fails and removes the version. The system informs the user and goes back to step 2. The user selects a correct version and the system checks for the integrity of the file, succeeds, and proceeds to restore it. Once done the system exits service mode and presents the user with the corresponding use case.

- Use 5. The user goes to the settings menu of a working system and requests the system to be restored. The system enters in service mode and checks for the last available version. The system downloads the latest available version from internet and shows the user the list of versions that it can restore to. The user selects a version and the system checks that the version is correct, in integrity terms, and proceeds to restore it. The system fails on the restoration process and informs the user, afterwards proceeds to restart itself and goes to step 1. The user goes to the settings menu of a working system and requests the system to be restored. The system enters in service mode and checks for the last available version. The system downloads the latest available version from internet and shows the user the list of versions that it can restore to. The user selects a version and the system checks that the version is correct, in integrity terms, and proceeds to restore it. Once done the system exits service mode and presents the user with the corresponding use case.

History: 2013
Modified: 2013

4.2.5 Check for update

Main Actor: User
Precondition: The system is initialised and the device is identified as a main device. There is no Critical Update Procedure requested.
Trigger: The user wants the system to check for updates on the moment.
Main Scenario:

- 1. The user initiates the case by accessing the Update menu.
- 2. The user informs the system that desires it to start a Update Check procedure.
- 3. The system detects that a Check for Updates procedure has been requested.
- 4. The system connects to the internet, to a update server, and retrieves a list of the latest versions for each of the installed components (including itself). For this end it does provide data about itself and the installed components to the update server which includes their versions but may also include general data.
- 5. The system checks for outdated components and creates a list of them based on the priority to update each of them. This priority may take into account the user own update preferences.
- 6. The system provides a user with the list of available updates for current situation of the system, sorted according to their priority. This situation may be regarding the internet connectivity, energy levels or other conditions.
- 7. The user proceeds with the Perform Update Use Case.

Alternative Cases:

- 2a. The system has planned a Check for Updates task, and triggers it by requesting the task to be executed.
- 4a1. The system detects that there is no access to the internet.
- 4a1.1. The system shows the User a informative screen indicating that no internet connection could be established. It provides the user with a option to plan for a Check for Updates procedure as soon as the Internet Connection can be re-established if it is not already planned (the case will start on step 2a).
- 4a1.2. The system detects that the Check for Updates task was planned, and therefore postpones the task to the next time an Internet Connection is established (the use case will start on step 2a). The system may provide the user with a non-intrusive notification if the update settings allow it.
- 4b. The system detects that it has just Checked for Updates, therefore indicates the user that there should be no need to do it again, to avoid bandwidth costs, but offers the user to force the Check. If the user agrees, a Check for Updates procedure is requested, indicating that it needs to be forced. This alternative case is not triggered if the Check for Updates procedure is in a forced status.
- 4c. The system is informed that a critical update is available to itself. Due to this exceptional situation, the System informs the user that a critical update procedure is under way. The

system downloads the update without user interaction and plans a task to execute this update when possible, if the user update settings allow it; on base to which it also informs the user of the availability of this update. The Use Case Check for Updates is disabled while this condition is under-way.

- 5a. The system detects that the user has indicated that certain products should not be updated under the current situation. Therefore saves the data of the possible update but doesn't list it on the created update list unless the are of maximum priority.
- 7a. The system detects that the Update Procedure has not been directly initiated by the user and proceeds with the Perform Update Use Case based on the created list.
- 7b. The system detects that the Update Procedure has not been directly initiated by the user, and that the user settings forbid for automatic update procedures to be held. Therefore it will only proceed with the Perform Update Use Case on case of critical updates, otherwise only showing a notification.

Satisfaction of the Client: 3
Dissatisfaction of the Client: 3
Dependencies: Perform Update
Conflicts: Restore System
Additional Notes: —
Author: Josep
Validation Scenarios:

- Use 1. The user accesses the Update screen inside the Settings menu. The user requests for a Check for Update procedure to be held on an immediate basis. The system tries to contact the update server for new updates, and successfully establishes a connection with it. The system provides the update server with a list of the installed components as well with their versions and general data about them and the system. The update server provides a list of the available updates based on the provided data as well as the recommended update priority. Based on the user own preferences it orders the list on updates that could be held at the moment and updates that are going to be postponed. The user reviews the list and checks the updates to be performed, once acceptance a Perform Update Use Case is initiated.
- Use 2. The system detects that a Check for Updates procedure is planned to execute in the current period of time. The system tries to contact the update server for new updates, and successfully establishes a connection with it. The system provides the update server with a list of the installed components as well with their versions and general data about them and the system. The update server provides a list of the available updates based on the provided data as well as the recommended update priority. Based on the user own preferences it orders the list on updates that could be held at the moment and updates that are going to be postponed. Based on the ordered list it proceeds to initiate the Perform Update Use Case.
- Use 3. The user accesses the Update screen inside the Settings menu. The user requests for a Check for Update procedure to be held on an immediate basis. The system tries to contact the update server for new updates, but either is unable to perform the operation or detects that there is no internet connection. The system provides the user with an informative screen indicating the problem, and providing a quick shortcut to diagnose and solve internet connectivity problems. The user asks the system to perform the task at a later date, when the internet connection becomes available. The system plans for a task to be held when this conditions arises.
- Use 4. The user accesses the Update screen inside the Settings menu. The user requests for a Check for Update procedure to be held on an immediate basis. The system tries to contact the update server for new updates, but either is unable to perform the operation or detects that there is no internet connection. The system checks for the update settings and provides the user with a non-intrusive information screen if the user update settings allow so. The system plans for a task on Check for Updates to be executed as soon as Internet Connection becomes available.
- Use 5. The user accesses the Update screen inside the Settings menu. The user requests for a Check for Update procedure to be held on an immediate basis. The system detects that it has recently performed a Update Check Procedure. The system informs the user of the situation, and provides a quick overview of the planned update settings as well as the option to force a check for updates. The user requests for an Update procedure to be forced.

The system tries to contact the update server for new updates, and successfully establishes a connection with it. The system provides the update server with a list of the installed components as well with their versions and general data about them and the system. The update server provides a list of the available updates based on the provided data as well as the recommended update priority. Based on the user own preferences it orders the list on updates that could be held at the moment and updates that are going to be postponed. The user reviews the list and checks the updates to be performed, once acceptance a Perform Update Use Case is initiated.

- Use 6. The user accesses the Update screen inside the Settings menu. The user requests for a Check for Update procedure to be held on an immediate basis. The system tries to contact the update server for new updates, and successfully establishes a connection with it. The system provides the update server with a list of the installed components as well with their versions and general data about them and the system. The update server detects that the system needs for a critical update and can't serve for other updates until this update is applied. The update server informs the system of the situation. The system indicates the user that a critical update is necessary before any other Check for Updates procedure can be performed. The user accepts the update and requests for a Perform Update Use Case, which is initiated.
- Use 7. The system detects that a Check for Updates procedure is planned to execute in the current period of time. The system tries to contact the update server for new updates, and successfully establishes a connection with it. The system provides the update server with a list of the installed components as well with their versions and general data about them and the system. The update server detects that the system needs for a critical update and can't serve for other updates until this update is applied. The update server informs the system of the situation. The system proceeds to initiate on an immediate basis a Perform Update Use Case if the user update settings allow for it. Otherwise, it informs the user of the availability of this update if the user update settings allow for it.
- Use 8. The user accesses the Update screen inside the Settings menu. The user requests for a Check for Update procedure to be held on an immediate basis. The system tries to contact the update server for new updates, and successfully establishes a connection with it. The system provides the update server with a list of the installed components as well with their versions and general data about them and the system. The update server provides a list of the available updates based on the provided data as well as the recommended update priority. Based on the user own preferences it orders the list on updates that could be held at the moment and updates that are going to be postponed. If the user update settings indicate it, the system filters the update list and doesn't list the updates available for selected components unless they are deemed of maximum priority; but will always show a notification indicating that there are some components that are not checked for updates. The user reviews the list and checks the updates to be performed, once acceptance a Perform Update Use Case is initiated.
- Use 9. The system detects that a Check for Updates procedure is planned to execute in the current period of time. The system tries to contact the update server for new updates, and successfully establishes a connection with it. The system provides the update server with a list of the installed components as well with their versions and general data about them and the system. The update server provides a list of the available updates based on the provided data as well as the recommended update priority. Based on the user own preferences it orders the list on updates that could be held at the moment and updates that are going to be postponed. The system detects that no update is of critical priority and that the user settings forbid for the automatic execution of the Perform Update Use Case. Therefore it shows a notification to the user informing it about the situation.

History: 2013
Modified: 2013

4.2.6 Perform Update

Main Actor: User
Precondition: A Check for Update procedure has been held, and therefore a update list is available.
Trigger: The user requests for the Perform Update task to be held.
Main Scenario:

- 1. The user requests for a Perform Update procedure to be held on an immediate basis.
- 2. The system checks for the freshness of the Update List.
- 3. The system proceeds to contact the update server with a list of the desired updates.
- 4. The update server checks for the availability of the updates.
- 5. The update server provides the system with a list of the locations where the desired updates can be downloaded from.
- 6. The system downloads the updates and checks for the validity of it. This step can be held on a parallel basis with step 7.
- 7. The system proceeds to install the updates.
- 8. The system informs the user of the update installation operation.

Alternative Cases:

- 1a. The system detects that a Perform Update procedure has been requested.
- 2a. The system detects that the Update List is old, and proceeds to initiate for a Check for Update Use Case. After the procedure, the system proceeds with step 3.
- 2b. The system detects that a Critical Update is available and proceeds with step 3.
- 3a. The system detects that no internet connection is available.
- 3a1. The system informs the user of the situation, and gives the user the possibility to plan for a Perform Update Use Case when an Internet Connection becomes available; it will always be done if the update is critical.
- 3a2. The system detects that the perform update procedure was planned. If the user settings allow for so, performs to plan for the task to be held once Internet Connection becomes available, always if the update is critical, otherwise it informs the user of the situation.
- 4a. The update server detects that certain requested updates are outdated, or that the update list is itself outdated or invalid and informs the system.
- 4a1. The system informs the user of the situation, and gives the user the possibility to execute a Check for Update Use Case; it will always be done if the update is critical.
- 4a2. The system detects that the perform update procedure was planned. If the user settings allow for so, performs to execute the Check for Update use case, always if the update is critical, otherwise it informs the user of the situation.
- 6a. The system can't download for a update or detects that an update is corrupt, and notifies the update server. The update server provides for an alternate source and the system restarts the step 6 for this update. In case there is no alternate source the system proceeds with the next update.
- 7a. The system detects that a update requires certain components (including system components) that are in use. The system marks this components as ready for update and programs an update procedure on them (step 7) once they become free to update.
- 7b. The system can't install an update. The system proceeds with next update (step 6) and postpones till the end. If it is already the end, proceeds to step 8.
- 8a. The system detects that the procedure was planned, and therefore it proceeds to show the user a notification about the success of the update operation.
- 8b. The system detects that a system restart is required, due to the need of critical core components to be updated. The system proceeds to restart itself if the conditions are adequate.

Satisfaction of the Client: 1
Dissatisfaction of the Client: 5
Dependencies: Check for update
Conflicts: —
Additional Notes: The situation 7b. is not expected to last to a point where it is the only update

left to install. This use case is designed having in mind the different component dependencies and therefore might not be able to update all components at once, but instead require certain update of the other components before a update might even be applied.
Author: Josep
Validation Scenarios:

- Use 1. The user accesses the Update screen inside the Settings menu. The user requests for a Perform Update procedure to be held on an immediate basis. The system checks for the freshness of the selected updates list. The system proceeds to contact the update server with the list of selected updates, and this last one checks for the correctness of it as well as for the availability of the listed updates. The update server reports back to the system with a list of the servers where it can download the desired updates from. The system proceeds to download each of the updates from the provided list and check for the correctness of them. Independently, once a update becomes ready it proceeds to install it performing the necessary procedures. Once this process has been ended the system shows the user a list of the processed updates with their final outcome.
- Use 2. The system detects that a Perform Update procedure is planned to execute in the current period of time. The system checks for the freshness of the selected updates list. The system proceeds to contact the update server with the list of selected updates, and this last one checks for the correctness of it as well as for the availability of the listed updates. The update server reports back to the system with a list of the servers where it can download the desired updates from. The system proceeds to download each of the updates from the provided list and check for the correctness of them. Independently, once a update becomes ready it proceeds to install it performing the necessary procedures. Once this process has been ended the system shows the user a notification with a quick overview of the update procedure, allowing for a complete overview should the user want to.
- Use 3. The user accesses the Update screen inside the Settings menu. The user requests for a Perform Update procedure to be held on an immediate basis. The system checks for the freshness of the selected updates list, and detects that the list is outdated or not correct. The system informs the user of the situation and proceeds to initiate the Check for Update Use Case. After the procedure, the system proceeds to contact the update server with the list of selected updates, and this last one checks for the correctness of it as well as for the availability of the listed updates. The update server reports back to the system with a list of the servers where it can download the desired updates from. The system proceeds to download each of the updates from the provided list and check for the correctness of them. Independently, once a update becomes ready it proceeds to install it performing the necessary procedures. Once this process has been ended the system shows the user a list of the processed updates with their final outcome.
- Use 4. The user accesses the Update screen inside the Settings menu. The user requests for a Perform Update procedure to be held on an immediate basis. The system detects that a critical update is requested. The system proceeds to contact the update server with the list of selected updates, and this last one checks for the correctness of it as well as for the availability of the listed updates. The update server reports back to the system with a list of the servers where it can download the desired updates from. The system proceeds to download each of the updates from the provided list and check for the correctness of them. Independently, once a update becomes ready it proceeds to install it performing the necessary procedures. Once this process has been ended the system shows the user a list of the processed updates with their final outcome. Due to the update being critical, it shows the user a notification stating that the system will enter the update procedure once the adequate conditions are met.
- Use 5. The user accesses the Update screen inside the Settings menu. The user requests for a Perform Update procedure to be held on an immediate basis. The system checks for the freshness of the selected updates list. The system proceeds to contact the update server with the list of selected updates, but detects that no internet connection is possible. The system shows the user the possibility of programming the execution of the Perform Update task when an internet connection becomes available and shows the user the possibility to run a diagnostics on internet connection.
- Use 6. The system detects that a Perform Update procedure is planned to execute in the current period of time. The system checks for the freshness of the selected updates list. The

system proceeds to contact the update server with the list of selected updates, but detects that no internet connection is possible. The system proceeds to plan the perform update procedure to execute when an Internet Connection becomes available.

- Use 7. The user accesses the Update screen inside the Settings menu. The user requests for a Perform Update procedure to be held on an immediate basis. The system checks for the freshness of the selected updates list. The system proceeds to contact the update server with the list of selected updates, and this last one checks for the correctness of it and detects it as invalid or outdated. The system informs the user of the situation, and proceeds to execute a check for updates procedure. Once ended, it proceeds to send again the list of desired updates to the update server, which check for the correctness of it as well as for the availability of the listed updates. The update server reports back to the system with a list of the servers where it can download the desired updates from. The system proceeds to download each of the updates from the provided list and check for the correctness of them. Independently, once a update becomes ready it proceeds to install it performing the necessary procedures. Once this process has been ended the system shows the user a list of the processed updates with their final outcome.
- Use 8. The system detects that a Perform Update procedure is planned to execute in the current period of time. The system checks for the freshness of the selected updates list. The system proceeds to contact the update server with the list of selected updates, and this last one checks for the correctness of it as well as for the availability of the listed updates. The update server reports that the list is outdated and the system proceeds to start a check for updates procedure, once done proceeds to contact again the update server which checks for the correctness of it and the availability of the updates. The update server reports back to the system with a list of the servers where it can download the desired updates from. The system proceeds to download each of the updates from the provided list and check for the correctness of them. Independently, once a update becomes ready it proceeds to install it performing the necessary procedures. Once this process has been ended the system shows the user a notification with a quick overview of the update procedure, allowing for a complete overview should the user want to.
- Use 9. The user accesses the Update screen inside the Settings menu. The user requests for a Perform Update procedure to be held on an immediate basis. The system checks for the freshness of the selected updates list. The system proceeds to contact the update server with the list of selected updates, and this last one checks for the correctness of it as well as for the availability of the listed updates. The update server reports back to the system with a list of the servers where it can download the desired updates from. The system proceeds to download each of the updates from the provided list and check for the correctness of them. For a given update, the system detects that the update is not correct. It proceeds to contact to the update server and notify the problem, and request anew for the update. The system downloads again the update but detects again the same problem, and the update server notifies the system that there is no alternate source to download from; the system proceeds with the next update. Independently, once a update becomes ready it proceeds to install it performing the necessary procedures. Once this process has been ended the system shows the user a list of the processed updates with their final outcome. The system makes emphasis on the failure to install an update and informs the user that this problem will be reviewed.
- Use 10. The user accesses the Update screen inside the Settings menu. The user requests for a Perform Update procedure to be held on an immediate basis. The system checks for the freshness of the selected updates list. The system proceeds to contact the update server with the list of selected updates, and this last one checks for the correctness of it as well as for the availability of the listed updates. The update server reports back to the system with a list of the servers where it can download the desired updates from. The system proceeds to download each of the updates from the provided list and check for the correctness of them. Independently, once a update becomes ready it proceeds to install it performing the necessary procedures. The system detects that a component can't be updated and marks it to be installed once all other updates are installed. Once this process has been ended the system proceeds to install the update that couldn't be updated, and after that the system shows the user a list of the processed updates with their final outcome.
- Use 11. The user accesses the Update screen inside the Settings menu. The user requests

for a Perform Update procedure to be held on an immediate basis. The system checks for the freshness of the selected updates list. The system proceeds to contact the update server with the list of selected updates, and this last one checks for the correctness of it as well as for the availability of the listed updates. The update server reports back to the system with a list of the servers where it can download the desired updates from. The system proceeds to download each of the updates from the provided list and check for the correctness of them. Independently, once a update becomes ready it proceeds to install it performing the necessary procedures. The system detects that the component is in use and marks this component as ready for update. Once this process has been ended the system shows the user a list of the processed updates with their final outcome. Once the component is closed the system proceeds to update it before the user can use it.

History: 2013
Modified: 2013

4.2.7 Install Component

Main Actor: User
Precondition: The system has been initialized, and the component is available to the user for installation.
Trigger: The user requests for a certain component to be installed.
Main Scenario:

- 1. The user requests for a component installation.
- 2. The system contacts the installation server and forwards the installation request, which checks whether the user is eligible for the component installation.
- 3. The installation server provides the location of the component's package altogether with the license for it.
- 4. The system downloads both the component and the license.
- 5. The system installs the component.
- 6. The system informs the user of the status of the installation and about the license that has been installed and allows the component to be used.

Alternative Cases:

- 1a. The system asks for a component installation.
- 2a. The system can't contact the installation server, therefore plans the component installation till when the internet connection is re-established and notifies the user.
- 2b. The installation server detects that the installation request is invalid, therefore denying the request and notifying the system in the process. The system informs the user about the failed request and provides alternate solutions.
- 4a. The system can't download the packages provided by the installation server. The system notifies the error and contacts again the installation server asking for a new package location. The installation server provides for a new package location and returns to step 4; if no alternate package location is available it notifies the system about it and this last notifies the user.
- 5a. Te system can't install the component. If this situation arises the system notifies the installation server and also the user of the problem.

Satisfaction of the Client: 1
Dissatisfaction of the Client: 5
Dependencies: Perform Purchase
Conflicts: —
Additional Notes: In this use case, the steps 3 and 4 might be specified separately. Concretely, both the license and the component may be located on different servers, but that will be defined by the solution.
Author: Josep
Validation Scenarios:

- Use 1. The user initiates a request for a component installation. The system start the request by contacting the installation server and transferring the installation request. Following this

step, this last checks that the installation request is valid and succeeds, so it provides the system back with the location of the component package and license. The system proceeds to successfully download and afterwards install the package and license without trouble. As a next step, it informs the user about the success of the operation and makes the component ready to use.

- Use 2. The system detects that a component installation has been requested. The system start the request by contacting the installation server and transferring the installation request. Following this step, this last checks that the installation request is valid and succeeds, so it provides the system back with the location of the component package and license. The system proceeds to successfully download and afterwards install the package and license without trouble. As a next step, it informs the user about the success of the operation and makes the component ready to use.
- Use 3. The system detects that a component installation has been requested. The system start the request by contacting the installation server and transferring the installation request. However, the system can't transfer the request and therefore postpones the component installation task till a internet connection can be re-established. The system notifies the user that the installation will be postponed.
- Use 4. The user initiates a request for a component installation. The system start the request by contacting the installation server and transferring the installation request. Following this step, this last checks that the installation request is valid and fails, saving the request for further review. The system informs the user that a problem with the component installation has occurred, and provides a diagnostic shortcut to help him solve this problem.
- Use 5. The user initiates a request for a component installation. The system start the request by contacting the installation server and transferring the installation request. Following this step, this last checks that the installation request is valid and succeeds, so it provides the system back with the location of the component package and license. The system proceeds to download the elements but fails. It notifies the installation server and asks for a new location. The server notifies that no alternate location is available so the system postpones the installation to a further date and notifies the user.
- Use 6. The user initiates a request for a component installation. The system start the request by contacting the installation server and transferring the installation request. Following this step, this last checks that the installation request is valid and succeeds, so it provides the system back with the location of the component package and license. The system proceeds to successfully download and afterwards install the package, but detects a problem while doing so. Notifies both the installation server and the user and postpones the component installation task to a further date.

History: 2013
Modified: 2013

4.2.8 Component feature Addition

Main Actor: User
Precondition: The system is initialised, the component is installed and with the adequate permissions, the user settings allow this Use Case to be held.
Trigger: The user starts a request for a component expansion.
Main Scenario:

- 1. The user starts a component expansion request.
- 2. The system detects that a request has been initiated and contacts the installation server with the expansion request.
- 3. The installation server checks that the expansion request is valid and has been authorised.
- 4. The installation server provides the system with the expansion package location.
- 5. The system downloads the expansion package with the adequate component expansion license.
- 6. The system installs the expansion package and registers the new expansion license.
- 7. The component is notified of the new expansion package.
- 8. The system notifies the user about the expansion procedure.

Alternative Cases:

- 1a. A component requests for a component expansion.
- 2a. The system can't contact the installation server. Notifies the user of the situation and postpones the operation till a later stage.
- 3a. The installation server denies the expansion request. Therefore notifies the system of the failure, which subsequently notifies the user.
- 5a. The system can't download the package from the established location, and thus informs the installation server of this issue. The installation server provides for an alternate location and the system downloads the package from it. In case that the installation server can't provide for another package location, the it informs the system which in turns also notifies the user.
- 6a. The system encounters an installation problem and notifies the user, the component and the installation server. It also postpones the component expansion.

Satisfaction of the Client: 5
Dissatisfaction of the Client: 1
Dependencies: Component installation
Conflicts: —
Additional Notes: This use case might also be used to update the own component, therefore replicating the Check for Updates Use Case in a minor scale.
Author: Josep
Validation Scenarios:

- Use 1. The user starts a component expansion request. The system starts a component expansion request by querying the installation server for the expansion package and license. The installation server provides for the expansion package and license and the system downloads successfully these elements. It proceeds to install the elements, with success, and afterwards informs the component of the installation result. After, it also informs the user of the outcome of the procedure and the readiness of the component to use the new content.
- Use 2. The system detects that a component expansion request has been issued by a component and starts it by querying the installation server for the expansion package and license. The installation server provides for the expansion package and license and the system downloads successfully these elements. It proceeds to install the elements, with success, and afterwards informs the component of the installation result. After, it also informs the user of the outcome of the procedure and the readiness of the component to use the new content.
- Use 3. The user starts a component expansion request. The system starts a component expansion request by querying the installation server for the expansion package and license. Unfortunately the query is dismissed because the system is unable to reach the installation server. The component request is postponed and the user, who is also notified, given the possibility to dismiss it any time.
- Use 4. The user starts a component expansion request. The system starts a component expansion request by querying the installation server for the expansion package and license. The installation server processes the request and issues a denial. The system receives the outcome and notifies both the component and the user.
- Use 5. The user starts a component expansion request. The system starts a component expansion request by querying the installation server for the expansion package and license. The installation server provides for the expansion package and license but the system can't download these. Therefore the system queries the installation server for another location, but this one notifies the system that no alternate installation location is available. Due to this problem the component expansion is postponed and the user given the ability to cancel it any time.
- Use 6. The user starts a component expansion request. The system starts a component expansion request by querying the installation server for the expansion package and license. The installation server provides for the expansion package and license and the system downloads successfully these elements. It proceeds to install the elements but fails. Due to this failure the component, the user and the installation server are notified; while also postponing the expansion procedure to a further date.

History: 2013
Modified: 2013

4.2.9 Perform Diagnosis

Main Actor: User
Precondition: —
Trigger: A diagnosis has been requested.
Main Scenario:

- 1. The user requests a diagnosis on a certain problem.
- 2. The system provides for a diagnosis screen where it shows the checks on that field in general, querying a help server if possible, and the progress on them.
- 3. The system processes all the checks and encounters one or several errors.
- 4. The system provides the user with the option to try to apply several automatic fixes. For each problem, it also provides the user with an informative sheet about why that problem may have appeared and a list of related community questions on that aspect.
- 5. The user tells the system to proceed with applying the automated fixes.
- 6. The system does a set of fixes according to the problems detected and informs the user of the outcome.
- 7. The user checks the review and accepts the outcome of the diagnosis.

Alternative Cases:

- 1a. The system encounters an error while performing one critical task and automatically opens the diagnosis screen.
- 3a. The system can't find any problem regarding a general issue, giving the ability to forcefully apply fixes. Nevertheless, it shows an informative screen regarding possible causes for a problem and general check procedures and provides the user with contact information for further technical review.
- 4a. The system encounters several problems but can't find any automated fix for them. It provides the user with the opportunity to search online for an automated fix and an informative page about possible causes for the problem.
- 6a. The system can't apply one or more automated fixes for a concrete problem, notifying the user of it once all automated fixes have been applied.
- 7a. The user may choose to restart the case by checking again for the same problem or another one, thus starting again on step 2.

Satisfaction of the Client: 5
Dissatisfaction of the Client: 5
Dependencies: Check for update
Conflicts: —
Additional Notes: It's worth noting that this Use Case must be available even when the system has not been initialised.
Author: Josep
Validation Scenarios:

- Use 1. The user accesses the diagnosis screen. The user selects the problem he faces from a list of system-available problems, optionally searching online for new definitions/community support. The system then creates a list of common checks for the problem and finding one or more errors, checking from the help server on the internet if available. The user reviews the list of checks and selects the fixes that desires to be applied, once confirmation the system starts applying the fixes one by one. Once ended, the system shows the user with an informative screen regarding the fixes applied, why they were applied and the outcome of those. Additionally, it also provides the user with a quick link to a community website where similar issues can be found.
- Use 2. The system detects that a critical error has occurred and automatically starts the diagnosis procedure. The system then creates a list of common fixes for the problem, checking from the help server on the internet if available. The system starts applying the fixes one by one. Once ended, the system shows the user with an informative screen regarding the fixes

applied, why they were applied and the outcome of those. Additionally, it also provides the user with a quick link to a community website where similar issues can be found.

- Use 3. The user accesses the diagnosis screen. The user selects the problem he faces from a list of system-available problems, optionally searching online for new definitions/community support. The system then creates a list of common checks for the problem, checking from the help server on the internet if available. After processing the checks, the system realises that no error was found, and therefore informs the user of this outcome. The system offers the user a list of fixes that can be forcefully applied, as well with a link to related issues on the community.
- Use 4. The user accesses the diagnosis screen. The user selects the problem he faces from a list of system-available problems, optionally searching online for new definitions/community support. The system then creates a list of common checks for the problem, checking from the help server on the internet if available. After processing the checks, the system realises that no error was found, and therefore informs the user of this outcome. The system offers the user a list of fixes that can be forcefully applied, as well with a link to related issues on the community.
- Use 5. The user accesses the diagnosis screen. The user selects the problem he faces from a list of system-available problems, optionally searching online for new definitions/community support. The system then creates a list of common checks for the problem and finding one or more errors, checking from the help server on the internet if available. Additionally, the system can't find any related fixes for this problem, so it provides the user with a list of checks and community questions related to this problem.
- Use 6. The user accesses the diagnosis screen. The user selects the problem he faces from a list of system-available problems, optionally searching online for new definitions/community support. The system then creates a list of common checks for the problem and finding one or more errors, checking from the help server on the internet if available. The user reviews the list of checks and selects the fixes that desires to be applied, once confirmation the system starts applying the fixes one by one. The system encounters an error while applying one or more fixes, and notes them down. Once ended, the system shows the user with an informative screen regarding the fixes applied, why they were applied and the outcome of those. It also shows a list of the fixes that it was not able to apply and the common problems related to those. Additionally, it also provides the user with a quick link to a community website where similar issues can be found.
- Use 7. The user accesses the diagnosis screen. The user selects the problem he faces from a list of system-available problems, optionally searching online for new definitions/community support. The system then creates a list of common checks for the problem and finding one or more errors, checking from the help server on the internet if available. The user reviews the list of checks and selects the fixes that desires to be applied, once confirmation the system starts applying the fixes one by one. Once ended, the system shows the user with an informative screen regarding the checks applied, why they were applied and the outcome of those. Additionally, it also provides the user with a quick link to a community website where similar issues can be found. The user decides that the outcome of the diagnosis is not satisfactory and decides to restart it for the same problem or for similar problems suggested by the system; or for another totally different problem.

History: 2013
Modified: 2013

4.2.10 Reset Settings

Main Actor: User
Precondition: —
Trigger: The user wants to force a global reset of the system settings.
Main Scenario:

- 1. The user accesses the Reset Settings screen.
- 2. The system provides the user with an overall information about the settings that will be erased, how they will be returned to the defaults and the overall consequences of the action. Additionally, it is offered on which settings not to remove (from system or components).

- 3. The user reviews the information and proceeds through a double confirmation procedure.
- 4. The system erases the settings and shows the user a resume screen.
- 5. The user reviews the outcome of the operation and accepts.
- 6. The system proceeds to reload all the components with the new default settings.

Alternative Cases:

- 1a. The system is forced to reset the settings and starts the reset settings (for example, after an update procedure).
- 2a. As it is a forced procedure, the system proceeds without user confirmation. This is only available for system settings reset.
- 4a. The system can't erase some settings on the moment, so a settings remove will be planned after the system restarts itself.
- 5a. As it is a forced procedure, the system proceeds without user confirmation. This is only available for system settings reset.
- 6a. If there are some components that couldn't have their settings removed, the system proceeds to close the components and try again. If that's not possible, a task will be planned for system restart.
- 6b. After a system settings reset, the system will be automatically restarted. Once restart, a Initialise Device Use Case will be carried out.

Satisfaction of the Client: 1
Dissatisfaction of the Client: 5
Dependencies: Initialise Device
Conflicts: Restore system, Backup Settings
Additional Notes: It's worth noting that this Use Case must be available even when the system has not been initialised.
Author: Josep
Validation Scenarios:

- Use 1. The user accesses the Reset Settings menu inside the settings screen. The system provides for a list of all the installed components that currently have saved settings on the system, as well as with the settings that the own system has and the size of each of them. The system also provides comprehensive list of the general settings that each component might be able to save, allowing for selective removal of settings and features. The user selects the settings that wants to remove and, afterwards, accepts. After a double-confirmation procedure the system proceeds to delete every feature; on confirmation a screen will be shown to the user indicating the outcome of the operation. After the user accepts the results, the system proceeds to reload every component that has had its settings modified.
- Use 2. The system detects that a forced settings reset procedure has been initiated. The system proceeds to enumerate and list the settings to be deleted, showing the user the list of settings to be deleted and the data that will be lost. Once the listing is complete, the system proceeds to erase all the listed settings. When the settings have been deleted the system will show a screen stating the results. Afterwards, a system reload will be carried out. Once this is done the system will start the Initialise Device Use Case.
- Use 3. The user accesses the Reset Settings menu inside the settings screen. The system provides for a list of all the installed components that currently have saved settings on the system, as well as with the settings that the own system has and the size of each of them. The system also provides comprehensive list of the general settings that each component might be able to save, allowing for selective removal of settings and features. The user selects the settings that wants to remove and, afterwards, accepts. After a double-confirmation procedure the system proceeds to delete every feature but finds out that can't delete some, so they will be marked as such; on confirmation a screen will be shown to the user indicating the outcome of the operation. After the user accepts the results, the system proceeds to reload every component that has had its settings modified. For those components whose settings couldn't be removed, they will be restarted having their settings removed after they are shut down. If that's not possible, then they will be marked to have their settings removed upon system restart.

History: 2013
Modified: 2013

4.2.11 Backup Settings

Main Actor: User
Precondition: The system has been initialised.
Trigger: The user wants to save his settings on a secondary system.
Main Scenario:

- 1. The user accesses the Backup Settings menu from the Settings screen.
- 2. The user indicates the system that wants to do a system backup.
- 3. The system indicates the user the available methods to do a system backup, the locations where this data will be stored as well as the conditions of use of these methods.
- 4. The user selects a methods and provides for the necessary configuration settings for the method including the components and settings that will be on the backup. Afterwards, the user confirms the order.
- 5. The system proceeds to save the settings using the selected method.
- 6. Once the settings have been saved, the system provides the user with a resume of the operation.
- 7. The user checks the outcome of the operation and finalises the use case.

Alternative Cases:

- 1a. The system detects that an Backup Settings task was planned.
- 2a/3a/4a. In case of an automatic Backup Settings, the system already has the backup method selected and configured.
- 2b. The user indicates the system that wants to do an already configured backup. The use case jumps to the step 5.
- 5a. The system finds an error while saving the settings using the provided method. On that case, the system notifies the user of the failure to do so and returns to step 3.
- 5b. In case of an automatic Backup Settings the system might resort to a default backup method if it finds an error while saving and the settings allow so, otherwise informs the user and finalises the use case.
- 5c/6a/7a. The user might choose to exit the menu any time, therefore the system will provide all communications via notifications.

Satisfaction of the Client: 5
Dissatisfaction of the Client: 2
Dependencies: Restore Settings
Conflicts: Reset Settings
Additional Notes: —
Author: Josep
Validation Scenarios:

- Use 1. The user proceeds to the Backup Settings menu. The user indicates the system that it desires to do a Backup Settings procedure. The system proceeds to provide the user with a list of the available backup methods along with information regarding each method and the conditions of use of it. The user chooses from within this list for a backup method and proceeds. The system provides the user with the configuration settings for the backup method. After the user selects the components and settings to backup along with the configuration settings, the system proceeds to save the backup using those. After ending, the system shows the user with a resume screen showing the output of the operation and the location of the new backup. The user reviews it and accepts the review.
- Use 2. The system detects that an automatic Backup Settings procedure has been requested. The system selects automatically the backup method based on the configured settings, as well as the method's configuration. Afterwards, the system proceeds to backup the data using the selected conditions and elements. After the backup operation has ended the system shows a notification to the user showing the outcome of the operation and the different data that was recorded.
- Use 3. The user proceeds to the Backup Settings menu. The user indicates the system that it desires to do a Backup Settings procedure. The system proceeds to provide the user with a list of the available backup methods along with information regarding each method and the

conditions of use of it. The user chooses from within this list for a backup method and proceeds. The system provides the user with the configuration settings for the backup method. After the user selects the components and settings to backup along with the configuration settings, the system proceeds to save the backup using those. During the saving operation the system finds an error. The system notifies the user of the error and the user proceeds to select another backup method. After configuring the method the system retries to save the settings this time with success. After ending, the system shows the user with a resume screen showing the output of the operation and the location of the new backup. The user reviews it and accepts the review.

- Use 4. The system detects that an automatic Backup Settings procedure has been requested. The system selects automatically the backup method based on the configured settings, as well as the method's configuration. Afterwards, the system proceeds to backup the data using the selected conditions and elements. The system encounters an error while doing the backup operation, so it resorts to an alternate method if configured; otherwise it notifies the user of the failure and programs the operation for a later execution. After the backup operation has ended the system shows a notification to the user showing the outcome of the operation and the different data that was recorded.
- Use 5. The user proceeds to the Backup Settings menu. The user indicates the system that it desires to do a Backup Settings procedure. The system proceeds to provide the user with a list of the available backup methods along with information regarding each method and the conditions of use of it. The user chooses from within this list for a backup method and proceeds. The system provides the user with the configuration settings for the backup method. After the user selects the components and settings to backup along with the configuration settings, the system proceeds to save the backup using those. The user decides to exit the menu and continue manual operation. After the system ends the backup procedure, the system notifies the user of the outcome of it.
- Use 6. The user proceeds to the Backup Settings menu. The user indicates the system that it desires to do a Backup Settings procedure. The system proceeds to provide the user with a list of the available backup methods along with information regarding each method and the conditions of use of it. The user chooses from within this list for a custom backup method containing all the settings and proceeds. The system proceeds to save the backup using those. After the system ends the backup procedure, the system notifies the user of the outcome of it.

History: 2013
Modified: 2013

4.2.12 Restore Settings

Main Actor: User
Precondition: —
Trigger: The user wants to restore the settings.
Main Scenario:

- 1. The user accesses the Restore Settings menu from the Settings screen.
- 2. The system offers the user a list of the restore settings methods that are available to it along with the backups that each one has.
- 3. The user selects a backup from the list along with the method used.
- 4. The user goes through a double-confirmation procedure after selecting which settings will be restored, also for components.
- 5. The system starts the restoration procedure and provides the user with a resume of the restore operation when ended.
- 6. The user reviews the restore operation and confirms it.
- 7. The system reloads the restored components.

Alternative Cases:

- 1a. The system starts a automatic Restore Settings procedure after a certain Update or Restore procedure. A migrate procedure may also apply.

- 2a/3a/4a. On the case of an automated Restore Settings procedure the restore method, backup and settings may automatically be selected, thus avoiding these steps and going directly to step 5.
- 5a. The system may encounter an error while restoring the settings. If that's the case the system must roll back the changes already done and go back to step 2a.
- 5b. In case of an automatic procedure, if the system encounters an error then it must start the case Reset Settings.
- 7a. The operation might require some critical components to be reloaded, the system then plans for a system restart after the procedure.
- 7b. If the operation is planned by the system, then the restart is done automatically.

Satisfaction of the Client: SatisfClient
Dissatisfaction of the Client: DissatClient
Dependencies: —
Conflicts: —
Additional Notes: This procedure may have an special behaviour on update to migrate the backups.
Author: Josep
Validation Scenarios:

- Use 1. The user starts the use case by going to the Restore Settings menu through the settings screen. The user then asks the system for a list of Restore Settings backups. The system provides the user with a list of Restore Settings backups along with the methods used and the settings that have been stored in each of those. The user then selects the desired backup and the settings to be restored along it and goes through a double confirmation procedure to start the task. The system then proceeds to restore the backup settings. Once completed, the system provides the user with a list of settings that have been restored along with other details concerning the operation. The user then reviews the details and confirms the operation. The system proceeds to reload each of the modified components.
- Use 2. The system detects that an automated Restore Settings procedure has been requested. The system collects the list of available Restore Settings backups and selects the one based on the previously provided settings, that also indicate which elements and configurations to be restored. The system starts the procedure. When done, it shows the user a notification indicating a resume of the operation. It reloads the components restored and automatically restarts the system if the settings restored affected one of the critical components of the system.
- Use 3. The user starts the use case by going to the Restore Settings menu through the settings screen. The user then asks the system for a list of Restore Settings backups. The system provides the user with a list of Restore Settings backups along with the methods used and the settings that have been stored in each of those. The user then selects the desired backup and the settings to be restored along it and goes through a double confirmation procedure to start the task. The system then proceeds to restore the backup settings. The system encounters an error while restoring the settings and proceeds to roll back the changes, and after to inform the user of it. The system offers again a list of the available restore system points, to which the user provides the adequate settings. The system then proceeds to restore the settings. Once completed, the system provides the user with a list of settings that have been restored along with other details concerning the operation. The user then reviews the details and confirms the operation. The system proceeds to reload each of the modified components.
- Use 4. The system detects that an automated Restore Settings procedure has been requested. The system collects the list of available Restore Settings backups and selects the one based on the previously provided settings, that also indicate which elements and configurations to be restored. The system starts the procedure. The system encounters an error within the procedure, so it proceeds to roll back the changes and initiate a Reset Settings procedure.

History: 2013
Modified: 2013

4.2.13 Access Plugin

Main Actor: User
Precondition: The system has been initialised and the component has been installed.
Trigger: The user wants to start an installed plugin component.
Main Scenario:

- 1. The user goes into the main plugin management screen and selects the desired plugin.
- 2. The system checks for the correctness of the plugin component and its license to be valid.
- 3. The system starts the selected plugin component.
- 4. The component initialised itself and shows its main screen.

Alternative Cases:

- 1a. The system starts the use case invoked by another use case or because of it being planned.
- 2a. The system can't find the installed plugin. The system shows the user a screen to search for the plugin on an online environment, starts use case Access Appstore.
- 2b. The system detects that the plugin installation is corrupt. The plugin then checks for it online and if found starts the Delete Installed Component Use Case followed by the Install Component Use Case.
- 2c. The system detects that the license is invalid. The system then forwards the user to a purchase screen by starting the Purchase Component Use Case.
- 3a. The system detects that the component needs to be initialised for a first time, therefore starts in on the corresponding mode and allows the component to be configured before continuing with step 4.
- 3b/4a. The system can't start the component or a critical failure occurs. The system then notifies the user and starts the use case Reset Settings for this component.

Satisfaction of the Client: 1
Dissatisfaction of the Client: 5
Dependencies: Access Appstore, Delete Installed Component, Install Component, Purchase Component
Conflicts: —
Additional Notes: —
Author: Josep
Validation Scenarios:

- Use 1. The user goes into the plugin management screen where all the plugins are located, and looks for the desired plugin. Upon finding it the user requests the system to start the plugin. The system does plugin checks against the plugin structure to check for the correctness of it. Afterwards, it checks the component license to check for the validity of the user request. Following this step, the system proceeds to start the plugin execution and checks on whether it needs to be initialised for the first time. The component then starts itself and does basic initialisation procedures, not first-time ones, and proceeds to show its main screen with which the user can interact.
- Use 2. The system detects that an Access Plugin procedure was planned. The system does plugin checks against the plugin structure to check for the correctness of it. Afterwards, it checks the component license to check for the validity of the user request. Following this step, the system proceeds to start the plugin execution and checks on whether it needs to be initialised for the first time. The component then starts itself and does basic initialisation procedures, not first-time ones, and proceeds to show its main screen with which the user can interact.
- Use 3. The user goes into the plugin management screen where all the plugins are located, and looks for the desired plugin. As the system can't find the desired plugin, the system starts the case Access Appstore. After successfully completing the case, the system processes a request to start the plugin. The system does plugin checks against the plugin structure to check for the correctness of it. Afterwards, it checks the component license to check for the validity of the user request. Following this step, the system proceeds to start the plugin execution and checks on whether it needs to be initialised for the first time. The component then starts itself and does basic initialisation procedures, not first-time ones, and proceeds to show its main screen with which the user can interact.

- Use 4. The user goes into the plugin management screen where all the plugins are located, and looks for the desired plugin. Upon finding it the user requests the system to start the plugin. The system does plugin checks against the plugin structure to check for the correctness of it and it detects errors. Following this it automatically starts a reinstallation procedure by starting the Delete Installed Component Use Case and upon completion of it the Install Component Use Case. Afterwards, it checks the component license to check for the validity of the user request. Following this step, the system proceeds to start the plugin execution and checks on whether it needs to be initialised for the first time. The component then starts itself and does basic initialisation procedures, not first-time ones, and proceeds to show its main screen with which the user can interact.
- Use 5. The user goes into the plugin management screen where all the plugins are located, and looks for the desired plugin. Upon finding it the user requests the system to start the plugin. The system does plugin checks against the plugin structure to check for the correctness of it. Afterwards, it checks the component license to check for the validity of the user request, which fails. When the system detects the fail, the system starts the use case Purchase Component. Following this step, the system proceeds to start the plugin execution and checks on whether it needs to be initialised for the first time. The component then starts itself and does basic initialisation procedures, not first-time ones, and proceeds to show its main screen with which the user can interact.
- Use 6. The user goes into the plugin management screen where all the plugins are located, and looks for the desired plugin. Upon finding it the user requests the system to start the plugin. The system does plugin checks against the plugin structure to check for the correctness of it. Afterwards, it checks the component license to check for the validity of the user request. Following this step, the system proceeds to start the plugin execution and checks on whether it needs to be initialised for the first time. The system detects that the component needs to do a first-time initialisation so it provides the user with the necessary configuration screens from the component. After this, the component then starts itself and does basic initialisation procedures, not first-time ones, and proceeds to show its main screen with which the user can interact.
- Use 7. The user goes into the plugin management screen where all the plugins are located, and looks for the desired plugin. Upon finding it the user requests the system to start the plugin. The system does plugin checks against the plugin structure to check for the correctness of it. Afterwards, it checks the component license to check for the validity of the user request. Following this step, the system proceeds to start the plugin execution and checks on whether it needs to be initialised for the first time. The component then starts itself and does basic initialisation procedures, not first-time ones, but fails at starting. Due to this problem, the system then proceeds to start an automatic Reset Settings use case for this component. After it, the system starts again the plugin and detects that the component needs to do a first-time initialisation so it provides the user with the necessary configuration screens from the component. After this, the component then starts itself and does basic initialisation procedures, not first-time ones, and proceeds to show its main screen with which the user can interact.

History: 2013
Modified: 2013

4.2.14 Select Component

Main Actor: User
Precondition: The system has been initialised and the corresponding component is installed.
Trigger: The user wants to select or enable the component.
Main Scenario:

- 1. The user goes to the Select Component menu in the settings screen.
- 2. The system provides the user with a list of the installed components.
- 3. The user searches for and selects the desired component.
- 4. The system checks for the correctness of the desired component component and the validity of its license. Afterwards, the system provides the user with what the end result will be like with the selected component.

- 5. The user goes through a double confirmation screen to change the component.
- 6. The system starts the selected component and asks for it to configure itself using the already selected (other) components.
- 7. The component configures itself automatically.
- 8. The system provides the user with a changed component.

Alternative Cases:

- 1a. The system may detect that a Change Component use case is requested and therefore initiate the use case.
- 2a/3a/5a. In case of an automated Change Component use case, these steps will be omitted.
- 3a. The system can't find the desired component and starts the use case Access Appstore.
- 4a. The system detects that the selected component has an invalid structure. Therefore, starts the use case Delete Installed Component and afterwards the Install Component use case.
- 4b. The system detects that the selected component has an invalid license. Therefore, starts the use case Purchase Component.
- 4c. The system can't display a preview with the selected component because it doesn't support it.
- 6a. The system can't start the selected component, so it goes through the use cases Delete Installed Component and then Install Component.
- 7a. The component needs additional configuration by the user, so the system provides the user with a configuration screen for it.
- 7b. The component was previously run and has already the configuration, so the component loads it.

Satisfaction of the Client: 1
Dissatisfaction of the Client: 5
Dependencies: —
Conflicts: —
Additional Notes: —
Author: Josep
Validation Scenarios:

- Use 1. The user accesses the Select Component menu through the Settings screen. Afterwards the user gets a list of the already installed components from the system, from which it searches for the desired component. The user selects the desired component, to which the system performs integrity checks and also checks that its license is valid. The user reviews the result with the selected component and goes through a double-confirmation procedure to accept for the change. The system proceeds to change the component and prepares the component to configure itself. The component does automatic configuration steps and when ready the system applies it. Lastly the system presents the user with the updated component.
- Use 2. The system detects that a change component procedure has been requested. The system performs integrity checks on the requested component and also checks that its license is valid. The system proceeds to change the component and prepares the component to configure itself. The component does automatic configuration steps and when ready the system applies it. Lastly the system presents the user with the updated component.
- Use 3. The user accesses the Change Component menu through the Settings screen. Afterwards the user gets a list of the already installed components from the system, from which it searches for the desired component. The user selects the desired component but the system can't find it so starts the use case Access Appstore. After it, the system performs integrity checks to the component and also checks that its license is valid. The user reviews the result with the selected component and goes through a double-confirmation procedure to accept for the change. The system proceeds to change the component and prepares the component to configure itself. The component does automatic configuration steps and when ready the system applies it. Lastly the system presents the user with the updated component.
- Use 4. The user accesses the Select Component menu through the Settings screen. Afterwards the user gets a list of the already installed components from the system, from which

it searches for the desired component. The user selects the desired component, to which the system performs integrity checks but fails. The system starts the use case Delete Installed Component and afterwards Install Component, after which also checks that its license is valid. The user reviews the result with the selected component and goes through a double-confirmation procedure to accept for the change. The system proceeds to change the component and prepares the component to configure itself. The component does automatic configuration steps and when ready the system applies it. Lastly the system presents the user with the updated component.

- Use 5. The user accesses the Select Component menu through the Settings screen. Afterwards the user gets a list of the already installed components from the system, from which it searches for the desired component. The user selects the desired component, to which the system performs integrity checks and also checks that its license is valid but fails. The system then starts the use case Purchase Component. After it, the user reviews the result with the selected component and goes through a double-confirmation procedure to accept for the change. The system proceeds to change the component and prepares the component to configure itself. The component does automatic configuration steps and when ready the system applies it. Lastly the system presents the user with the updated component.
- Use 6. The user accesses the Select Component menu through the Settings screen. Afterwards the user gets a list of the already installed components from the system, from which it searches for the desired component. The user selects the desired component, to which the system performs integrity checks and also checks that its license is valid. The user can't review the result because the component doesn't allow for it, so goes through a double-confirmation procedure to accept for the change. The system proceeds to change the component and prepares the component to configure itself. The component does automatic configuration steps and when ready the system applies it. Lastly the system presents the user with the updated component.
- Use 7. The user accesses the Select Component menu through the Settings screen. Afterwards the user gets a list of the already installed components from the system, from which it searches for the desired component. The user selects the desired component, to which the system performs integrity checks and also checks that its license is valid. The user reviews the result with the selected component and goes through a double-confirmation procedure to accept for the change. The system proceeds to change the component but fails so starts the use case Delete Installed Component and after it the use case Install Component. Afterwards prepares the component to configure itself. The component does automatic configuration steps and when ready the system applies it. Lastly the system presents the user with the updated component.
- Use 8. The user accesses the Select Component menu through the Settings screen. Afterwards the user gets a list of the already installed components from the system, from which it searches for the desired component. The user selects the desired component, to which the system performs integrity checks and also checks that its license is valid. The user reviews the result with the selected component and goes through a double-confirmation procedure to accept for the change. The system proceeds to change the component and prepares the component to configure itself. The component does automatic configuration steps but detects that additional configuration is required so the system presents the user with a series of screen to provide for it. When ended the system applies the component. Lastly the system presents the user with the updated component.
- Use 9. The user accesses the Select Component menu through the Settings screen. Afterwards the user gets a list of the already installed components from the system, from which it searches for the desired component. The user selects the desired component, to which the system performs integrity checks and also checks that its license is valid. The user reviews the result with the selected component and goes through a double-confirmation procedure to accept for the change. The system proceeds to change the component and prepares the component to configure itself. The component detects that a previous configurations is available, so it loads it and when ready the system applies it. Lastly the system presents the user with the updated component.

History: 2013
Modified: 2013

4.2.15 Configure Component

Main Actor: User
Precondition: The system has been initialised and the component is installed.
Trigger: The user wants to configure a component.
Main Scenario:

- 1. The user goes to the Configure Component Menu in the Settings screen and selects the component that wants to configure.
- 2. The system searches for the component and checks for its correctness and the validity of the license.
- 3. The system queries the component for the settings screen, which provides to the user.
- 4. The user provides the settings configuration as indicated by the system and accepts the changes.
- 5. The system checks for the validity of the settings provided by the user.
- 6. The system applies the changes to the component and reloads it.

Alternative Cases:

- 1a. The component may decide to start this use case manually.
- 1b. The system will start this use case if necessary on the component's first-time usage.
- 2a. The system can't find the component and therefore starts the use case Access Appstore.
- 2b. The system detects that the component is corrupt and starts the use case Delete Installed Component and thereafter Install Component.
- 2c. The system detects that the component's license is invalid and starts the use case Perform Purchase.
- 3a. The component fails to provide the settings screen, therefore the system proceeds to notify the user of the situation and start the use case Reset Settings or Perform Diagnosis for the component, depending on the user input.
- 5a. The system detects that the settings provided by the user are invalid and therefore returns to step 4.
- 6a. The system detects that the component is a critical system component and therefore prompts the user to restart the system in order for the changes to be applied. The system marks the component to have its settings restored on next start.

Satisfaction of the Client: 1
Dissatisfaction of the Client: 5
Dependencies: —
Conflicts: —
Additional Notes: —
Author: Josep
Validation Scenarios:

- Use 1. The user accesses the Configure Component menu from the Settings screen. On it, it searches for the desired component, which the system finds, and asks the component to provide the configuration screen. The system checks for the correctness of the component and the validity of its license, both being satisfactory. Afterwards, it prompts the component for the configuration screen which provides swiftly to the user. The user then fills in the configuration screen and accepts the modifications. The system checks for the validity of the changes and succeeds, therefore modifying the component settings and reloading it.
- Use 2. The component requests the system to provide the user with the component's configuration menu. The system prompts the component for the configuration screen which provides swiftly to the user. The user then fills in the configuration screen and accepts the modifications. The system checks for the validity of the changes and succeeds, therefore modifying the component settings and reloading it.
- Use 3. The system detects that this case is initiated by the Access Component use case. The system prompts the component for the configuration screen which provides swiftly to the user. The user then fills in the configuration screen and accepts the modifications. The system checks for the validity of the changes and succeeds, therefore modifying the component settings and reloading it.

- Use 4. The user accesses the Configure Component menu from the Settings screen. On it, it searches for the desired component, which the system can't find. Therefore, the system starts the use case Access AppStore. After it, the system asks the component to provide the configuration screen. The system checks for the correctness of the component and the validity of its license, both being satisfactory. Afterwards, it prompts the component for the configuration screen which provides swiftly to the user. The user then fills in the configuration screen and accepts the modifications. The system checks for the validity of the changes and succeeds, therefore modifying the component settings and reloading it.

- Use 5. The user accesses the Configure Component menu from the Settings screen. On it, it searches for the desired component, which the system finds, and asks the component to provide the configuration screen. The system checks for the correctness of the component and fails. The system starts the use case Delete Installed Component and Install Component. Then it checks validity of the component's license, both being satisfactory. Afterwards, it prompts the component for the configuration screen which provides swiftly to the user. The user then fills in the configuration screen and accepts the modifications. The system checks for the validity of the changes and succeeds, therefore modifying the component settings and reloading it.

- Use 6. The user accesses the Configure Component menu from the Settings screen. On it, it searches for the desired component, which the system finds, and asks the component to provide the configuration screen. The system checks for the correctness of the component and the validity of its license but fails on this last. The system then starts the use case Perform Purchase. Afterwards, it prompts the component for the configuration screen which provides swiftly to the user. The user then fills in the configuration screen and accepts the modifications. The system checks for the validity of the changes and succeeds, therefore modifying the component settings and reloading it.

- Use 7. The user accesses the Configure Component menu from the Settings screen. On it, it searches for the desired component, which the system finds, and asks the component to provide the configuration screen. The system checks for the correctness of the component and the validity of its license, both being satisfactory. Afterwards, it prompts the component for the configuration screen but this last notifies that it can't due to an internal error. Te system then notifies the user of the problem and prompts the user to start the use case Reset Settings or Perform Diagnosis for the component. After, the system successfully provides the configuration screen to the user. The user then fills in the configuration screen and accepts the modifications. The system checks for the validity of the changes and succeeds, therefore modifying the component settings and reloading it.

- Use 8. The user accesses the Configure Component menu from the Settings screen. On it, it searches for the desired component, which the system finds, and asks the component to provide the configuration screen. The system checks for the correctness of the component and the validity of its license, both being satisfactory. Afterwards, it prompts the component for the configuration screen which provides swiftly to the user. The user then fills in the configuration screen and accepts the modifications. The system checks for the validity of the changes and fails therefore prompting the user again for the settings. The user inputs this time correctly the settings and the system succeeds on checking them, therefore modifying the component settings and reloading it.

- Use 9. The user accesses the Configure Component menu from the Settings screen. On it, it searches for the desired component, which the system finds, and asks the component to provide the configuration screen. The system checks for the correctness of the component and the validity of its license, both being satisfactory. Afterwards, it prompts the component for the configuration screen which provides swiftly to the user. The user then fills in the configuration screen and accepts the modifications. The system checks for the validity of the changes and succeeds, but realises that it is a critical system component. The system informs the user of the situation and programs a settings change for when the system is restarted.

History: 2013
Modified: 2013

4.2.16 Delete Installed Component

Main Actor: User
Precondition: The system has been initialised and the component is installed.
Trigger: The user wants to delete a installed component.
Main Scenario:

- 1. The user accesses the Manage Installed Components menu in the Settings screen.
- 2. The system provides the user with a list of installed components.
- 3. The user selects the appropriate component from the list and double confirms the deletion.
- 4. The system prompts the user on whether it wants to preserve the settings and/or the stored data of the component.
- 5. The user decides on which data wants to remove and accepts.
- 6. The system proceeds to remove the selected data.
- 7. The system informs the user of the success of the operation.

Alternative Cases:

- 1a. The system starts the use case from another use case, therefore the steps 2-5 will be omitted and only the component's main data will be removed.
- 3a. The system detects that the selected component is a system critical one, therefore doesn't allow the user to ask for its deletion. The use case ends on this step.
- 6a. The system detects that the component is in use and therefore ends it. If after that it still detects that there is some data in use, it indicates the user that a system restart is needed.
- 7a. If the step 6a is done, the system prompts the user to restart the system.

Satisfaction of the Client: 1
Dissatisfaction of the Client: 5
Dependencies: —
Conflicts: —
Additional Notes: —
Author: Josep
Validation Scenarios:

- Use 1. The user accesses the Manage Installed Components menu from the Settings screen. The system provides the user with a list of already installed components, from which the user selects the desired component. The user then confirms the component to be deleted twice. The system then asks the user on which data to keep, such as settings or other component-specific data. The user selects the data to be removed and confirm again. The system proceeds to remove the selected data and after that it informs the user on the success of the operation.
- Use 2. The system starts the use case prompted by another use case. The system proceeds to remove the main component's data and after that it informs the user on the success of the operation.
- Use 3. The user accesses the Manage Installed Components menu from the Settings screen. The system provides the user with a list of already installed components, from which the user selects the desired component. The user then confirms the component but the system informs the user that the component is listed as a system component and therefore can't be removed.
- Use 1. The user accesses the Manage Installed Components menu from the Settings screen. The system provides the user with a list of already installed components, from which the user selects the desired component. The user then confirms the component to be deleted twice. The system then asks the user on which data to keep, such as settings or other component-specific data. The user selects the data to be removed and confirm again. The system detects that the component is in use and terminates it, after it proceeds to remove the selected data. The system then detects that there is data that can't be removed and marks it to be deleted on restart, so it informs the user of the outcome of the operation and prompts it to restart the system in order to fully remove the remaining data.

History: 2013
Modified: 2013

4.2.17 Select Active Plugins

Main Actor: User
Precondition: The system is initialised.
Trigger: The user wants to select the active plugins.
Main Scenario:

- 1. The user accesses the Select Active Plugins menu in the Settings screen.
- 2. The system provides the user with a list of plugins already installed on the system and from these marking the ones that are currently active.
- 3. The user selects the plugins that would like to have available on the dashboard from the list and confirms.
- 4. The system checks that the selection is valid and for the allowance of the current configuration.
- 5. The system notifies the other components and the user of the change.

Alternative Cases:

- 1a. A component may request this case to be started.
- 2a. In case that the case has been started by a component request, that component will automatically be selected/unselected according to its petition.
- 4a. The system detects that the selection is not compatible with the current configuration. The system provides the user with either a Perform Diagnosis use case or to return to step 3.

Satisfaction of the Client: 1
Dissatisfaction of the Client: 5
Dependencies: —
Conflicts: —
Additional Notes: This use case differs from the Select Component one because on the former the plugins are just selected and not ran like on the later one.
Author: Josep
Validation Scenarios:

- Use 1. The user accesses the select Active Plugins menu on the Settings screen. Following this step, the system provides a list of the installed plugins marking the ones that are already active. The user then modifies the selection of active plugins and confirms. The system then queries the current configuration with the selection provided by the user and checks that there is no conflict. After that, the system notifies all the currently selected components of the change, and the user of the success of the operation.
- Use 2. A component asks for the user case to be held. Following this step, the system provides a list of the installed plugins marking the ones that are already active plus performing the action requested by the component on the list. The user then modifies the selection of active plugins and confirms. The system then queries the current configuration with the selection provided by the user and checks that there is no conflict. After that, the system notifies all the currently selected components of the change, and the user of the success of the operation.
- Use 3. The user accesses the select Active Plugins menu on the Settings screen. Following this step, the system provides a list of the installed plugins marking the ones that are already active. The user then modifies the selection of active plugins and confirms. The system then queries the current configuration with the selection provided by the user and checks that there is no conflict but fails. The system then provides the user with the option to modify the selection or to start the use case Perform Diagnosis. The user modifies the selection and this time the system checks successfully that there's no conflict. After that, the system notifies all the currently selected components of the change, and the user of the success of the operation.

History: 2013
Modified: 2013

4.2.18 Access AppStore

Main Actor: User
Precondition: The system has been initialised.
Trigger: The user wants to access the AppStore.
Main Scenario:

- 1. The user accesses the AppStore screen.
- 2. The system contacts the AppStore server with the already vinculated user account.
- 3. The AppStore server returns back a category-based list of recommended components, special offers and other information that it may deem of importance based on the user account.
- 4. The user then searches for a component that wants to review.
- 5. The system queries the AppStore server for the component and provides it to the user.
- 6. At this point, the user may decide to start the use case Perform Purchase.

Alternative Cases:

- 1a. The system detects that the use case execution has been requested.
- 2a. The system detects that the device is not vinculated with any user account. Therefore the system sends the AppStore server a list of the currently installed components.
- 2b. The system detects that it is the first time that the user accesses the AppStore, therefore it start the Vinculate Account use case.
- 2c. The server can't contact the AppStore server, therefore it asks the user to try again later. The use case ends here.
- 5a. The server can't find the component and notifies the system. The use case returns to step 4.

Satisfaction of the Client: 3
Dissatisfaction of the Client: 3
Dependencies: Vinculate Account, Perform Purchase
Conflicts: —
Additional Notes: —
Author: Josep
Validation Scenarios:

- Use 1. The user accesses the AppStore screen. The system then queries the AppStore providing the vinculated account for a recommended list of components and categories. The user then selects the component that it desires to view and the system queries again the AppStore server for the component's information. The user may then start the use case Perform Purchase.
- Use 2. The system detects that a use case start has been requested. The system then queries the AppStore for a recommended list of components and categories. The user then selects the component that it desires to view and the system queries again the AppStore server for the component's information. The user may then start the use case Perform Purchase.
- Use 3. The user accesses the AppStore screen. The system detects that no user account is vinculated, and therefore queries the AppStore for a recommended list of components and categories by sending the currently installed components. The user then selects the component that it desires to view and the system queries again the AppStore server for the component's information. The user may then start the use case Perform Purchase.
- Use 4. The user accesses the AppStore screen. The system also detects that it is the first time that the user accesses the AppStore, and therefore starts the use case Vinculate Account. The system then queries the AppStore providing the vinculated account for a recommended list of components and categories. The user then selects the component that it desires to view and the system queries again the AppStore server for the component's information. The user may then start the use case Perform Purchase.
- Use 5. The user accesses the AppStore screen. The system then queries the AppStore providing the vinculated account for a recommended list of components and categories. The user then selects the component that it desires to view and the system queries again the AppStore server for the component's information. The server notifies the system that no

component could be found and and the later notifies also the user. The user then searches for another component and the server finds it, so the system provides the user with the information for this component. The user may then start the use case Perform Purchase.

History: 2013
Modified: 2013

4.2.19 Perform Purchase

Main Actor: User
Precondition: The system has been initialised and an account has been vinculated.
Trigger: The user wants to purchase a component.
Main Scenario:

- 1. The user accesses the Component Purchase screen from the Component Information screen provided by the AppStore.
- 2. The AppStore server provides the user with an overview of what kind of information will the Component use.
- 3. The user accepts the conditions of the information the Component will share.
- 4. The AppStore server provides the user with an overview of the price that it will be paying and a list of payments methods, including the ones saved by the user.
- 5. The user accepts the conditions and selects a payment method and inputs the corresponding data.
- 6. The AppStore server proceeds to process the selected options.
- 7. The AppStore informs the user when the operation has been processed with the outcome.
- 8. The user reviews the information and accepts.
- 9. The AppStore notifies the system of the user of a new component available, and this latter starts the use case Install Component.

Alternative Cases:

- 1a. The system detects that a Perfom Purchase method has been requested.
- 2a. The AppStore server detects that the user is not eligible for the components installation, and therefore informs the user of the situation. The Appstore provides a set of alternative Components that are similar to the requested one.
- 3a. The user unchecks some optional features from the component, so that certain information is not accessed.
- 6a. The Appstore server detects that certain data is not correct and therefore informs the user, returning the use case to step 5.
- 7a. The Appstore server can't process the payment with the data provided by the user, therefore informing the user and returning to step 5.

Satisfaction of the Client: 1
Dissatisfaction of the Client: 5
Dependencies: Access Appstore, Vinculate Account
Conflicts: —
Additional Notes: A component may not be available for purchase on publisher's demand, therefore its justified to show the component information with the contact information of the publisher.
Author: Josep
Validation Scenarios:

- Use 1. The user accesses the Component Purchase screen from the Component Information screen after the use case Access Appstore. The Appstore server provides the user with the registered information regarding the desired component. The user then decides to proceed with the purchase. The Appstore then provides the user with a list of the available payment processing methods available on the user's country. The user selects the desired payment method and inputs the required data associated with it. The Appstore then processes the payment request using the provided information. After successfully checking the data and having proceed to ask for the money transfer, the Appstore communicates the user of the success of the operation. The user then reviews the data and accepts. The Appstore server

proceeds to notify the user's system of the outcome of the operation and a use case Install Component begins.

- Use 2. The system detects that a Perform Purchase use case has been requested. The Appstore server provides the user with the registered information regarding the desired component. The user then decides to proceed with the purchase. The Appstore then provides the user with a list of the available payment processing methods available on the user's country. The user selects the desired payment method and inputs the required data associated with it. The Appstore then processes the payment request using the provided information. After successfully checking the data and having proceed to ask for the money transfer, the Appstore communicates the user of the success of the operation. The user then reviews the data and accepts. The Appstore server proceeds to notify the user's system of the outcome of the operation and a use case Install Component begins.
- Use 3. The user accesses the Component Purchase screen from the Component Information screen after the use case Access Appstore. The Appstore server provides the user with the registered information regarding the desired component notifiying him that the selected component is not available for his region. The user then selects another component from the alternative component list provide dby the server decides to proceed with the purchase. The user reviews the information and proceeds with the purchase The Appstore then provides the user with a list of the available payment processing methods available on the user's country. The user selects the desired payment method and inputs the required data associated with it. The Appstore then processes the payment request using the provided information. After successfully checking the data and having proceed to ask for the money transfer, the Appstore communicates the user of the success of the operation. The user then reviews the data and accepts. The Appstore server proceeds to notify the user's system of the outcome of the operation and a use case Install Component begins.
- Use 4. The user accesses the Component Purchase screen from the Component Information screen after the use case Access Appstore. The Appstore server provides the user with the registered information regarding the desired component. The user then decides to proceed with the purchase but chooses to refuse certain conditions of use of the component, therefore disabling certain features. The Appstore then provides the user with a list of the available payment processing methods available on the user's country. The user selects the desired payment method and inputs the required data associated with it. The Appstore then processes the payment request using the provided information. After successfully checking the data and having proceed to ask for the money transfer, the Appstore communicates the user of the success of the operation. The user then reviews the data and accepts. The Appstore server proceeds to notify the user's system of the outcome of the operation and a use case Install Component begins.
- Use 5. The user accesses the Component Purchase screen from the Component Information screen after the use case Access Appstore. The Appstore server provides the user with the registered information regarding the desired component. The user then decides to proceed with the purchase. The Appstore then provides the user with a list of the available payment processing methods available on the user's country. The user selects the desired payment method and inputs the required data associated with it. The Appstore then detects that certain inputted data is incorrect and informs the user. The user inputs the data again, this time correctly. The Appstore proceeds to process the payment. After successfully checking the data and having proceed to ask for the money transfer, the Appstore communicates the user of the success of the operation. The user then reviews the data and accepts. The Appstore server proceeds to notify the user's system of the outcome of the operation and a use case Install Component begins.
- Use 6. The user accesses the Component Purchase screen from the Component Information screen after the use case Access Appstore. The Appstore server provides the user with the registered information regarding the desired component. The user then decides to proceed with the purchase. The Appstore then provides the user with a list of the available payment processing methods available on the user's country. The user selects the desired payment method and inputs the required data associated with it. The Appstore then processes the payment request using the provided information. After successfully checking the data the Appstore server asks for the money transfer but an external error occurs. Therefore the Appstore system informs the user of it and offers the user the possibility to change the

method or trying again. The user selects to retry and this time the Appstore server successfully complies with the request, thus the Appstore communicates the user of the success of the operation. The user then reviews the data and accepts. The Appstore server proceeds to notify the user's system of the outcome of the operation and a use case Install Component begins.

History: 2013
Modified: 2013

4.2.20 Vinculate Account

Main Actor: User
Precondition: —
Trigger: The system requires the user to vinculate itself with the AppStore server.
Main Scenario:

- 1. The user goes to the Account screen in the settings menu.
- 2. The system checks that the user has not a vinculated account on the server.
- 3. The system provides the user with relevant information regarding the account vinculation, including the terms and conditions.
- 4. The user accepts the base conditions for account vinculation.
- 5. The system asks the user with the relevant data in order to create an account.
- 6. The user provides the system with the relevant data and proceeds.
- 7. The system contacts the Registration server and creates the account. Thereafter, informs the user of the result.
- 8. The user reviews the result and accepts.

Alternative Cases:

- 1a. The system detects that a Vinculate Account use case has been requested.
- 2a. The system detects that an account is already vinculated. Therefore, the system shows the user the registered information, the user reviews it and ends the case.
- 2b. The system detects that an account is already vincualted. The user decides to edit the information and continues with the use case on step 5 using prefilled data.
- 2c. The system can't establish a connection with the Registration server, therefore notifying the user and offering the possibility to start a Diagnosis Use Case.
- 6a. The user provides the system with incorrect data. The system then prompts the user to correct the mistakes and the use case returns to step 5.
- 7a. The server detects that the inputed data contains payment information. Therefore, proceeds to validate it. In case the payment information is incorrect the server notifies the system and returns to step 5.

Satisfaction of the Client: 1
Dissatisfaction of the Client: 5
Dependencies: —
Conflicts: —
Additional Notes: This use case may be started without account vinculation.
Author: Josep
Validation Scenarios:

- Use 1. The user goes into the account menu in the settings screen. The system then successfully contacts the Registration server and observes that no account is currently vinculated, thus informing the user. The user then proceeds to vinculate the account by accepting the terms and conditions shown by the system. The system then prompts the user for extra information regarding its personal data which the user inputs. Afterwards, the user inputs the data and confirms. After this step, the system validates the information inputted by the user and proceeds to contact the server in order to create the account. The server double checks the provided information and proceeds to create an account. After successfully creating one, the server contacts the system in order to notify it of the success of the operation.

- Use 2. The system detects that a Vinculate Account use case has been requested. The user then proceeds to vinculate the account by accepting the terms and conditions shown by the system. The system then prompts the user for extra information regarding its personal data which the user inputs. Afterwards, the user inputs the data and confirms. After this step, the system validates the information inputted by the user and proceeds to contact the server in order to create the account. The server double checks the provided information and proceeds to create an account. After successfully creating one, the server contacts the system in order to notify it of the success of the operation.
- Use 3. The user goes into the account menu in the settings screen. The system then successfully contacts the Registration server and observes that an account is already vinculated. The user reviews the information and ends the use case.
- Use 4. The user goes into the account menu in the settings screen. The system then successfully contacts the Registration server and observes that an account is already vinculated. The user reviews the information and decides to change some of the present information. The system then prompts the user for information regarding its personal data, which is prefilled with the user's actual data, that the user inputs. Afterwards, the user inputs the data and confirms. After this step, the system validates the information inputted by the user and proceeds to contact the server in order to create the account. The server double checks the provided information and proceeds to create an account. After successfully creating one, the server contacts the system in order to notify it of the success of the operation.
- Use 5. The user goes into the account menu in the settings screen. The system then contacts the Registration server but detects that it is not possible to establish a connection. The system informs the user of the problem and offers the user the possibility to start a Diagnosis Use Case.
- Use 6. The user goes into the account menu in the settings screen. The system then successfully contacts the Registration server and observes that no account is currently vinculated, thus informing the user. The user then proceeds to vinculate the account by accepting the terms and conditions shown by the system. The system then prompts the user for extra information regarding its personal data which the user inputs. Afterwards, the user inputs the data and confirms. After this step, the system validates the information inputted by the user but detects an error. Therefore prompts the user to solve the detected issues. The user corrects the data and submits again, to which point the system successfully validates it and proceeds to contact the server in order to create the account. The server double checks the provided information and proceeds to create an account. After successfully creating one, the server contacts the system in order to notify it of the success of the operation.
- Use 7. The user goes into the account menu in the settings screen. The system then successfully contacts the Registration server and observes that no account is currently vinculated, thus informing the user. The user then proceeds to vinculate the account by accepting the terms and conditions shown by the system. The system then prompts the user for extra information regarding its personal data which the user inputs. Afterwards, the user inputs the data and confirms. After this step, the system validates the information inputted by the user and proceeds to contact the server in order to create the account. The server double checks the provided information and detects that there is payment data present. Therefore proceeds to validate the data, but detects that payment data is incorrect or invalid. It proceeds to notify the system, which in turn notifies the user. The user then inputs the correct data and the system communicates it to the server. The server successfully checks all the data and proceeds to create an account. After successfully creating one, the server contacts the system in order to notify it of the success of the operation.

History: 2013
Modified: 2013

4.2.21 Select Dashboard Plugins

Main Actor: User
Precondition: The system has been initialised.
Trigger: The user wants to select which plugins are visible on his dashboard.
Main Scenario:

- 1. The user goes to the Plugin Management menu in the Settings screen.
- 2. The system provides the user with a list of the plugins currently in use, as well as the plugins that are embedded into the system and can't be changed.
- 3. The user then chooses to change the currently selected plugins.
- 4. The system then provides the user a view with each of the installed plugins general features, as well as the approximate resource consumption they may take and the effect they may take on the system.
- 5. The user then chooses from the list the plugins he would like to have available on the system and confirms.
- 6. The system checks for incompatibilities and proceeds to do the changes. Afterwards, presents the user with an informative screen.
- 7. The user accepts the review.

Alternative Cases:

- 2a. The system detects that the user has permissions to modify the system's behavior. Therefore, prompts the user on whether it wants to modify its personal settings or the whole of the system.
- 4a. The system has no extra plugins installed. Therefore the system informs the user and prompts on whether it want to start the use case Access AppStore.
- 6a. The system detects that the user has chosen some incompatible plugin combinations. Therefore, it informs the user of the problem and returns to step 5.
- 6b. The system detects that there is a problem while disabling or enabling certain plugin. It informs the user and proceeds.
- 6c. The system detects that the system needs to be restarted after enabling a plugin, therefore it prompts the user on step 7.

Satisfaction of the Client: 1
Dissatisfaction of the Client: 5
Dependencies: —
Conflicts: —
Additional Notes: —
Author: Josep
Validation Scenarios:

- Use 1. The user goes to the Select Dashboard Plugins screen in the Settings menu. The system then provides the list of the currently active plugins for the user. The user then selects to change the configuration and the system proceeds to show him a list of the currently installed plugins as well as the plugins that are already installed on the system. It marks as well the plugins that can't be disabled or enabled depending on their functionality. The user then selects the plugins he desires to be enabled and afterwards he confirms the selection. The system then processes the selection and checks for incompatibilities. On the point of not finding any, it proceeds to apply the new configuration. After applying the necessary changes, it shows the user a briefing screen.
- Use 2. The user goes to the Select Dashboard Plugins screen in the Settings menu. The system detects that the user has permission to change the default settings of the system. The system then provides the list of the currently active plugins for the system. The user then selects to change the configuration and the system proceeds to show him a list of the currently installed plugins as well as the plugins that are already installed on the system. It marks as well the plugins that can't be disabled or enabled depending on their functionality. The user then selects the plugins he desires to be enabled and afterwards he confirms the selection. The system then processes the selection and checks for incompatibilities. On the point of not finding any, it proceeds to apply the new configuration. After applying the necessary changes, it shows the user a briefing screen.

- Use 3. The user goes to the Select Dashboard Plugins screen in the Settings menu. The system then provides the list of the currently active plugins for the user. The user then selects to change the configuration but the system detects that no extra plugins are installed, thus not being able to provide the list with a choice. The system then prompts the user to start the use case Access AppStore to install more plugins.
- Use 4. The user goes to the Select Dashboard Plugins screen in the Settings menu. The system then provides the list of the currently active plugins for the user. The user then selects to change the configuration and the system proceeds to show him a list of the currently installed plugins as well as the plugins that are already installed on the system. It marks as well the plugins that can't be disabled or enabled depending on their functionality. The user then selects the plugins he desires to be enabled and afterwards he confirms the selection. The system then processes the selection and checks for incompatibilities, therefore detecting them. The system notifies the user of the problem and prompts the user to solve the incompatibilities by either disabling one of the conflicting plugins or running a Diagnosis Use Case. After the user makes the request again the system proceeds and doesn't find any incompatibility, thus proceeding to apply the new configuration. After applying the necessary changes, it shows the user a briefing screen.
- Use 5. The user goes to the Select Dashboard Plugins screen in the Settings menu. The system then provides the list of the currently active plugins for the user. The user then selects to change the configuration and the system proceeds to show him a list of the currently installed plugins as well as the plugins that are already installed on the system. It marks as well the plugins that can't be disabled or enabled depending on their functionality. The user then selects the plugins he desires to be enabled and afterwards he confirms the selection. The system then processes the selection and checks for incompatibilities. On the point of not finding any, it proceeds to apply the new configuration but detects problems while applying it. After applying the necessary changes, it shows the user a briefing screen noting down the plugins whose status couldn't be modified.
- Use 6. The user goes to the Select Dashboard Plugins screen in the Settings menu. The system then provides the list of the currently active plugins for the user. The user then selects to change the configuration and the system proceeds to show him a list of the currently installed plugins as well as the plugins that are already installed on the system. It marks as well the plugins that can't be disabled or enabled depending on their functionality. The user then selects the plugins he desires to be enabled and afterwards he confirms the selection. The system then processes the selection and checks for incompatibilities. On the point of not finding any, it proceeds to apply the new configuration. After applying the necessary changes the system detects that it needs to restart itself in order to apply fully the new configuration, therefore it shows the user a briefing screen and prompts him to proceed to restart the system.

History: 2013
Modified: 2013

4.2.22 Add or Modify Recognised Device

Main Actor: User
Precondition: The system has been initialised and the user has the required privileges.
Trigger: The user wants to add a new recognised device.
Main Scenario:

- 1. The user accesses the Recognised Devices management screen from the Settings menu.
- 2. The system provides the user with a list of the currently enabled Recognised Devices and the newly detected devices, altogether with a briefing on the number of secondary devices.
- 3. The user then chooses to add a new Trusted Device.
- 4. The system provides a list of all the devices that are eligible to be added as a Recognised Device, as well as an option to force the detection of new devices.
- 5. The user selects a device from the list and indicates the level of trust.
- 6. The system then proceeds to add the device as a recognized device, therefore allowing the device a higher level of access if the user decided so.

- 7. The system informs the user of the outcome of the operation, and informs him that in case the device can't access with its new level of trustfulness, the device might need to be restarted.

Alternative Cases:

- 3a. The user clicks on one of the newly recognised devices, the system then prompts the user for the level of trust and continues with step 6.
- 3b. The user selects one of the already recognised devices, the system then prompts the user for the level of trust and continues with step 6.
- 5a1. The user wants to add manually a new device that's not available on the screen.
- 5a2. The system then prompts the user to manually introduce the device information.
- 5a3. The user introduces the information and accepts.
- 5a4. The system adds a new device to the list and proceeds to the step 6.
- 5b. The user selects a device that was previously blocked. The system then double confirms the user action and proceeds to step 6.
- 6a. The system can't add the device as a recognised device, therefore it prompts the user to start a diagnosis case or returning to step 4.

Satisfaction of the Client: 2
Dissatisfaction of the Client: 4
Dependencies: —
Conflicts: —
Additional Notes: By default all devices are allowed, thus this case might only be used to set a device as recognised by the user. This case can't be started by the system itself due to its criticality on the security of the system.
Author: Josep
Validation Scenarios:

- Use 1. The user selects the Trusted Devices management screen in the Settings menu. The system then displays a list of the devices whose role has not been established, such as a "guest device" or a "trusted device", as well as currently trusted devices. The user then clicks on the option to add a new device and the system shows the user a full list of the currently recognised devices. The user then selects one of the devices and chooses to add it as a recognised device. The system then provides the user with a list of the currently available levels of trust. The user chooses one of the levels of trust and accepts. The system then proceeds to add the device as a recognised device with the selected level of trust. Afterwards, the system shows the user with a briefing screen on the outcome of the operation.
- Use 2. The user selects the Trusted Devices management screen in the Settings menu. The system then displays a list of the devices whose role has not been established, such as a "guest device" or a "trusted device", as well as currently trusted devices. The user then clicks on one of the newly recognised devices. The system then provides the user with a list of the currently available levels of trust. The user chooses one of the levels of trust and accepts. The system then proceeds to add the device as a recognised device with the selected level of trust. Afterwards, the system shows the user with a briefing screen on the outcome of the operation.
- Use 3. The user selects the Trusted Devices management screen in the Settings menu. The system then displays a list of the devices whose role has not been established, such as a "guest device" or a "trusted device", as well as currently trusted devices. The user then clicks on one of the already recognised devices. The system then provides the user with a list of the currently available levels of trust for that device. The user chooses one of the levels of trust and accepts. The system then proceeds to add the device as a recognised device with the selected level of trust. Afterwards, the system shows the user with a briefing screen on the outcome of the operation.
- Use 4. The user selects the Trusted Devices management screen in the Settings menu. The system then displays a list of the devices whose role has not been established, such as a "guest device" or a "trusted device", as well as currently trusted devices. The user then clicks on the option to add a new device and the system shows the user a full list of the currently recognised devices. The user then realises that the desired device is not on the list and therefore clicks the option to add a new recognised device. The system then prompts

the user for the device information. After the user accepts, the system then provides the user with a list of the currently available levels of trust. The user chooses one of the levels of trust and accepts. The system then proceeds to add the device as a recognised device with the selected level of trust. Afterwards, the system shows the user with a briefing screen on the outcome of the operation and informs the user that the next time a device is shown with the parameters that were registered, it will be recognised with its corresponding level of trust.

- Use 5. The user selects the Trusted Devices management screen in the Settings menu. The system then displays a list of the devices whose role has not been established, such as a "guest device" or a "trusted device", as well as currently trusted devices. The user then clicks on the option to add a new device and the system shows the user a full list of the currently recognised devices. The user then selects one of the devices and chooses to add it as a recognised device. The system recognised the device as previously blocked and prompts the user for confirmation on the action. The system then provides the user with a list of the currently available levels of trust. The user chooses one of the levels of trust and accepts. The system then proceeds to add the device as a recognised device with the selected level of trust. Afterwards, the system shows the user with a briefing screen on the outcome of the operation.
- Use 6. The user selects the Trusted Devices management screen in the Settings menu. The system then displays a list of the devices whose role has not been established, such as a "guest device" or a "trusted device", as well as currently trusted devices. The user then clicks on the option to add a new device and the system shows the user a full list of the currently recognised devices. The user then selects one of the devices and chooses to add it as a recognised device. The system then provides the user with a list of the currently available levels of trust. The user chooses one of the levels of trust and accepts. The system then proceeds to add the device as a recognised device with the selected level of trust but fails. The system then informs the user of the situation and allows the user to select another device or to start a Perform Diagnosis Use Case.

History: 2013
Modified: 2013

4.2.23 Modify System Settings

Main Actor: User
Precondition: The system has been initialised and the user has the required privileges.
Trigger: The user wants to modify the system settings.
Main Scenario:

- 1. The user accesses the System Settings screen from the Settings menu.
- 2. The system then provides the user with a list of the currently available system settings.
- 3. The user then selects which setting he wants to be modified.
- 4. The system then provides the user with the currently available options for the selected setting.
- 5. The user then selects the option that desires for that setting and accepts.
- 6. The system then proceeds to apply the new configuration.
- 7. The system informs the user of the outcome of the operation.

Alternative Cases:

- 3a. The system detects that the selected setting can't be modified due to some factors, for example a system restart, and thus prompts the user to resolve this factors.
- 6a. The system detects that the desired option is not currently valid or applicable, therefore informing the user and returning to step 5.
- 6b. The system detects an error while trying to apply the changes, therefore informing the user and returning to step 5.
- 7a. The system detects that some components need to be reloaded, or the system itself needs to be restarted, after the operation and prompts the user to do so.

Satisfaction of the Client: 1
Dissatisfaction of the Client: 5
Dependencies: —
Conflicts: —
Additional Notes: —
Author: Josep
Validation Scenarios:

- Use 1. The user goes to the System Settings screen from the Settings menu. The system then provides the user with a list of the system currently selected system settings. The user then selects a setting to modify. The system then provides the user with a list of the available options for that setting, as well as custom input if that setting allows for it. The user then selects or inputs the appropriate data and accepts. The system then proceeds to apply the desired changes and, afterwards, informs the user of the outcome of the operation.
- Use 2. The user goes to the System Settings screen from the Settings menu. The system then provides the user with a list of the system currently selected system settings. The user then selects a setting to modify, but the system detects that the selected setting can't be modified due to some factors. The system then prompts the user to resolve this factors and offers the possibility to start a Perform Diagnosis Use Case.
- Use 3. The user goes to the System Settings screen from the Settings menu. The system then provides the user with a list of the system currently selected system settings. The user then selects a setting to modify. The system then provides the user with a list of the available options for that setting, as well as custom input if that setting allows for it. The user then selects or inputs the appropriate data and accepts, but the system detects that the selected data is not valid. Therefore the system prompts the user to correct the data and proceed. The user then inputs the data and the system successfully validates it. The system then proceeds to apply the desired changes and, afterwards, informs the user of the outcome of the operation.
- Use 4. The user goes to the System Settings screen from the Settings menu. The system then provides the user with a list of the system currently selected system settings. The user then selects a setting to modify. The system then provides the user with a list of the available options for that setting, as well as custom input if that setting allows for it. The user then selects or inputs the appropriate data and accepts. The system then proceeds to apply the desired changes but encounters an error while doing so. Therefore it informs the user of the problem and offers the user the possibility to choose again another system setting or start a Perform Diagnosis Use Case.
- Use 5. The user goes to the System Settings screen from the Settings menu. The system then provides the user with a list of the system currently selected system settings. The user then selects a setting to modify. The system then provides the user with a list of the available options for that setting, as well as custom input if that setting allows for it. The user then selects or inputs the appropriate data and accepts. The system then proceeds to apply the desired changes and detects that a partial component reload or a full system restart is required, therefore prompting the user to proceed with the required actions altogether with showing a briefing screen of the operation.

History: 2013
Modified: 2013

4.2.24 Delete Recognised Device

Main Actor: User
Precondition: The system has been initialised and the user has the required privileges.
Trigger: The user wants to delete a currently registered Recognised Device.
Main Scenario:

- 1. The user accesses the Recognised Device management screen from the Settings menu.
- 2. The system then provides the user with a list of currently registered devices, as well as other ones.
- 3. The user then selects a registered device.
- 4. The system provides the information of the trusted device.
- 5. The user then selects the option to delete the trusted device.
- 6. The system then asks the user for double confirmation on the action.
- 7. The user confirms the action and proceeds.
- 8. The system proceeds to perform the operation and, afterwards, provides the user with a briefing screen.

Alternative Cases:

- 6a. The system detects that the selected device is one of the initial devices, therefore informing the user of the impossibility to remove it as a trusted device and returning to step 2 or offering the possibility to start a Perform Diagnosis Use Case.
- 8a. The system detects an error while trying to perform the operation and prompts the user to start a Perform Diagnosis Use Case.

Satisfaction of the Client: 1
Dissatisfaction of the Client: 5
Dependencies: —
Conflicts: —
Additional Notes: —
Author: Josep
Validation Scenarios:

- Use 1. The user accesses the Recognised Devices menu in the Settings screen. The system then provides the user with a list of the available registered devices and other devices. The user then proceeds to select the desired device to be deleted. The system the provides the user with a briefing screen on that device, so that the user can correctly identify it, as well as the option to delete the device. The user then selects to delete the device, to which the system double confirms the action. After the user input, the system proceeds to delete the recognised device from the system and informs the user afterwards.
- Use 2. The user accesses the Recognised Devices menu in the Settings screen. The system then provides the user with a list of the available registered devices and other devices. The user then proceeds to select the desired device to be deleted. The system the provides the user with a briefing screen on that device, so that the user can correctly identify it, as well as the option to delete the device. The user then selects to delete the device, but the system detects that the device is one of the initial devices. The system informs the user of the impossibility to perform the action and offers him with a Perform Diagnosis Use Case.
- Use 3. The user accesses the Recognised Devices menu in the Settings screen. The system then provides the user with a list of the available registered devices and other devices. The user then proceeds to select the desired device to be deleted. The system the provides the user with a briefing screen on that device, so that the user can correctly identify it, as well as the option to delete the device. The user then selects to delete the device, to which the system double confirms the action. After the user input, the system proceeds to delete the recognised device from the system but detects an error while doing so. Therefore informs the user of the situation and offers him the ability to start a Perform Diagnosis Use Case.

History: 2013
Modified: 2013

4.3 Resume

In this chapter we have seen the Golden Topic of this thesis, describing the requisites any proposed solution must have in order to comply the goals of it and solve the proposed problems. We have obtained from the previous chapters the necessary knowledge to define and group the ones we considered most important for the objectives we have had presented. As a resume of the structure presented in this chapter, it resumes into a infotainment system highly resembling to the mobile world. This is due to the fact that the system has to provide support for Content Management capabilities, as stated on the goals of the thesis, and therefore we have based the requirements for this aspect on the already present solutions on the mobile world. This fact will influence on the development of the proposed solution, on the next chapter, on the conceptualisation of the architecture and the outcome of any prototype.

5 Solution

In this chapter we will explain the designed architecture, based on the requirements, goals and alternative solutions exposed in the previous chapters. Along it, we will also observe how the proposed requirements are fulfilled, along with the limitations of the design which will also be discussed in the next chapter. In order to achieve this, we will first show the designed architecture and then we will go down step by step on why each portion of it has been designed meanwhile we also explain them in detail.

5.1 Architecture

In this section we will analyse the hardware architecture that compromises the thesis, as well as the base technologies we are bound to use and how can we adapt and wrap our thesis around them.

5.1.1 Architecture Diagram

This is the architecture diagram that represents the general hardware set-up of the thesis along with the base technologies being used.

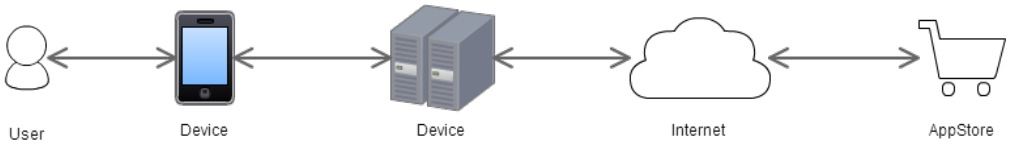


Figure 5.1: Architecture Diagram

5.1.2 Justification of the Architecture

Our architecture proposal is centred around a server-client architecture. That means, that apart from any of the infrastructure that might be developed by the manufacturer, or the RACE project in our case, our thesis will be providing for a totally independent server that will be on charge to communicate with the available services. This server will be the main gateway between our clients and any functionality that they might require, that is, access to the vehicle's features or even to internet will have to be allowed by it. By using this architecture we make sure to provide a solution to one of the major security concerns, client tampering. Due to the criticality of our system we can't allow that certain clients are affected by other clients on the network, and we need a device that itself plays the role of a moderator in order to avoid for any malfunction to be taken care of without the other clients ever noticing. Furthermore, this architecture allows for any extension on the system to be fully centralised, thus avoiding the necessity of any client changes or any manufacturer-specific enchantments. Further discussion on this topic is present on the chapter Discussion, where we will evaluate the different alternatives and the main reasons on why they were ruled out.

5.2 Hardware Architecture

In this section the selected Hardware Architecture will be explained. Although not a big part itself of the thesis, we have to take into account that it has greatly influenced the choice of components on the Software Architecture. It is important to be noted that none of these hardware considerations have been selected on the work and, instead, they have been imposed. Different alternatives and proposals will be taken into account into the Discussion chapter.

5.2.1 Hardware Architecture Diagram

Follows is the Hardware Architecture Diagram. This diagram is of extreme importance in order to understand the inner workings of the system architecture, and will play a key role in the definition of the Software Architecture in the next section.

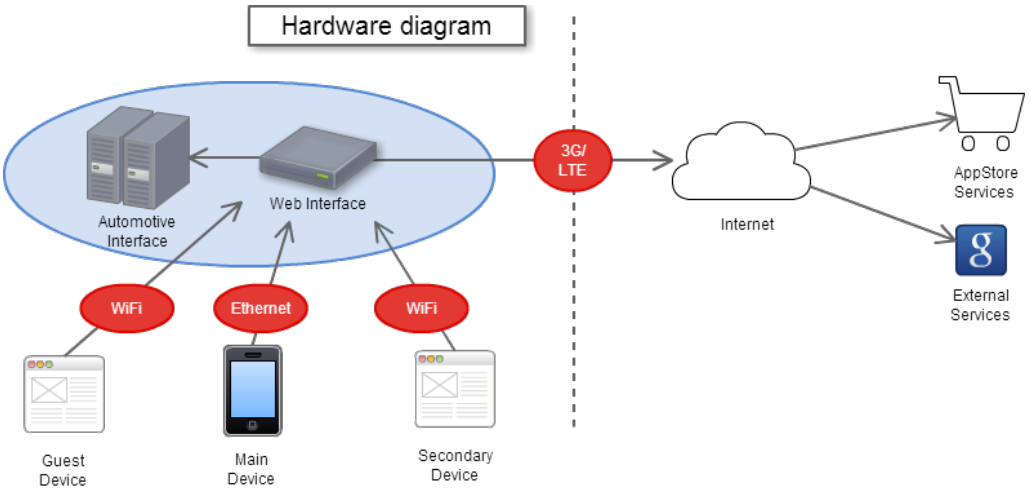


Figure 5.2: Hardware Architecture Diagram

5.2.2 Raspberry PI

Raspberry PI is a Single-board computer which is designed to be both extremely cheap and with very low power consumption rates. The RACE project will be actively using Raspberry PI as their main server to provide for the interface to the vehicle's features. To that extent, it is a work's requirement for our server to be established on the same device, therefore we have to take into account the following premises:

- The device has a 700 Mhz processor and 256 MB of RAM.
- The device is running a Linux-based operating system.

This two considerations will heavily impact into our software choice and furthermore are limiting our techniques of development and imposing heavily requisites. Concretely, the developed software will have to take into account the following parts mostly present in embedded devices projects.

RAM limitation This is, perhaps, the most serious problem that has been confronted up to date. Being a server as such and having to provide for services to the multiple clients might require a huge usage of system resources, specially if the main load is centred on the server. Therefore, an informal requirement is added to the solution regarding extremely-high efficiency and another regarding the capability to manage the used resources and to scale down the number of clients or the features being provided.

CPU limitation As a second-in-line requirement we have the added feature of low CPU consumption. Being in an infrastructure as ours it is not considered to have a system capable of adapting to fluctuating processing speeds, but of a system capable of managing its very own resources whilst providing for stable information transmission capabilities. As of such, our system also has to take into account that another system, the one developed by RACE, will be partly on the same system and as such it might take vital resources for our very own; therefore requiring the software to be able to adapt to this circumstances.

OS limitation The choice of the running operating system might have tampered our efforts to provide for the best State of the Art technology proposals. The collaboration with the RACE project bounds us to use a Linux operating system and therefore any of the complements that might be required in the software implementation has to work necessarily on this system. This is not a huge limitation in some aspects, because our system already has the requirement to be completely architecture-independent, but of course might pose some obstacles for our initial proposal.

5.2.3 Nexus VII Tablet

This device is going to represent a standard client system. This tablet uses State of the Art technology and software, therefore allowing us to practically not restrain the software development on the client on any means. Through our architecture choice we will be using a Web-based approach on the client, but this is not a problem either for this tablet as it is coupled with the best browser on terms of both JavaScript performance [Tea13] and HTML5-standards compliance [Lee13] according to the most prestigious tests available on internet, Google Chrome.

5.2.4 RACE Components

The RACE project will be providing certain manufacturer-specific components such as the WiFi identification procedures and the database entries indicating which devices should be considered as car-embedded devices (also called main devices along this whole document). Although this fact and that we have collaborated closely in the development of these procedures, those shall not be considered as part of this thesis itself. The fact is that the barrier between the Manufacturer and our System is clearly defined, as we can see in te previous Architecture Diagram, but in this case the components we just cited will be running also on the same Server we will be using, and therefore will be helping it along by providing the defined required server procedures (which are explained in this very same chapter).

5.3 Software Architecture

In this section we will explain the Software Architecture that we have selected, along with the main reason before this. In our choices the main objective was to satisfy the maximal number of requirements whilst keeping a delicate equilibrium between them. The technologies that have been chosen here can be justified based on the chapter Related Work.

5.3.1 Software Architecture Diagram

Following is a the Software Architecture Diagram depicting the planned approach for the our architecture. This diagram depicts the planned approach to the different subsections of the suggested solution for our work, altogether as explaining in a detailed way the designed interaction between the multiple components.

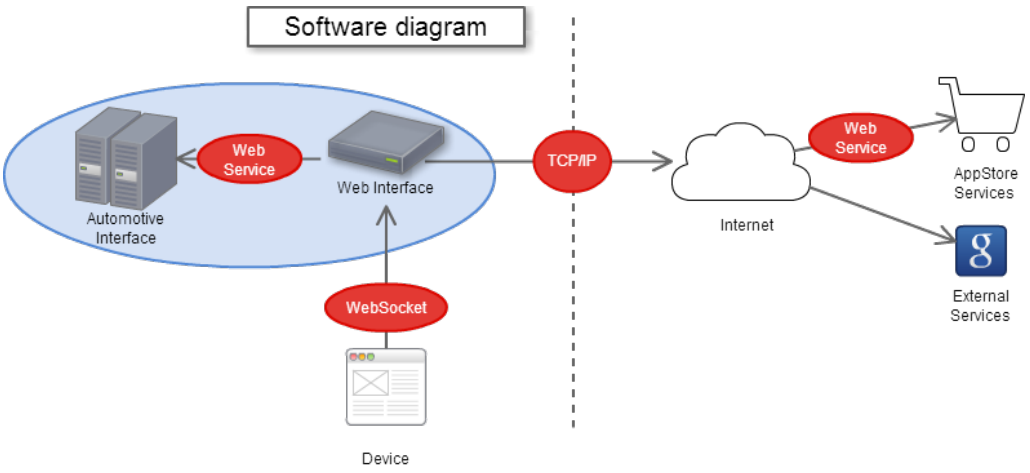


Figure 5.3: Software Architecture Diagram

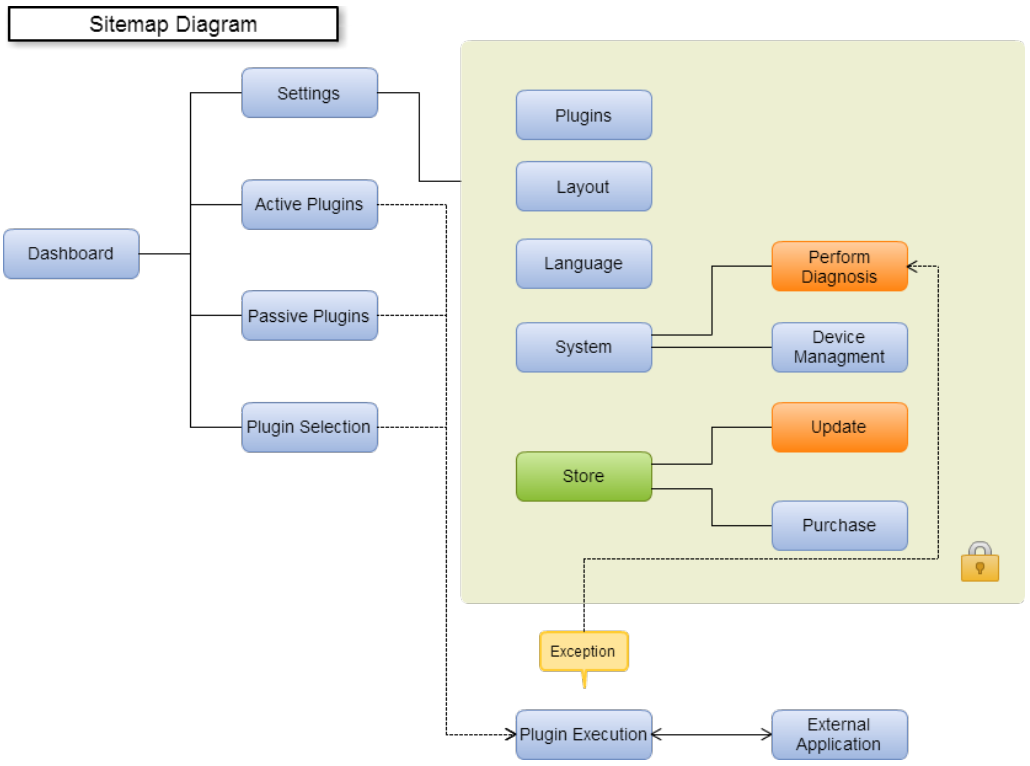


Figure 5.4: Sitemap Diagram

5.3.2 System

In this section we will be commenting on the different roles and features that our system will be exhibiting as well and analysing the different scenarios that it will manage. Apart from providing the basis technologies and features that are associated with a Web-Server with Web-Sockets communication, it'll be providing an additional set of functionalities specially designed for our system. As follows we will be explaining these features which combine both server- and client-side specifics. The system is also responsible to handle the trusted devices, and therefore corresponds with the "Add or Modify Recognised Device" and "Delete Recognised Device" use cases, a full list can be seen on the Technology Diagram.

Bootloader

Due to the modular design of our system, we are considering this special component of it. This component is the one responsible for the system update and recovery tasks, in order to comply with the relative requisites. It is the last line of defence against a total system failure and therefore it must comply with the maximum quality standards. Angular key of the whole system, it will need to check for the system integrity every time this last is started anew and provide the user for recovering activities, rolling back any possible undesired changes, and also providing for the ability to update the system. This component's features are heavily based on current Operating System bootloaders, and therefore offers a similar functionality set. With this component we aim to provide functionality of the use cases relating to device initialization and restoration, due to it being designed as standalone from the system.

System Update One of the star features of the bootloader is to replace the system contents with a new package thus upgrading the system. Whilst this procedure must comply with all the necessary security checks, it must also ensure to be quick and responsiveness by providing the user a maximum execution time and progress information. This system update procedure is also responsible for preserving the already existing settings, providing for the necessary changes in case they are.

Self Update The bootloader must be able to update itself and the underlying system, thus this component will be the one with a most low-level approach to the underlying system architecture. The incorporation of this feature is due to the consciousness that our system will be running on another infrastructure which may contain critical exploits that might affect our system at the end. Along with the general strategy to allow components and the system itself to be updated, this functionality specification goes along the "Check for Update" and "Perform Update" use cases.

Settings Reset The component must be able to provide for a settings roll-over in order to return the system to its very defaults in case of a situation that might require it. This feature is set to remove any local custom data stored by the user either on the system side, the modules side or both. Inside this feature we also include the ability to disable selected system modules that might be causing an unexpected behaviour. This functionality is paired with the use case "Reset Settings".

Integrity Checks The bootloader is the module responsible for the system's identification and load, therefore providing the loaded system with the appropriate environment to execute. To this extent, a swift but extensive verification role must be playing for it to ensure that the system or itself has not been modified by any external source and providing for a solution (to apply one the bootloader's features) to be applied.

System Rollback This feature is based on the already known "System Restore" that can be found on modern versions of the "Microsoft Windows" Operating System. It's main functionality is to allow the system to return to a previous working status. To this extent, the system needs to make "Snapshots" of its configuration every specified time while also preserving the different system versions. The idea behind this feature is that the different system and module versions will be preserved (up to a maximum amount) altogether with the different configurations of those.

Therefore, the system will be able to downgrade itself to a previous version and/or apply a previous configuration, thus allowing the user a easy way to solve any system problems. This feature is exactly coupled with the corresponding use case labelled “Restore System“.

Error Handling

The system is the maximum authority in the whole deployment scenario, and therefore it must control and enforce certain rules. For instance, the system must be able to detect when an error has occurred on itself or one of the running components and solve it. This affirmation is deemed to resolve any doubts regarding the stability of the system as a whole, therefore the system will handle when a critical error occurs to a component and try to find out a solution to it, restarting the component or rolling back its settings. This feature means that in the most extreme cases the system will detect a non-standard procedure on itself and take the appropriate measures to restore its normal behaviour (for example, by using the bootloader), without almost no user interaction and avoiding to a maximum extend getting the user to realise the problem. To comply with this objective we propose the use of different testing tools to ensure that a plugin has not itself got victim of its own bad programming. This feature is embedded in order to be able to fulfil one of the “Perform Diagnosis“ use case and the “Diagnostics“ requirement.

Performance Assurance

One of the key requirements of our system, in terms of user experience, is the fact that we ensure a maximum responsiveness time. This is required to avoid the user itself from overloading the system, as it is web-based, as well to avoid the user from any distractions. Therefore the system must be able to manage, to a maximum extent for the given technologies, each component resources usage and prioritize the ones that the user is using in detriment of others. For this objective, we have deemed to use components typically used on testing directly on the different components, to test their efficiency on the fly.

Component Update

The system must be able to provide updates for the components in an orderly and correct manner while also taking into account the different component requirements (that is, certification procedures). To this extent, the system must be able to itself update, install and remove the deemed components. On our solution this is based on a combination of factors. In first place, the system can be provided of a new component by providing the package to it via either internet or through a personal device. On second place, the system is able to check against a server on the internet for new updates.

Diagnosis

As a key requirement and itself being used in multiple Use Cases, we have taken into account the consideration that our solution could execute diagnostics into itself and therefore be able to propose active solutions to the user. The idea that we used has come from the Problem Handling procedures on “Microsoft Windows“ and most recently to the “Fix It“ innovations. To this extent, we propose availability of a list of common known problems and solutions to it, that can be actively enhanced by means of system updates or directly accessed on the internet. Furthermore, we aim onto the community-based support and therefore propose the user to submit his problem to a community website where experts could help him. Also, the idea revolves around the fact that if a similar problem has been detected and submitted previously, then the user would be redirected to this one. With this proposal we aim to replicate the success rate of Open-Source software by actively supporting the in-user problem solving. As a second part of the “Perform Diagnosis“ use case, due to different components being involved, this feature is also bound to it.

5.3.3 Components

In this section we will be explaining the proposed component architecture for this thesis. This architecture is set to solely represent the maximum aspiration on this thesis: providing sandboxing-like capabilities to a Web-based software. To this extent we will first analyse the proposed approach and also the main reasons on why it was selected, afterwards detecting the different type of components that have been selected for this work. This section is paired with the "Content" part of the Content Management System goal that has been defined for this thesis.

Component-based Approach

We have developed a custom sandboxing-based approach for our component architecture. This system is possible in Web-based environments because of the existence of the following factors. On the next section we will be exposing these features in a diagram.

Sandboxing The concept that we're tackling on this part is the "sandboxing" part that has been obtained from the use case analysis. The idea of sandboxing comes from the fact that by it's usage each component can be isolated from the others, with the "parent" components having a higher degree of privileges that that of the children, thus leading to the concept of layered architecture [AM11].

Messaging Queue In order to communicate between the different layers, the concept of "Message Queue" needs to be used. With this concept in mind each layer is able to communicate with the others by simply passing a "message" to the other layer, who can choose on whether to interpret it or not and identify it with security features in place therefore rendering the message harmless [TN00].

Centralised Communication API Our system will only provide one way to communicate with the exterior, via a centralised communication API. This means, all the traffic towards any external request will be held through the API and therefore will be able to be monitored and supervised in order to prevent unexpected or undeclared behaviour. This concept is understood under the hood of the Message Queue mechanism, as each layer needs to contact it's immediate superior in order to send a message.

Dynamic Feature Detection Our system has been designed in such way that the features exposed by the car interface will be automatically available on the different components regardless of their specification. That is, upon complying with the component's privilege requirements, that component will be able to execute the different methods if they are available.

Component Interaction Diagram

Next is the component interaction diagram, this element is key to understand the underlined process of interaction between the different elements in our system.

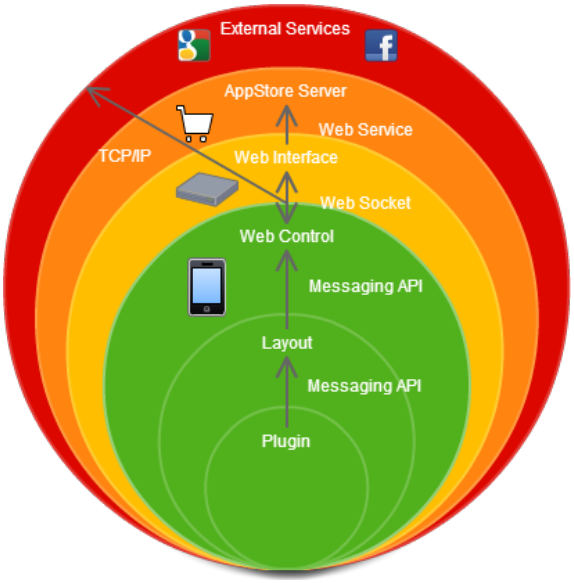


Figure 5.5: Component Interaction Diagram

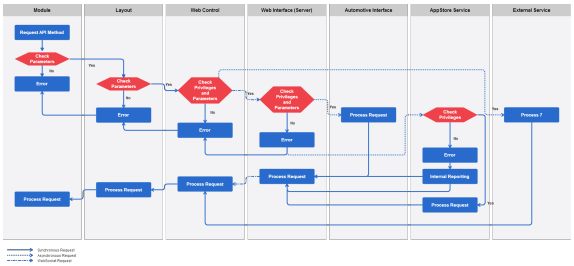


Figure 5.6: Component Request Process

Component Management

In this section we will be explaining the client-side layered conceptual architecture on our work. In concrete, we will expose the different roles accepted inside our system and the level of subordination of them. Each of the different component types will be exposed in the the following sections.

Root Layer This layer is the uppermost one and is represented itself by the base web site. That means, this "layer" itself is under the direct DOM structure and therefore has no boundaries apart from those imposed by the own browser. This layer will be in charge of the communication with the Server itself via the use of Web Sockets. Therefore will need to identify every requester and their credentials, along with providing them to the Server for further checks. Apart, will also be in charge to provide for the main Error Handling and fall-back, as well as managing the resources that the other components might be using. Accessing this layer should only be available through the Messaging API.

Layout Layer This is the first client-side layer, we could represent it as the actual Window Manager in an operating system environment. Therefore its main job resides on managing the location of the multiple elements as well as providing access to a common set of multiple features that

should be available to this layer. These features will be later explained in the section “Layouts“. In concrete, the Layout Layer has to provide for showing notifications to the user, informing the user of any problems that might occur and maintain a two-way communication with the Root Layer about any status from the multiple components.

Plugin Layer This is the lower layer, the so called “Component Layer“. In this layer we can find the main extension point for our application, the plugins. Components located in this layer are completely sandboxed and can only access external information via querying the superior layer (Layout Layer). The main idea of this layer is that any component running on it should be able to execute the vast majority of its own process without any need for external help. Also this layer should adapt itself to the styles provided by the superior layer in order to provide a smooth-looking interface along the whole system.

Component Trusting

A key security feature of our system, and also one of the most important requirements, has to do with the way a component is identified and trusted into our system. The final objective is to avoid piracy and illegitimate component installation, with its main focus on curving any possible exploits on our system and protecting also the user’s integrity and security. To this extent, we propose to allow each system to identify the planned installed software on their own, thus removing the need for the system to have an active connection for any component expansion or update.

Trusted A trusted level on a component means that this component has been recognised as being signed by a trusted authority, therefore assuring the system of the source of it. A trusted component might not be able to operate on the required role thought, with only a handful of authorities being allowed to sign the system components for instance; thus assuring a standard of quality for the system infrastructure.

Self-signed A self signed component, also called untrusted, is that component that represents a valid signing structure but that has not been trusted on the system. This doesn’t necessarily mean that the component is an evil one, but that the certificate authority that announces is not recognised by the system yet, or that is a too old version for the system and therefore a new version of the component should be provided. This components will be allowed to be installed under the sole user’s responsibility, but also have the possibility to be registered with the device in order to help the user find trusted versions of it.

Developing We can consider into this category those components product of the development, that have still not achieved a final stage or that are pending of validation by a competent authority. Due to the necessity of the developers to test their components on a wider scale community, this kind of components will be issued by a trusted authority a temporal signing allowing the component to be distributed to a group of general enthusiasts (but refraining from general-public release).

Plugins

The main extension component of our system, the so called “Plugin Components“. This components aim to solve the requisites of expansion, whilst also complying with the modular pattern. This pattern is basic for our system, in order to provide extensibility and flexibility points to our work which otherwise would lack and fall on the same level as of the existing systems. A Plugin can be considered as a whole entity on itself and shall compromise of its very own deployed package, with all the associated data that might require in order to operate, as well as external infrastructure that is not covered in this work, therefore rendering the plugin developer as the sole responsible for this as well as the security and coverage risks associated with it. This concept is coupled with the use cases regarding plugin selection and execution, due to the fact that we consider a plugin to be itself an application (as explained on the background chapter).

Plugin Execution State Any plugin must be able to be executed and therefore show a full interactive screen with the user. Apart from this base requirement, a plugin can have other execution states.

Suspended A suspended plugin is that which will be ran on the background on a very low-priority execution under time and resources limitation. A plugin has to register for this state as for a normal deployment no plugin will ever go into this status and instead be terminated. A plugin in Suspended status might also provide notifications to the user and therefore be started anytime.

Informative A plugin can be shown on the dashboard if the configuration and the plugin allow for it. In this state, the plugin will have a higher resource pool than in the Suspended status and will be able to provide direct feedback to the user. The informative status can have multiple display sizes, with the plugin deciding on which sizes to support.

Running A plugin in the running status will be providing the user with direct feedback information. That is, it will be occupying the central scenario and will have access to almost unlimited resource boundaries, except for those needed by the other superior layers.

Disabled A disabled plugin is that whose activity is null. A disabled plugin can be set to running state but will not be able to provide any information to the user or show any notifications to the user.

Plugin Definitions Inside the Plugin definition, though, we can find two implementation-able definitions.

Application Plugin The most logical outcome of defining a plugin is that which will be ran under the very own application, that is, in the Plugin Layer. This plugin will have a restricted set of features from the system in order to protect this last integrity and functionality. This plugin is the one that will be under the commercialization efforts and therefore will impact directly on the revenue and success of the designed system. Each plugin developer will be provided with a minimum API that he will be able to use in any deployed system, with extra features also available to the plugin developer based on the different systems. By default, only trusted plugins, these are the ones that come from trusted sources, will be able to be deployed. Though, the system must support for untrusted plugins, also called self-signed, to be deployed in order to provide for the plugin development.

System Module A system module is by definition a component with far higher privileges than a normal plugin. These modules are designed with the solely idea to help with system expansion by the manufacturers. This system has been designed with the key objective to avoid the need for any custom OEM modification, but we have taken into account the requirement to provide for some unique points to each manufacturer, to allow for a more brand-customised environment. These components require from previous development and test approval, due to the critically of them, and will be run in a server environment with the capability to modify the user experience on the client side too.

Layouts

This component is the one that mostly characterises the Layout Layer, as the very same name indicates. Whilst is the responsible for managing the subsequent plugins, it is inside the realm of the reality to have more than one doing this task on the same time. It's sole role as a Window Manager allows it to have direct contact with the underlying plugins and acts also as a bridge between them and the upper parts of the system, therefore having the capability to homogenise the behaviour of the system as a whole. As such, every Layout will provide a minimum set of common features

Notifications This feature encompasses the need to provide for constant feedback to the user. To achieve this goal, each layout needs to provide for a way to show information dynamically to the user, even when the plugins are not under active execution status. The layout has to control absolutely everything on how the notifications are displayed, though a good practise would be to notify itself to the superior layer for every notification, in order to take preventive measures if there is a plugin that's abusing the system. Apart from this, the Layout itself is also responsible to provide for any notifications center if it is decided as such. By allowing the control of the way the notifications are displayed, the location and the styling we allow for a full customisation of such feature.

Information Plugin Display Every layout can provide for the possibility to display one or multiple plugins in a informative status. That means, the plugins that claim to support the certain display will be eligible for this status. Previous resource consumption analysis, this will should allow for the user to obtain the maximum information without the need to set the plugin a running status. It is key in order to interpret this feature the knowledge regarding the different display sizes, which will be taken care of by the own Layout managing infrastructure. A plugin might advertise its availability to be run in a certain size as informative status, but the layout must proof that such a display is not going to affect the other plugin interfaces.

Plugin Selection As an imperative for any Layout, a plugin selection screen must be provided. This selection screen might be also coupled with a search feature, but nevertheless the layout must allow for any installed plugin eligible to be run to be executed. Therefore, the layout itself has no control of the executed plugin, instead it has to notify the superior layer of the user request for plugin execution.

Languages

A key requirement of any modern software nowadays is the need for Localisation. Such is the need, that non localised software tend to fail in the prospects of global growth in a totally dramatic manner. Therefore, and as empathised by one of the strong requirements proposed in the work analysis, we will be providing the system with a wide set of translations covering the major languages as well as the emerging ones, such as Catalan or Scottish. The main aim of this localisation service is therefore to also provide the different components with already localised translations in order to ease the global need for them to be translated. Even a partial localisation is better than nothing on the eyes of user experience. In order to simplify translation though, an service for submitting translation suggestions and corrections will be submitted, altogether with a public community-based service for translation such as Crowdin.

Styles

This component sole aim is to provide a continuous dynamic and innovative look to the system once deployed. Is a key feature for any system that the user sees the system as improving over time on both quality and effectiveness. To achieve this goal, we are proposing the ability for the system to change its appearance at user's will, with new styles being able to be obtained via internet. Styling is a key feature on Web Design also, and the possibilities that it nowadays provides makes it easy for our system to take a footnote on these thesis. In a turn to try to offer a homogeneous look, both layouts and plugins will be required to support at least basic styling based on the system settings. Apart from this, though, every component is capable of offering its very own custom themes apart from the system ones.

5.3.4 User Management

In this section we will be commenting on the different options selected for the user identification and management, altogether with the device identification and the user data storage. With the comprehensive set of selected features we aim to cover the major aspects that we can find nowadays in any modern system, as well as to provide mobility and flexibility to the user by allowing him to have as much of the same look and feel in any device that uses our system.

Profile-based Approach

In order to cover the user necessities in terms of data storage and settings sharing, we have turned ourselves into the same feature that modern Operating Systems have been promoting since their initial multi–user designs. That is, we propose to use the concept of "Profile" for every user or device. This concept allows us to provide for the much needed encapsulation and sand-boxing of the user data, as well as providing for the mobility factor due to the ease on bringing the profile to other environments. On a more extensive approach, having a profile-based scenario allows us for better configuration and data recovery tasks.

Profile Diagram

In this section we will be exposing the proposed Profile diagram as well as commenting the different fluctuation of the states, that will drastically change the way a profile is treated as shared.

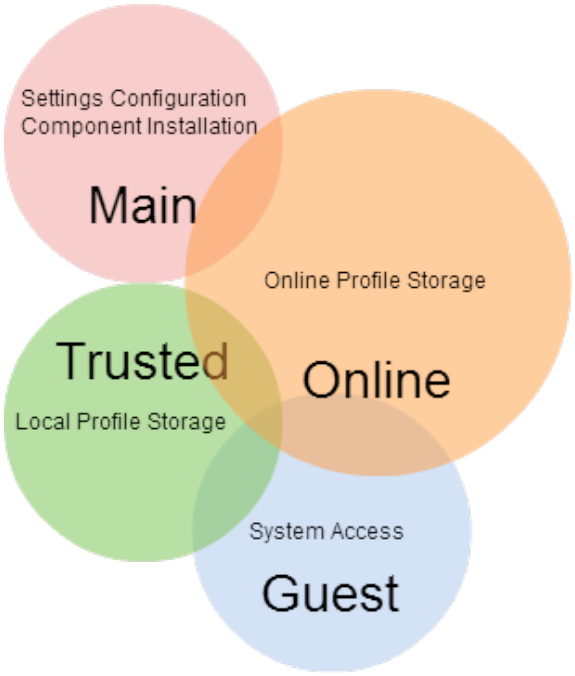


Figure 5.7: Profile Role Distribution

Main Device

A Main Device is, by definition a device that can be trusted with a higher level of privileges and execution than any other device. As a result, a user using a Main Device will always have a higher degree of freedom into the System Settings than a user another kind of device. We consider Main Devices the ones that are embedded into the vehicle by the Vehicle Manufacturer, considering as successors of those any upgrades to them. By using this definition, we ensure ourselves that the any user using this devices will at least have a physical presence on the car. That said, we can still distinguish two subtypes of main devices.

Principal Device We consider Principal Device the one that is on the frontal part of the vehicle. As such, this device is typically only accessible by the driver and, to a certain extent, the person next to it. Therefore, and as inheritance from previous HMI devices as we have already analysed, it is deemed to be the main vehicle's control device. On this role, it has access to special functionalities not able to be present on any other device, even though they can be configured to a certain extent to be accessible. These functionalities range from simple volume control to daylight mode or custom car settings. We recognise the critically of this device, and as such certain functionalities will be disabled while on driving mode or when not; as well as a differentiation from when the key is input or not. The extent of those will be the same as the ones in the previous HMI devices in order to comply with the already existing regulations. Also, System Update and Recovery procedures will be only accessible through a main device.

Secondary Device We can define as a Secondary Device the one that is also installed on the car, and therefore a physical access is required to use it, but not next to the driver. Such devices have a natural limited functionality against the normal devices, or a limited scope of actuation. Such an example could be, again, for the volume controls where the main device would control all of the car controls and a secondary device only the local ones. A secondary device has a minor degree of difference between having the key input or not, due to be itself configured in the general system settings. Also, but, can be configured to different degrees of freedom according to the identified user.

Guest Device

A guest device is any device that is not embedded in the car and therefore is using an external connection. Such devices have the maximum degree of freedom when regarding to accessing non car-based features, as well as being the ones that have the hugest chance to aim for system disruption. As such, these devices can't modify most system settings and are only considered as entertainment devices. Their use is solely regarded as to obtain information about the car multiple variables, as well as external ones regarding the car's otherwise public information such as speed or destination. These devices are also meant for temporary travellers and, as such, are the ones that will be benefited to a maximum extent regarding profile management. Although this last statement, if the user is not online or trusted the data will only be stored locally and won't be able to be used to this extent.

Trusted User/Device

A Trusted User, also called Trusted Device through this documentation, is such that is entitled to be have a huger degree of privileges and freedom whilst operating with the system than normal users or devices. This device or user is an evolution from the concept of Guest Device and therefore is not applicable to the concept of Main Device. This kind of devices need to be authorised by a Main Device, and as such will only be providing for extended features as long as this authorisation is in place; as described by the related use cases. A user might also be trusted, and therefore any device to which it is logged on will also be provided with that level of trust. When a user is trusted but does not have an online vinculation, then the data will be stored on a system basis for that user; to see the behaviour for Online Vinculated Users, check next section.

Online User

An Online User is one that is itself stored in our Online Servers, as explained in the next section. That means, that its data is stored in a centralised Online Cloud totally independent from any system. This way, when a user logs in in any device the system will download its profile information and proceed to provide the user with the look and feel that user has configured. Furthermore, data as plugin customisations and last used data will also be transferred, so the user will be able, in the most ideal of the cases where the new system has the same components installed, to use the exact same user interface as on the previous device. For further discussion on this issue, we dedicate a section in the next Chapter to it. When an Online Profile is authorised on a system, this authorisation is vinculated to the user and not the device. This means that the level of trust

is applicable to the user, and this is the preferred way for trusting because of the technical complications on identifying a device correctly and the possible tampering attempts to supplant this device. This section is coupled to the Appstore-related cases, because it provides the elements necessary to allow the profile to be on the online infrastructure, as explained on the next section.

5.3.5 Online Infrastructure

In this section we will be explaining the concept of Online Infrastructure that we've been taking into consideration on the whole thesis. Therefore, we will be analysing our considerations when selecting the solution for this problem, as well as with the main tasks assigned to it. It must be noted that due to the modular and encapsulated development of this work, this part is completely optional from the system itself. That means, that a system can function properly without ever getting a connection with our online services, but will lack of the extended features that differentiate this thesis from other HMI ones. As part of our Content Management System requirements, this is tightly bound to the Appstore Use Cases such as "Check for Update", "Perform Update" or "Access Appstore", them being part of the cluster of CMS cases and therefore being that the main reason on why this feature has been considered.

AppStore-based Approach

Our approach consists basically of a main Application Store cluster that will provide the bulk of post-sale services. This infrastructure can therefore be divided into different parts, and therefore clusters, but that part is out of scope for this work. With this approach we have tried to emulate the latest trends in Operating Systems post-sale business, such as the ones that Apple, Google or Microsoft have been actively promoting. This way, we aim to follow the global trend on having the vast majority of our income based on the single concept of selling components after the product has been bought. Taking advantage of this fact, we have designed a whole system around it that compromises other features such as storing data on the Cloud, providing Cloud Infrastructure to Application Developers or providing Community-based services to the user.

Cloud Infrastructure Diagram

In this section we will be analysing the suggested cloud diagram altogether with the different tasks that it will be representing and the way the software should interact between itself. We will also be observing the capabilities that each user or device has depending on their level of trust against the system.

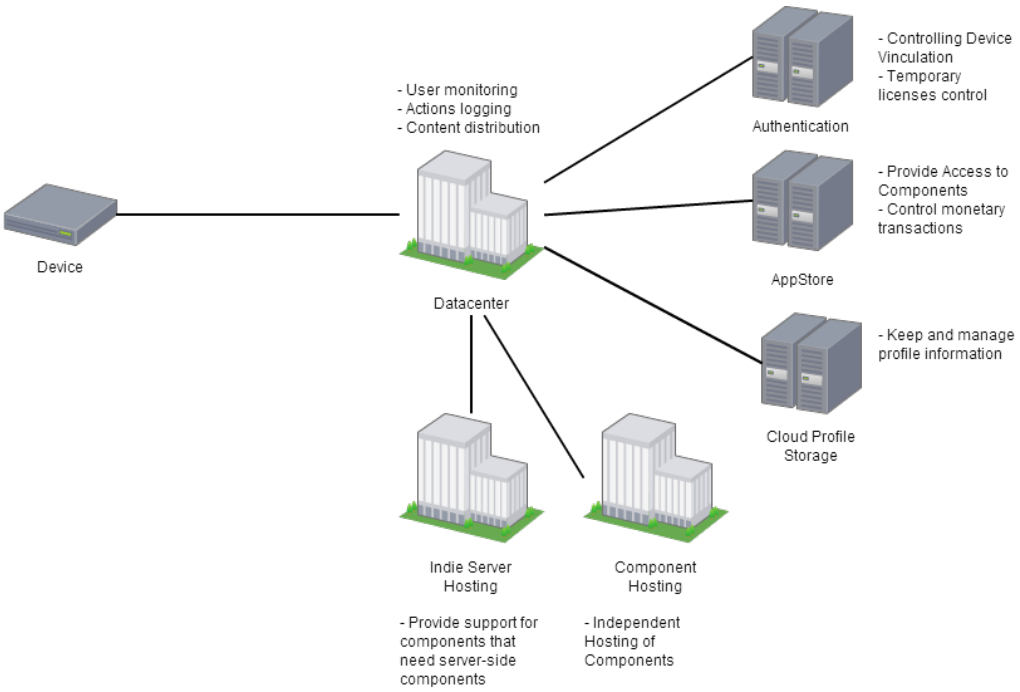


Figure 5.8: Cloud Infrastructure

Component Purchase

As promulgated by a huge number of Use Cases and identical explicitness on the Requirements part, we have deemed necessary the ability to be able to provide for a post-sale model by offering the possibility to the user to expand te functionalities associated with the system. Therefore, any system can grow without any boundaries apart from the technology, which we shall remind is ever evolving in this case, and simplifying enormously the work that the system will need after being released, only complying with enhancements proposed by the manufacturers or users and also technical fixes. This advantage leads us to see a huge increase in profits while keeping the maintenance costs to a relative minimum. To achieve this goal our AppStore system will be offering the possibility to purchase components as submitted by the indie developers, the own company or the own manufacturers, in order to expand the work of the system. To this extent, a partnership with major payment processors, such as Paypal, is required as well as providing the developers with the adequate documentation as discussed on the next chapter. A component might also be offering a trial mode or period, after which either it has to be purchased or deleted, as well as showing advertisement in order to support the own developer. Proper purchase of the component will lead to a permanent usage of the component, but this own component can allow for extra component purchases in order to enhance it. Such purchases must be explicitly declared and will be acquired as a separate pack for the respective component.

Updates

Coupled with the aforementioned feature, we can find the necessity to provide for software and component updates on the fly. This is achieved by using the same infrastructure as needed for the "component Purchase" feature. By making users access the AppStore in order to obtain updates we can also, via the use of proper social engineering, make the user become interested on software that otherwise would not have seen. For this to become true, it is extremely important to not interrupt the user during sensitive work and therefore only ask for user permission on big updates. The motto of this idea is to provide silent updates to the user as long as they are considered as small enchantments or hotfixes whilst having the user accept for bigger updates due to the possibility that they will take more resources or time to complete.

Trial and Demos

As explained on the previous section we have considered a Trial and Demo option for our system. This means that our system will be providing the developers to offer their products on a lease, that is, free of charge under certain limitations. Apart, developers will also be able to offer purchase-only products. The idea before the Trial of Products is basically that any product is always better sold after a user has been able to try it, in order to assure that the purchase is going to provide for the user's needs. Often compared with buying clothes, this system is highly effective in terms of marketing and has been widely used by all our referents in this section. By offering the developer the chance to show advertisement we can ensure that a group of indie developers are going to participate in this scheme due to the products being free of charge and the user only paying for a version without as in order to offer support to the own developer.

Online Services

One of the key innovations of our work, and based almost exclusively on Online Gaming services, is offering developers the possibility to use our own Online Infrastructure for their developing needs. Using this approach, similar to Microsoft Windows Azure, we can ensure that the developers will not need to have an external provider for the service, thus reducing costs and being able to offer the service as a percentage from the income that is transferred to the developer. By eliminating costly monthly fees we can ensure that even though a developer might be willing to give up a product, this product will continue to have its full set of functionality till a viable replacement is found. The main objective between this idea is therefore to avoid the users from feeling scammed after purchasing a product and this one becoming discontinued.

5.4 Technology Proposal

As indicated in the main architecture selection, we will be taking a Web-based approach in-between the client and the server in order to aim at the maximum number of devices and not bounding ourselves to a single device manufacturer. In fact, the main objective of this decision is to provide for a complete genericness on the used device, whilst performing at the maximum extent of the device's capabilities. That way, the user has the power over its user experience based on the choice of hardware and software allowing for hardware replacements after the vehicle has been shipped.

5.4.1 Technology Usage Diagram

In this section we will be exposing the diagram between technologies used and the use cases that they will be tackling. In this diagram we also have grouped the use cases according to functions, and gives a general overview on which technology is on charge of each ground (which later will be used on the discussion chapter). The elements market on yellow indicate external technologies that could be shared with the prototype.

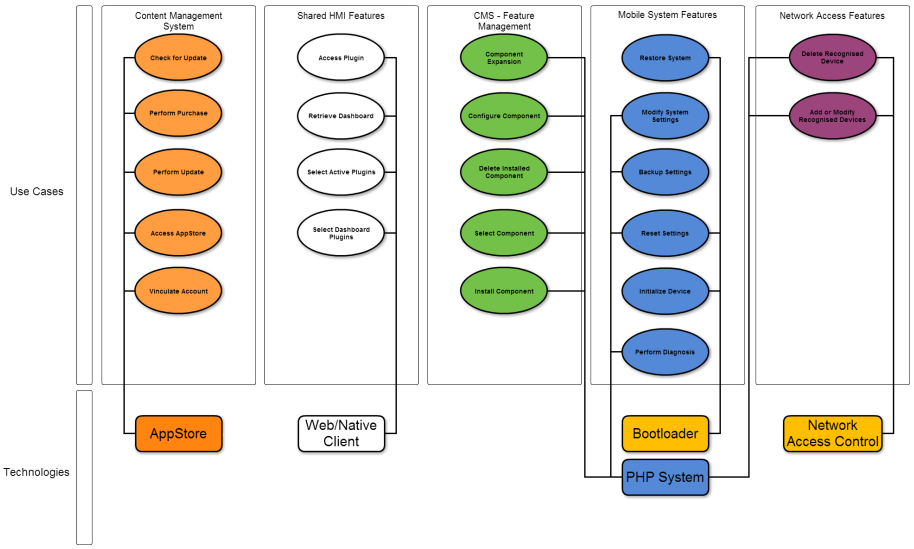


Figure 5.9: Technology-Use Case diagram

5.4.2 Server Technologies

In this section we will be explaining the major technology choices for the local server-side part of our application with the on-line part being discussed on a later stage. It must clearly be noted down that the main chose of technologies on this side has to do with the current hardware architecture and the imposed requirements.

PHP

This is meant to be the server technology, on which the vast majority of the server software is going to be based on. The choice of this technology is motivated on part because of the technically lightweight of the base software, but heavily because the already present AMP architecture (Apache, MySQL and PHP) on the server where this will be running. Therefore, and in order to adapt to the small pool of resources available from the device where the software will be located, the system needs to run along its counterparts. Therefore it is deemed as appropriate to use the already available suite instead of changing the choice of technologies and adding an extra overhead, to this extent choosing PHP technologies above the other set of available ones.

File system Storage

Although we have available an SQL Server, we do not deem as necessary such and therefore we aim only at a pure file system-based access in order to simplify for the access of user data. We also define that only a small amount of data is going to be stored in this way, on chapters that follow, and therefore is not deemed necessary a huger degree of control on its storage and access.

5.4.3 Client Technologies

In this section we will be explicating the selected client technologies, as well as the main reasons behind them. In this section we have only been constrained by the necessity of having at least a Web-based approach. Therefore, this section has selected the very best of what has been documented in the chapter Related Work.

Web Client

This will represent the visible part of our thesis versus the actor "User". The Web Client will be communicating with the Server in a typical Server-WWW architecture via the use of a RPC and Publish/Subscribe model. To this extent, we will be using a technology called "Web-Sockets" in order to provide a dual-channel communication strategy. By using this feature we will be able to provide real-time updates to every client, therefore converting the experience into a full OS-like style. The logic on the client will be controlled exclusively by JavaScript while also using the latest HTML5 draft specification that will allow us to use Message Queues while preserving sandboxing between elements thus covering the main concerns on the fields of security. This element is the one in charge of performing the communication with the server, and therefore is the one that complies with Use Cases as "Retrieve Dashboard" or "Access Plugin".

HTML5 IFrame element The new HTML5 IFrame element contains crucial new features in contrast with the old HTML 4.01 one. In concrete, we will be using extensively the feature "sandboxing" provided by the draft specification. This feature allows us to completely "sandbox", that is, encapsulate a certain page into a closed environment thus preventing it from affecting the normal behaviour of our system. Whilst this element can be encapsulated, it can be also fully manipulated by the parent (higher privileged) elements of it, therefore offering a huge degree of control over its process.

HTML5 Messaging API The new HTML5 Messaging API will form the basis of the insider communication in our product. That is, it will provide for the data transmission between the different encapsulated components and their superior layers of abstractions. Thanks to this element, we will be able to provide real-time data to the different components without loosing our sandboxed model. This element also allows us to simplify and centralise all the data-communication procedures that we will be using.

Native Client

Alongside with the Web Client we will be providing a Native Client for the most common known Operating Systems. This Native Client will provide for the needed HTML5 infrastructure in environments that don't support it. Also, it will allow us to interact natively with the device thus allowing for new enhancements on certain devices. To achieve this goal we will be using PhoneGap, as analysed previously, due to it being free of charge and because it provides us with the main features required for this work. Therefore the application will only be designed once, for web, and extra modules will be added to the Native Client one on-the-fly.

Trusting

This system is going to be based in the widely used concept of Certification Authorities (CA), thus only trusting those components that are claimed to be signed by a set of CA's that are recognised by it. The set of trusted CA's will be updated dynamically when they become available, on the

internet, and on every system update. As explained on the subsequent sections, these updates do not need to be over the net, thus allowing for this feature to be always up-to-date regarding the latest component installation. We differentiate, though, two different kind of trusting levels. As each component needs to identify itself and the manufacturer, we will assume that for a component to exist the corresponding identifying signature will also be existing.

Centralised Communication API

Our system will be communicating to the server via WebSockets, but this communication will only be effective when several client-only filters proof that the desired transmission data is error-free (this will also be done on the server).

5.4.4 Cloud Technologies

The Cloud technologies represent the group of technologies that will be used on the infrastructure associated with the thesis that will be located on the cloud. Although a requirement, this is not explicitly needed for the work as it provides only extended functionality that could be supplied by other standalone means. To provide for these technologies we have choose to use the same set of technologies as with our Server, due to the similarity with our already proposed Server technologies. To this extent, we propose the use of an open-source AppStore called "PrestaShop", that provides the needed AppStore functionalities (and which are out of scope for this thesis) and also is also based on PHP.

5.4.5 RACE Technologies

The communication with the RACE Project infrastructure will be done totally via Web Services, mainly due to it being the easiest way to provide an abstract specification on it and also because it will be based on the same patterns already used on the rest of the components (RPC, Publish/-Subscribe). This has been reached on agreement altogether with the RACE Project.

6 Discussion

In this chapter we will be discussing the outcome of the work in the proposed solution against the analysed requirements. We will be observing why certain decisions were made and which aspects, if any, have not been covered by the current solution. Furthermore, we will be observing possible alternatives to the selected ones, but in major words we will be reviewing on the real motives before each controversial decision.

6.1 Non-discussed issues

There have been a number of aspects that have not been proposed, either because they were imposed or because they were out of scope from this work. In this section, though, we will be observing those that have not been documented but are inside the work of the thesis. The reason on why they have not been documented on the work is because their final specification hugely resides on a real world implementation, and therefore can't be documented properly without reaching an advanced stage into the product development.

6.1.1 Development Procedure

Although we have been propitiating the idea before having third party business or individuals developing additional software or complements for our system, we have not discussed or offered a correct explanation on how these developing procedures are going to take place and neither on which documentation or indications they will have at the reach of their hand. Even though these impediments, we can reach the consensus that a minimal set of features are in need to be provided by our work in order for any developer to start developing software associated with our system.

Developing environment

One of the key steps in any software or hardware development is the ability to test a certain product against its target system. Doing so any developer can ensure that its software will be complying with its objective and work as expected. In order to help with this objective there are multiple alternatives to be taken into account, each of them promulgating its own requirements. As we aim to target at Indie Developers is key for the system to be of ease of use and simplistic, as well as providing the maximum information possible. From all the following alternatives there's none set as a major winner, to this extent we would propose all of them altogether as valid for proposing an environment.

Emulation The most logical option is that of providing for an emulator of an entire system. Whilst this option might require a relatively huge resources usage on the developer side, it can provide for a complete test environment coupled with the ability to check against multiple sources. For this instance, every car manufacturer could provide its own image to be emulated in order to allow the developers to check against a specific model or features. Also, it would allow businesses and manufacturers to automate tests against a wide range of versions in order to ensure their compatibility and avoid any conflicts on, for example, system modules.

Real-time deployment One of the typical solutions to this issue is the one to allow the developed software to be directly installed on a real device. The advantages of this strategy are quite clear as it allows the developers to test their software directly on the field and experiment with any issues or nonconformities directly without the need for user feedback. The disadvantages, but, are also clear on this option as it makes mandatory for any developer to have a compatible system purchased and therefore alienates Indie Developers. Also this option doesn't allow

for wide-scale testing and therefore might be incompatible with certain combinations (again, for system modules).

Online Service In a turn of events we propose the creation of an extensive testing environment as online service, expanding the emulation and bringing to higher levels. Though this option requires a higher development effort on the product part, it heavily rewards its completion. By ensuring for an online suite we can provide any developer with the ability to test his software on our systems, and receive real feedback of the outcome. Whilst not as direct as the Real-time deployment, this will allow developers to share their outcome with other multiple developers thus allowing for a higher cooperative level on application development.

Developing API

Having already discussed the developing environment, we have a key issue remaining. For any development a declared Application Programming Interface is needed. In our case, but, we can only provide a handful of features that have already been declared in the solution part. We have missed out the big issue regarding the features exposed by the end vehicle. This issue itself was raised with our RACE Project counterparts and a mid-term solution was reached as a consensus. In order to provide the developers with an updated API from an ever-changing specification, we decided to take as a base the RACE Project as a minimal API for any manufacturer to implement. This implies directly that the developer could use the documentation provided by the manufacturer and bypass any need for secondary documentation, because everything the system would do is to check the configuration and privileges part. On top of that, we are conscious that every model might have its own specification and therefore may require additional work on that aspect. To overcome all of these problems, we evaluated the following decision.

Dynamic API Generation Our main objective is therefore to bring to a minimum the need for documentation on our side, drawing from the documentation already established by RACE Project altogether with the one provided by the other manufacturers. In long term, we aim to propose a common framework with manufacturers to work together in the stabilisation for a minimum common framework, base for any custom features expansion; pretty much based on the current relationship between the WWW Council and the different browser manufacturers, with the first one providing for the standardisation and adopting suggestions provided by the seconds. To provide for this aim, our system would then dynamically generate the set of available methods drawing directly from the ones exposed by the vehicle (e.g., [MRS09]). Due to the standardisation between car manufacturers, the features provided would be of a similar calibre.

Development Framework We're concious that a transition to our software might imply heavy work on any previous HMI, in order to simplify the manufacturer's work we propose the creation of a developer framework similar of jQuery for JavaScript. That is, a framework that would expose the features of every model in a similar interface, while processing in an internal way the conversion to the appropriate transmission method. This proposal has also the clear advantage of propitiating the early adoption of our software due to the facility to integrate it into existing car models. Therefore, and because the Framework would be Open-Source, we ensure ourselves the maximum compatibility with any existing cars due to the most previsible growth of the so-called scene. This group of developers would then be in charge of providing unofficial compatibility with existing car models, as well as providing for installation documentation and procedures, while our business could centre in the implementation of it on new models and manufacturers while obtain official support from those.

Developer Help/Community

As exposed on all this work, and specially in this section, an innovation on this work is that is centred around the term of Indie Developers. Whilst there have been already solutions that have aimed on attracting Indie Developers to an already existing HMI, our idea is to make these developers as a key part of our system by developing it from direct feedback of those. In order to achieve this goal a heavy informative campaign is needed before any product model might go

live, so as to have an already existing component base that would be attractive to manufacturers and users alike. Also is important to keep up this level of collaboration and enhance and expand the value of this community by allowing these same developers and other users to help solve any problems they might encounter whilst in the use of our software. To comply with this aim we propose the creation and establishment of community-like features on our online infrastructure, with a huge component of media and social engineering to allow users that collaborate highly to achieve our goals to be rewarded accordingly. And the same concept could be applied by users that act against the principles and objective of this community. With this proposed incentives we aim to have a self-growing community that would, at the very end, be auto-propulsive on our work's success. The key idea is to get users get users, which is more effective and cheaper than heavy advertising campaigns for the use of a product that, at the end, will be chosen by a manufacturer based on the willingness of their users to use it.

Open-Source Code and Examples

In the same line as previously exposed, it is in our goals to provide our thesis as open-source. Altogether with this, we would also be providing with examples and actively develop to enhance this software. Whilst some might argue that this path is of risk for such a thesis, with a huge level of competence before it, we can easily claim that it is not. The reality is that on the current environment we have a lack of feasible products on this sense. Therefore, our best expectation would suggest for a monopoly-able scenario very much similar to the start of the already known Operating Systems. Whilst this has a potential for income on the short and medium term, at the end new competence would arise those being totally open source and therefore ruining our expectations because of better opportunities for developers. Having this possible scenario in mind, we have decided to take a different approach very similar to that of "Google Android" and start for an open-source minded system that would allow for the development of an open-source software with all the guarantees that we would have the monopoly on services to this software [Inc13d]. Therefore, our main business model would be solely oriented at the offering of online services such as the AppStore, Update services, maintaining the community and investing on the creation or sponsoring of new Indie projects. This under a policy of responsible billing and fair usage would avoid having open-source developers think of creating an alternative to our software due to the associated costs relating to these services. As commented in this section, one of the main reinvestment points of the income would be on enhancing the available software, develop new one, increase and improve the existing documentation and reward new software. All of this coupled with the maximum transparency levels that our business could provide would allow for the users to think positively of our business whilst providing a more humane feel to it.

6.1.2 Critical System Failure

One of the non-discussed points recalls the possibility of an unexpected all-out system failure. Such case, thought unfortunate and a direct implication of a failure of our security and self-redundancy protocols must still be considered. When one of these cases arise the most common solution would be for the user to bring the whole of the vehicle to a car manufacturer and for this one to repair it. Although this might be a feasible solution for some, a more alternative and innovative approach has been considered. The idea of allowing a user to solve itself its problems is not new and has been tested in multiple occasions along human history. Our main suggestion flies around the fact that nowadays high capacity solution can be provided while these being still portable (flash cards). As an added value to this fact, we can consider that most equipment today has support to read this format and therefore would make sense for the user to be able to extract this storage mechanism and connect it to another device. Following this suggestion, the remnant requires for software that help guide the system to submit any error that might have occurred and restore the software to be created. This effort, though, would not come without reward if easy enough, as it would highly reduce the customer dissatisfaction in case of an unexpected failure, whilst also providing first hand information to both the user and the associate business on the causes of it.

6.2 Main Problems of the current solution

In this section we will be discussing the most obvious problems regarding the current solution, the causes on why those originated and any possible alternatives that might be proposed. Although these alternative have been studied, an argument will be provided along those in order to proof for the failure to apply them.

6.2.1 Security

One of the main concerns of this thesis has been the one concerned with actual system security. As follows we will be analysing the two main concerns and observe on which solutions have been observed, all of those without fully solving the proposed problems.

Unwanted intrusion Avoiding a foreign intrusion or any unwanted modification of the system is crucial as it could end up provoking an accident and the associated possible civil and penal responsibilities. Such devastating scenario must, at all costs, be avoided. On first place, a number of legal actions and assurances must be taken for our product to be considered safe and protected by any customer that might use it. In second chance, we have to ensure that our system is as closed to intrusions as possible, and that the system modification options are the same of those on current systems. This assurances could partly lift the burden of any accident related to the software from itself, but still provide no clear assurance to the customer of a full extent. This fact is mainly due to the possibility to have external devices interact dynamically with our system.

Code protection Even through this assurances, we have still the concern on unwanted code retrieval and duplication. This is an unsolvable problem on the current technologies due to most of the client code being based in JavaScript, which as previously shown can't be protected against this fact. This problem could easily be solved by bringing part of the code to a closed environment on the server-side, but would again require higher resources on an already strained server, which due to the imposed architecture requirements is not possible to be modified.

6.2.2 Standards Enforcement

In this section we will be analysing the problems derivable from the fact that the already-existing standards, and future versions, need to be enforced on any trusted component and the problems derivable from those.

Component enforcement Every installed component needs to be complying with the existing regulations in order to be approved for use. That, but, requires in huge measure the fact that these components need extensive testing in order to become officially allowed to be installed. Due to the most probably huge number of components going to be submitted, and for the imperative need that each and every one of them to be complying with the standards we require of a huge number of automated tests to be ran on every component (and thus, also provided for the developers) in order to ensure the compliance with those. Apart from this fact, it is quite obvious that an human interaction would be needed in order to provide for the final approval. In order to achieve this we propose the creation of a system similar to "Steam Greenlight", where the users could positively vote the software they think comply with the regulations and therefore speed the position of that software in the approval procedures. Users that successfully predict the behaviour and compliance would be rewarded based on their collaboration information only if it was of use, thus avoiding those users that would always provide a green light. Along with this proposal, a maximal approval time would also be guaranteed, with its time to be debated according to the predicted time needed to approve the already submitted components.

Illegal Installations One of the ever-growing problems of our generation is that of piracy, where legal software is replicated illegally and therefore evicts users from having to pay to use that software. Whilst the already known techniques of Trial and Demo software have been provided, and others such as Application of the Month or similar on the sight, preventive action needs to be

taken in order to ensure to a maximum the existence of those Illegal distributions. It has been long argued that such a solution is not factible, till now there are still a lot of problems in this sense, but in our case it is more than the problem of having a loss of revenue due to piracy. By allowing illegal installations we would allow most potentially the installation of unwanted or disruptive software that could thwart any concentration effort on the vehicle and lead to the most disastrous case we could face. In order to avoid this, we have already considered that a user should take sole responsibility for the installation of software, but further legal action should be planned for those that make the software. Also, a blacklist and on-system checks should be proposed in order to ban certain software that might be to a high probability harmful for the user and/or the own system.

Driving mode The standards dictate that only a handful reduced set of features should be available to a user on a driving mode. This simple phrase has a direct meaning on both the privileges control on the client side and the component approval system on the online side. Therefore, every component claiming to be available for "driving mode" should be tested in such and ensured that complies with the respective standards. Still for the severity of this affirmation, we can ensure ourselves that only those devices executing in a "Principal Device" mode shall be affected by it, rendering this system to a nearly maximum extent without application on any of the other devices. To ensure this mode compliance, only officially approved applications will be allowed to be executed on non-developing systems. This is to burden the responsibility of the component correctness solely on the driver, thus eliminating any potential harm from the user.

6.2.3 Slow responsiveness

In this section we will be tackling the most probable problems regarding a lack of responsiveness on the server-side of the system. These limitations come both on the choice of technologies and hardware, with both of them being considered as mostly inappropriate for the proposed system. Alternatives will be discussed on following sections, but on this section we will centre around which techniques have been already proposed and which have been long-sighted in order to improve the software should heavy problems arise with it. Due to the current state of affairs our system is in a foreseeable manner not capable of providing its uses on large-scale deployments, such as buses, where the amount of clients is undoubtedly high and the responsiveness requirements are also considerably high. Our system therefore has been designed with a "responsive" mentality, therefore centring around the need for he user to feel that the application never "hangs" and is at any time informing the user of the status on the current operation. By making the application more human, we can expect the user to soften its stance against heavily loaded systems whose actuation might last a few minutes. On a second hand, we also have specified component control systems that will ensure that no component "hangs" or spends too much time on a task by controlling the resources that a component uses. Such behaviours will be notified whenever possible and corrective action will be taken in order to ensure that the user never has the feeling that something has gone wrong.

6.2.4 Online Profile

In the previous chapter we have defined the concept of Online Profile, altogether with the notion that this is an optional concept for any system due to the necessity to have access to Internet in order to grasp it. Although the idea is perfectly innovative and stands with the latest trends in mobile Operating Systems, it provides legal and technical problems on bringing it as effective as we have deemed, we explain them as follows.

Component Sharing

In this section we will discuss whether any component that is installed on a system should be usable by all users or not. Theoretically a component installed in a small scale transport, such as a car, should be available by the maximum number of users of that transport. These would amount for five to six licenses on a medium basis, which would represent no problem. The problem arises when we talk about huge transports such as a bus or even bigger such as ferries or aeroplanes.

Though that's quite on the future for this system, it is good to start asking ourselves what action should be done on those cases. The idea we are bringing around here is that when a product needs to be used in one of these environments, its licensing costs are going to go on increase. That is, because the number of people using this product is higher than what it the sale was originally meant for, therefore increasing substantially the revenue that provides this component to the company that purchases it.

Component Downloading

In this section we will be discussing on how to handle the unclear situations where an online user is using a license that is not purchased for the device but on the user. This section is tightly bound with the one following called Licenses on which can only be applied if the licenses are used on a per-user basis and not on a per-device basis, where this section would be meaningless. Following we will be discussing the proposed solutions for a user license-based system, due to the technical and legal complexities associated with such.

Cache This technique consists primarily on downloading and storing the component on a local-basis on the device, therefore avoiding any system-based interaction. This option itself might be cached by the system, but in any case the system won't install the package, as it will provide for the package checks. With this option, only those user accounts with the package activated will be able to use it. This excludes the main device of the account, to which the package will be automatically installed. This option, but, implies that extra security should be added on our side in order to avoid unauthorized usage of a product that has not been bought, altogether with the legal assurances that they would need to be provided from our side.

Temporary License With this option any user can grant a temporary license for a component to any device. In order to this, the system would need to have a tracking on how many licenses may be granted to the user and for how long would they be enabled to be active, if the authors deems as such. This solution inherently has its very own complexities because it requires the developers to highly specify on which scenarios may a user use the software more than in their main device, but would hugely help the user while using devices on a temporary basis and moreover would be a perfect way to allow other users to test the full functionality of a certain component. Overall, but, this requires extra management from on the user side which in turn worsens the user experience in the task to share its experience. This option, though, also requires from further legal investigation which is totally out of scope on this work.

6.2.5 Licenses

In this section we will discuss to which element the licenses should be awarded. Licenses are due from the purchase of a component, after which the user has the right to use the component with all the features that were announced on the license purchase. It is also noted that the developer controls the type of license purchased and any other license possibilities, as configuring the aforementioned section. A developer, though, can't revoke a license from a user as this right is exclusively reserved to the own business and can only be done with a justified cause.

Device

Awarding the license to the Device means that each user using this device would be able to use a purchased product without any restriction unless the owner of the device decides to. Therefore, the developer would be selling in a style similar to that of the current Application Stores for mobile devices. That would mean that once a user has purchased the product, any number of users might be using the product without any possible control, therefore opening the door also to possible complications on the legal side due to unauthorized users using a non purchased software. To this extent, we observe the following aspects.

User Limitation An interesting idea while considering assigning licenses to single devices is to contemplate which kind of use will that device have regarding the sold license. Therefore, different licenses can be sold depending on the kind of user, as such as “Enterprise“ user, one that belongs to a company, “High Occupation“ users, the ones that carry a huge number of passengers, such as a Ferry or a Bus or “Limited“ users, such as those driving a car or a motorcycle.

Device Owner In accordance with the aforementioned text, the purchaser would then be able to control which users could use a certain piece of purchased software, therefore managing them and only allowing certain devices to use them. This way, for instance, a high occupation user could purchase a component for use only on certain devices, such as the embedded ones, while purchasing a license for another component that would be in turn available to normal users.

User

Awarding the license to the user means that this user would be able to use the component in more than a single device. By allowing this fact we avoid the situation where a user owns multiple devices and doesn’t want to purchase it for every single one. This method reduces therefore the expected revenue of the developer on a first glance, but by adding the possibility to set the amount of devices on which a component might be used may simplify and aid the developer’s revenue.

Amount of Devices The first idea that comes into mind when speaking of user-based component licenses is the fact that a developer’s income ends when the user purchases the component, because the user takes ownership of this purchase and therefore can use it in a unlimited amount of devices. Therefore and available option would be to simply end this by providing the developer the ability to select on how many devices the component license might be used on. On from this, a standard price per extra spot, also configurable, would be provided thus allowing the user to purchase allowance for extra devices.

Online connection This kind of license has the advantage to simplify the offline installations in contrast with the one previously exposed. It’s main advantage is that it allows for a single user to register a certain kind of component once the device goes online, and for the server to keep control if multiple users try to take advantage of this situation by denying them. Therefore, at the end a user could avoid committing unwanted mistakes by only installing on the owned devices and the developer could be assured against unauthorized mass installation because they would be denied online synchronization of the component if that was the case.

6.3 Alternatives to the selected technologies

In this section we will be discussing the multiple alternatives to the currently used technologies, and the main reasons why they weren’t even considered in the market analysis. The listing of these technologies is therefore useful in the objective to further improve and specialize the product by reducing the base framework and unneeded resources associated with the base where the product is going to be working on.

6.3.1 Client Technologies

In this chapter we will analyse the different client-side technologies that are current alternatives to the already selected Web-based model. On this case, we will solely analyse which kind of technology could be considered or which feature could be dropped in order to achieve a greater array of compatibility.

Native Client

In this section we will be analysing the impact of the decision regarding the imposition of the necessity of a native framework in order to comply fully with device integration. Explicitly, we

will be discussing about the necessity of setting such a requirement given the current status of the standard, web, client definitions and technologies used. Therefore, this requirement aims solely to fulfil the necessity to run native applications on the user-side such as navigation applications and a lifting of these limitations would possibly simplify application deployment and related work. As follows we will be analysing in this concrete case the possible alternatives that could be taken around this fact.

Web Limitations The idea behind using such a framework is to overcome the limitations associated with the web technologies. In concrete, it has been deemed necessary to use such a framework in order to provide for the sole necessity to run a native application. To this limitation we can provide alternatives that have already taken place in other similar architectural systems, such as the use of URI's. These, standard in most mobile-based operating systems, allow for a website to execute any registered application altogether with configuration information sent to it (e.g., [App13b] or [Mic13a]).

Alternatives As an alternative to resorting to this approach, a radical solution was proposed. In concrete, it would be ideal that all the applications that need to be provided with the product to be coupled with the same system in order to maintain coherence inside the structure. In this case, we proposed to have the navigation application built completely with web-based technologies in the very same fashion as other already existing ones. This is already commercialized in other products and therefore could be an easy alternative in order to simplify user experience by avoiding the hassle of having to change between the browsing application and the navigation one, and also allowing the user to have real-time navigation information on the screen [Inc12].

6.3.2 Server Technologies

In this section we will be discussing about the previously selected architecture and the items that have been forced into scene. In concrete, we will be proposing solutions to the innermost problems that have been found working with the selected technologies and now been integrated as limitations of the very own thesis. As already commented, these limitations were established as part of the thesis requirements and therefore have not been extensively researched, but we will be providing one of the multiple alternatives that could be used if such limitation was risen, leading perhaps to a partial reanalysis of the proposed architecture.

Motivation

The reason why an Apache + PHP + MySQL architecture was selected for this thesis has its roots into the fact that such a architecture is already used on the limited server hardware. Therefore, in a simple aim to reduce the server costs, and due to the way in which this architecture revolves around, it was deemed as essential to build a solution around this highly constraint limitations in order to use the maximum amount of resources on serving the responses.

Alternative

The proposed technology is Node.JS, a full-blown JavaScript based server. The selection of such is due to two facts: it is more efficient and it simplifies the developer's previous knowledge to adapt by removing PHP and Apache from the list of required technologies. Therefore, a single web developer could implement the whole structure thus reducing also the initial costs for the thesis. The use of this technology would imply an increase on performance respect the old architecture, and also lift limitations regarding the stateless environment regarding PHP [Tec13a].

6.3.3 AppStore Technologies

In this section we will be analysing the selected AppStore technology as well as viable alternatives to it. In concrete, we will analyse the main motivations behind the selection of such a solution along with a possible alternative, again not the full market has been researched due to it being totally out of scope for the environment provided for this work, for it. On this section we will be

taking as the base proposal being the current AppStore utility and why it is not useful from our point of view.

Motivation

The main reason why a selection on the current AppStore technology was made is because it uses the same set of technologies that we initially selected. To this extent, we decided to use an Open-source AppStore with an inherent AMP structure as PrestaShop, that provided us the ability to use the same base hardware platform as a common denominator for all the infrastructure, thus simplifying development. Our research, though, only provided a worthy competitor to this selection called "OpenCart".

Alternative

A collaboration with a cloud services provider (e.g., Amazon [Inc13a]) could be done on the first approach, helping our system overcome the possible deficiencies and providing security benefits against possible intrusions (e.g., Amazon includes built-in protection [Inc13b]).

7 Future Work

In this chapter we will be discussing the different approaches that could be taken after the finalisation of this work. We will be analysing the different ways to provide extended viability to the thesis and improve its analysis, altogether with different paths to convert the created analysis into a real market product.

7.1 OEM Integration

Along this work we have identified a vast set of stakeholders and features that interconnect with each others and satisfy their necessities. A major lack of this work is, though, the lack of major communication with the Manufacturers. Deemed as a key flaw of this thesis due to the reticence to abide by the RACE rules, due to being a base requirement of this thesis, this work has the ability to change that if further work on it is done.

7.1.1 Proposals

In this section we will be analysing the different proposals regarding the immediate demonstration of the viability of the work as a whole, that is, convincing arguments needed to defend this work in front of other elements that might be required for it to go ahead, be customer opinion or institution approval going through OEM satisfaction and willingness to adopt the work as their own main architecture for their developed products.

7.1.2 OEM Analysis

One of the main proposals is therefore the provide the results from this work to the different interested manufacturers and obtain their own evaluation of the outcome of the work, lastly incorporating them into the work thus enhancing it. This proposal is deemed as essential for the progress of this work, because this work requires for a vast sector of the market to approve the selected outcome. By doing this task, we would also provide vast enchantments to this work thanks to the direct collaboration of the multiple companies by obtaining their expertise and ability to perform trials on the outcome of this work.

7.1.3 Prototype

Another of the selected proposals is to provide for a prototype demonstrating the viability of the work already done. This prototype would therefore aid in our crusade to expand the base amount of manufacturers willing to join the revolution started by this work, as well as providing feasible proof to the world that such a work is possible. In the progress of implementing such a prototype would be a key element to follow strictly the specifications here provided, and if due problems arose then a viable change to the specifications could be possible thus enhancing the work itself.

7.2 Funding

Another of the main questions that lie ahead in the path of bringing this work to a viable product is that of funding. So far, the already selected funding goes tightly coupled with the project RACE. This association, though, is deemed not the best to this work due to the necessity of it to be extended to other systems already on use. Therefore, we propose the following steps.

7.2.1 OEM

The most logical and less traumatic step into funding this thesis is by using the direct collaboration of the different manufacturers into it by funding a standardisation consortium that would surely bring this work into reality by providing enough funding and resources for the result to be flawless. The major drawback of this approach is loosing all the independence strength and therefore falling into the standardisation of a few manufacturers in detriment of the vast majority.

7.2.2 Public Institutions

This would be the preferred outcome to progress with this thesis. By providing the public institutions with the ability to participate in a thesis of this magnitude we could get practically illimitable funding and resources due to the vast contracts that these institutions have with the manufacturers. Moreover, this kind of association will allow the developers of this work to avoid any huge manipulation of it by the manufacturers by removing them from the list of essentials stakeholders and relegating them to the position of preferred ones. As a last note, by doing this deal the work would surely abide by all the terms and regulations of the institutions and highly speed up the needed approval by them.

7.2.3 Micro-financing

Lastly the second most preferable option, thought the most complicated ones in terms of later standardisation, is the one of micro financing. This solution allows possible users to install this system into their own vehicles and therefore provide for a huge amount of testing on a first glance. The drawback is precisely the necessity to provide this, and therefore a stable product, on a very early stage of development due to the users being the major stakeholder of the thesis. While being able to achieve standardisation on a later stage without manufacturer intrusion, acceptance by the public institutions will also be standing at a higher degree of completion.

7.3 Testing

Highly coupled with the aforementioned section, on this section we will be discussing the possibility to have external help in the testing and integration efforts of the development and deployment of this work. Concretely, we will use paired solutions of integration and testing by changing the public to which the test editions will be directed to.

7.3.1 OEM

This would be the preferred option regarding to integration possibilities. Allowing the different OEM's to deploy the work to their systems and obtain their test results would amount for the best results all over the different methods analysed on this section. This would also comply with all the security features of the vast majority of countries, because the manufacturers would be providing this information altogether with the required legal assessment.

7.3.2 Business other than OEM's

This would mean having other business with a huge number of deployment scenarios preparing and testing the developed product. While this seems as a viable option problems arise immediately in the form of lack of standardisation and exhaustive testing altogether with the requirement to obtain legal assessment in the multiple countries and the fact that the exhaustive support should be provided to this businesses or institutions.

7.3.3 Particulars

This would be a highly recommended option no matter the funding or testing scenarios. The main motive behind this reasoning is that having a huge number of users test totally different scenarios

in their own systems would amount for years of testing being reduced to a matter of months. The drawback, as in the previous section, is the necessity to provide for a vast and extensive level of support to users who most likely will lack of the required knowledge in order to effectively diagnosis and solve any problems that are likely to arise along.

7.4 Product Enhancement

In this section we will be thoroughly discussing, based on the outcome of this work, the different enhancing possibilities that could be due to apply on an imminent basis or could need further research on them. These features have not been incorporated on the other sections of the work either because they are to a certain extent out of scope or because they lack required extra research which itself is out of scope for this work.

7.4.1 Security

One of the base pillars around which a vast part of this work revolves is that of security, to whom we owe the fanaticism expressed in some sections of it. This dedication is mainly due because the strict requirements by the compelling authorities for the kind of products to which this work relates to. As could be easily interpreted, this exactly means the never-ending need for further security analysis and improvements on this field, from the own product deployment and normal function safety to protecting it from external intrusions and modifications that could invalidate these rules.

Illegal Systems

In this subsection we will discuss the strategy that could be taken against those systems that try to use software that has not been authorised or in an unwanted basis. This use is naturally against the own system rules, but what also concerns us is the invalidity of the used software due to it not going through the extensive tests that normal software would need to comply with in order to be eligible as feasible components for the developed software.

On-line Ban One of the first ideas that come into force when tackling illegal system usage is that of the already existing mobile platforms, that is the base of the current console and on-line systems and therefore known to most users that use this kind of devices. By relegating control to on-line solutions we can easily simplify and enhance the identification of undesired systems and remove part of their functionality while also taking legal actions against the users or allowing themselves to go forward in this aspect. To achieve this objective the on-line ban pattern would be used, thus limiting on-line functionality and restricting the ability to update or use extra features due to the client system not passing the minimum verification tests. One of the tasks of the future work will be to do a full-through analysis on this strategy and provide the essentially required legal framework to act on this sense, without which this option is not even feasible.

System Blocking Another of the legal points that will be taken into account is the possibility to block partially the client system in case of an illegal usage. This option is limited if the own software is modified, and also requires a huge amount of legal framework investigation behind it in order to provide cover for its implementation and deployment. An added option to this would be the necessity to immediately take legal actions against the person that committed the infraction while preserving its minimal functionality and allowing for all security rules to be preserved.

Unauthorized Clients

Another of the major concerns of these systems is the more than enough possibility of external intrusion by unwanted clients. While already providing for previous approval of devices before use as a requirement, it is clearly noted that the case of an advanced attack against a huge system deployment could be done in order to take advantage of some features that could ultimately lead to security problems on the own system. In order to avoid this scenario it is deemed that a huge

security effort is needed in order to pave the way for this product to make into bigger deployment with dozens of simultaneous clients using it.

Cyber Attacks

Following the previous sections, it is of extreme importance to also have the infrastructure protected against external unwanted intrusions. Concretely, further research needs to be done into the cyber security field in order to provide for cover on the already defined systems. Furthermore, legal cover should be prepared for such a scenario with ready to implement practices available should such cases arise. Stress tests should be also taken against the systems and further proof of stability should be obtained before the system is even allowed to go live in order to provide the purchasing customers the much required seals of quality.

7.4.2 Feature Expansion

In this section we will be analysing the features that could be potentially considered for this thesis. This features have not been considered for the thesis due to them not being any of the base or extended requirements provided by the stakeholders and therefore are only considered as expanding the value of the product.

Custom Base System Integration

In this section we will be discussing multiple approaches to enhance the collaboration between the deployed work and the base system that will serve as a platform for its base workspace. Concretely, we will be analysing the possible interaction possibilities with a custom made base system, that is, a customized base Operating System for our deployed system thus reducing the amount of extra infrastructure that is not going to be used and therefore enhancing performance itself.

Keeping with Open Source This may seem obvious but it must be empathized when considering a custom built system. As such, the idea of building a custom one totally integrated with the work might occur, but this option would mean redoing a huge amount of already done work on open source systems that would also imply higher maintenance costs.

Reduce Infrastructure Overload One of the key advantages of using this approach is the ability to reduce to a maximum extent the extra components on our base system therefore giving an immediate boost on the productivity and performance of it. To a major extent, it also reduces security complications by removing the use of unneeded components which in turn might have security holes.

Feature Extension By using a smart selection of the base software we can avoid developing extra features by using the ones embedded into the system. Using this approach we can reduce to a minimum the integration part by reducing it to a simple configuration task of the system. Altogether with this, we would ensure ourselves of keeping up to date with updated software due to it being external, and open source so suppressing any need for extra licensing and costs.

Plugin Functionality Extension

In this section we will be analysing the extent on immediate plugin functionality extension. These extensions have been foreseen while building this work but have deemed as non essential due to them not being part of the main specification. Following we will provide a briefing on the three main expansion points that could be foreseen as the most probable path for future development to take place into.

Plugin Suspension Currently plugins are being executed on a full basis when on the background, allowing them to run in the background in an uncontrolled manner on the current planned version. This may lead to performance issues but it has been considered that the architectural complexity of such approach might be totally out of scope for this product. Therefore, is set as part of the tasks to be done on continuing this thesis to further analyse and propose a solution to this problem thus allowing the system to control the flow of the plugins being executed on a full basis.

Plugin side-by-side In an approach closer to the current tablet-based operating systems the possibility to run more than one application side-by-side has been proposed. This way, a user could be using two main functionalities at once without having to change between these plugins. This, of course, adds a level of complexity to the architecture that is out of scope for the requirements associated with this thesis and therefore has been deemed as part of the immediate future work package associated with this analysis.

Resource management One of the key flaws of this work is the lack of planned resource management, such as the storage or percentage utilization. These features need to be considered and embedded into the architecture as they are of maximum privileges and therefore can't be delegated to other layers. The problem regarding with this is the fact that currently there are no viable embedded options to handle this task, and therefore a further analysis on it should be made as it is out of scope for this thesis. Therefore, a possible work path for the future is to provide a viable outcome on this kind of management visualization and handling.

7.4.3 Alternatives to current Hardware Approach

There has been an extensive research on alternatives to the current Hardware Approach by other parties. Concretely, they propose two different approaches that should be considered before starting the implementation of the solution.

3D Interaction

This would mean that the users would be able to see the infotainment product with in a 3D stereoscopic way, therefore enhancing the experience of them by providing a surrounding experience. Although this feature provides a great bonus to any product, further security and usage constraints should be considered. [BAS12]

Hands-free Interaction

In this case we would be tackling interaction in a totally different approach. By removing the touch contact we would remove one of the highest security issues when interacting with touch devices while in-driving, as seen in the regulations analysis. This approach could be either way by providing audio-based [FLS12] or gesture-based interaction [OBTT12], but further analysis into these solutions should be done for this product.

7.5 Performance Tests

In this section we will be discussing to a greater extent the necessary tests related with the development of this product along with further scalability tests in order to ensure the functionality of the deployed architecture under the most likely different loads that are going to arise under its usage.

7.5.1 Architecture Tests

In this section we will be covering the necessity for a posterior test design and implementation for the deployed architecture. The analysed system has already been designed with a huge usage in mind, but further tests will be required in order to check on whether these premises are right and further corrective action will be in need of appliance in case they are not correct.

Common Performance Tests

There are a common set of tests that have become standardized under a section professionally known as "Performance Engineering". We are not going to go into each of these tests as these are out of scope for this section. The task that should be performed on a future basis is an analysis of them and a selected implementation of them based on our performance needs. [Mic13b]

Denial of Service (DDoS) attacks

In this section we will be analysing the necessity of our system to provide constant protection against such type of large scale attacks against our infrastructure. The deployed architecture needs to be ready to withstand and counter-attack such attacks while protecting the innermost data and providing normal usage to the clients. We can see the high level of difficulty related with this task in other successful attempts to steal data from clients using this fashion, which in turn have proved to be of a huge negative impact on the user experience. [She12]

7.5.2 Local Infrastructure Tests

The local infrastructure represents those having to do with the deployed system itself. This needs to be tested in order to provide for the maximum user usage under normal stress conditions, and if necessary further corrective action in regards of technology or hardware limitations should be taken into account. This is extremely necessary if the system ever aims to be deployed for a large amount of users using the same centralized unit, such as in multi person transports.

Infrastructure Security

Our major security directives have been oriented towards ensuring that the projected work is safe and complying with the currently available and standardized security directives. Due to the constrains on which this thesis is based, it was always assumed that the Infrastructure on which this work would be deployed would itself be fool-proof, tested and designed to withstand the most common security threats for the architecture to be deployed on the development of this work. Therefore it is obvious that a major aspect on Security has been omitted, certainly because the product has been designed on an architecture independent basis. To workaround this aspect and ensure that it will be safe enough to not commit a natural prejudice against the deployed work the fact of ensuring this security before release of a platform should be done. To this extent an out-of-scope analysis on the security requisites and development or selection of standards and tests should be carried out against the infrastructure.

Infrastructure Pass-through handling

The infrastructure will therefore also be providing external devices access to the Internet should the deployed user allow so. This situation would most probably occur under huge system deployments, such as those deployed in public transport or on big personnel transport scenarios. For these cases, certain stress tests need to be carried out in order to ensure that it will be able to withstand the demand without affecting the stability of the system. If such situations were to happen a future work on analysing the root causes and finding a workaround for them must be done, task which is also out of scope for this work due to it needing an already built system.

7.6 User Experience Tests

As any other kind of User oriented application a huge emphasis has been taken into the development of a valid User Experience. This core requirement of our work should be transferred with equal priority and correctness to the developed work. In order to ensure that, a set of tests are also going to be undertaken on the developed system that will punctuate the validity of the implementation regarding the desired output. These tasks are of extreme importance if the product has to be ultimately successful, and that's the main justification on the specification of them in this section.

Figure 7.1: In a future scope, large amounts of users may be using one system (caption: Lufthansa Board-Connect [Mar13])

7.6.1 Web Consistency Testing

Due to the genericness of the target application that this work defines, it has come to a necessity the fact that consistency across multiple devices must be assured. We consider this fact of such criticality that we deem as required for the future development of the application to have web consistency tests made along it. This tests will ensure that the developed application works on the same fashion in all the devices that are deemed as supported, and to and end will ultimately define this much needed list. Following the rules will allow for the application to have the same behaviour independently on the device the user is using as long as it is supported by the work and therefore the same look and feel. [Men11]

7.6.2 Functional Testing

This kind of testing will provide an insight on the correct work procedures developed. Most software nowadays has a key lack on correct testing and therefore should not be available for end user deployment due to high probability of software failure. In order to prevent this, functional testing must be applied to the product development already on early stages. This will consist on the identification of the tasks that the software is expected to perform and analyses on its

behaviour for multiple running cases. The correct appliance of this directive will imply a fewer likeness of deployment errors, as well as highly simplify the procurement of the much needed quality awards that will surely impact on the final decision from manufacturers and customers on the product usage. [Kan99]

7.6.3 Usability Testing

That is, user-based testing on a certain product. In this case, we would require potential users of our application to test its functionality in order to orient it to the maximum level of productivity and best user experience that could be achieved. To achieve this, we would be coupling with the partners that have been defined in a previous section and use their feedback while on development to improve, enhance and modify or add functionalities that might simplify and ease the every day usage of the application. This kind of testing is therefore the one that will give the best indication on whether the product is able for publishing and as such is the one that has the biggest impact in the outcome of these tests and overviews. [Nie94]

8 Resume

After the work done in this thesis we have concluded that it is feasible to create a new and innovative system able to replace the current ones and provide a feasible solution to the problems presented by the current products on the markets. With this thesis itself we have been able to present a unified architecture capable to bring a substantial innovation to a sector that has not yet adapted to the current state of the art technologies on the world. We achieved the compilation of the latests innovations on the fastest-evolving IT sectors, the mobile and web world, within a solution that will also solve the problems present on any HMI design for automotive environments. The platform design that we have presented is still on its early stages and will therefore need to be further improved and consulted with sector experts, but still this fact we firmly believe that is able to become a successful prototype on its current status. Taking into consideration the multiple hardware constrains of what used to be a closed environment, we have been able to provide the platform with a regenerative environment via the installation of complements and updates that will allow the user to customize the HMI to its personal taste and avoid having to live with software flaws. Altogether, this model allows for a source of income after selling the main automotive environment, which envisages a better income scenario for manufacturers altogether. The objectives and the requirements have been fulfilled with a satisfactory result, even exceeding the expected. To this extent, we have jumped from a architecture that only presented the necessity to allow for applications to be purchased to a full environment that allows also for language customisations and full user interface revamps with the same simplicity. In the current state of the art there is no product like the one proposed on this work, and therefore this has become the alternative to a market where each developer seeks frontal confrontation with each other therefore rendering the user experience as a negligible asset, practise that at the end makes the user unaware of the different features each HMI system has. Although this affirmation, we can clearly observe a general shift on manufacturer policy that indicates that our approach will be the main developing line for the next-generation HMI systems and therefore confirming and justifying the arguments proposed on this thesis.

This thesis itself has brought me to know a world that was previously closed to me as is the world of HMI on automotive environments. I had previously shown limited interest in this kind of architectures, but with this thesis I have been able to discover a whole world that could certainly offer high perspectives.

Bibliography

[AB13] MoSync AB. Mosync sdk - native mobile app development for multiple platforms using a single code base, September 2013.

[AFG+10] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia. A view of cloud computing. *ACM Computing Surveys Vol. 53, No 4,*, April 2010.

[AKS+10] Florian Alt, Dagmar Kern, Fabian Schulte, Bastian Pfleging, Alireza Sahami Shirazi, and Albrecht Schmidt. Enabling micro-entertainment in vehicles based on context information. *AutomotiveUI '10 Proceedings of the 2th International Conference on Automotive User Interfaces and Interactive Vehicular Applications*, 11-12 November 2010.

[AM11] Jason Ansel and Petr Marchenko. Language-independent sandboxing of just-in-time compilation and self-modifying code. *ACM*, 4-8 June 2011.

[App13a] Appcelerator. Appcelerator platform, September 2013.

[App13b] AppUrl. Registering and using a url scheme in android, September 2013.

[ATT13] ATT. Att developer apis, September 2013.

[Aud13] Audi. Audi connect brochure, September 2013.

[Aut13] AutoTrader.com. Infotainment systems: A comparison, September 2013.

[BAS12] Nora Broy, Elisabeth André, and Albrecht Schmidt. Is stereoscopic 3d a better choice for information representation in the car? *4th International Conference on Automotive User Interfaces and Interactive Vehicular Applications*, 17-19 October 2012.

[BBP10] Raja Bose, Jörg Brakensiek, and Keun-Young Park. Terminal mode – transforming mobile devices into automotive application platforms. *AutomotiveUI '10 Proceedings of the 2th International Conference on Automotive User Interfaces and Interactive Vehicular Applications*, 11-12 November 2010.

[BHS+10] Prithvi Bisht, Timothy Hinrichs, Nazari Skrupsky, Radoslaw Bobrowicz, and V.N. Venkatakrishnan. Notamper: Automatic blackbox detection of parameter tampering opportunities in web applications, 4-8 October 2010.

[Bou05] A. Boulanger. Open-source versus proprietary software: Is one more reliable and secure than the other? *IBM Systems Journal*, Vol 44, No 2, 2005.

[bwi09] bwired. Open source vs. closed source (proprietary) software, 2009.

[Car11] Mathieu Carbou. Reverse ajax, part 1: Introduction to comet, 19 July 2011.

[Che13] Chevrolet. Discover chevrolet mylink, September 2013.

[Com13a] Ford Motor Company. *MyFord Touch User Guide*, September 2013.

[Com13b] MIT Mobile Computing. Mit mobile web framework, September 2013.

[Con11] WWW Consortium. Cascading style sheets (css) specification, 12 May 2011.

[Con12] WWW Consortium. Xmlhttprequest specification, 6 December 2012.

[Con13] WWW Consortium. Html5 draft specification, 6 August 2013.

[dMHW+09] Guido de Melo, Frank Honold, Michael Weber, Mark Poguntke, and André Berton. Towards a flexible ui model for automotive human-machine interaction. *AutomotiveUI '09 Proceedings of the 1st International Conference on Automotive User Interfaces and Interactive Vehicular Applications*, 21-22 September 2009.

[EAGK03] Patrick Th. Eugster, Pascal A.Felber, Rachid Guerraoui, and Anne-Marie Kermarrec. The many faces of publish/subscribe. *ACM Computing Surveys*, Vol. 35, No. 2, June 2003.

[Erk12] Jussi-Pekka Erkkilä. Websocket security analysis, Autumn 2012.

[Fal13] Alborz Fallah. New mazda 3 future-proofs infotainment system, 12 July 2013.

[Fav13] Loie Favre. Sony xperia ui vs. stock android: Comparing manufacturer-branded roms, September 2013.

[FBH+11] Peter Fröhlich, Matthias Baldauf, Marion Hagen, Stefan Suette, Dietmar Schabus, and Andrew L. Kun. Investigating safety services on the motorway: the role of realistic visualization. *AutomotiveUI '11 Proceedings of the 3rd International Conference on Automotive User Interfaces and Interactive Vehicular Applications*, 1-2 December 2011.

[FLS12] Johan Fagerlönn, Stefan Lindberg, and Anna Sirkka. Graded auditory warnings during in-vehicle use: Using sound to guide drivers without additional noise. *4th International Conference on Automotive User Interfaces and Interactive Vehicular Applications*, 17-19 October 2012.

[Fou04] The Apache Software Foundation. Apache license, version 2.0, January 2004.

[Fou13a] The Apache Software Foundation. About apache cordova™, September 2013.

[Fou13b] The Dojo Foundation. Dojo framework backers, September 2013.

[Fou13c] The Dojo Foundation. Dojo mobile, September 2013.

[Fur10] Y. Furukawa, editor. *Web-Based Control Application using WebSocket*, 2010.

[Gen13] Genivi. Genivi alliance, open-source infotainment, September 2013.

[GHHW10] Andrew W. Gellatly, Cody Hansen, Matthew Highstrom, and John P. Weiss. Journey: General motors' move to incorporate contextual design into its next generation of automotive hmi designs. *AutomotiveUI '10 Proceedings of the 2th International Conference on Automotive User Interfaces and Interactive Vehicular Applications*, 11-12 November 2010.

[Gro13] The PHP Group. Php reference (manual) v5.3, August 2013.

[HG13] Ian Hickso and Inc. Google. The websocket api, 26 July 2013.

[HGM+12] Steffen Hess, Anne Gross, Andreas Maier, Marius Orfgen, and Gerrit Meixner. Standardizing model-based in-vehicle infotainment development in the german automotive industry. *4th International Conference on Automotive User Interfaces and Interactive Vehicular Applications*, 17-19 October 2012.

[HGmWH96] Cheng-Hsueh A. Hsieh, John C. Gyllenhaal, and Wen mei W. Hwu. Java bytecode to native code translation: The caffeine prototype and preliminary results. *Proceedings of the 29th Annual International Symposium on Microarchitecture*, 2-4 December 1996.

[HJC10] Daz L. Hibberd, Samantha L. Jamson, and Oliver M. J. Carsten. Managing in-vehicle distractions - evidence from the psychological refractory period paradigm. *AutomotiveUI '10 Proceedings of the 2th International Conference on Automotive User Interfaces and Interactive Vehicular Applications*, 11-12 November 2010.

[Ido] Robert David Idol. Looking beyond http: The future of web application protocols.

[IL13] HK Innotrends Ltd. Ca-fi, android infotainment system, September 2013.

[iN13] iTers News. Tesla motors unwraps virtual ui-centric car infotainment system, 9 January 2013.

[Inc12] Telenav Inc. Scout™ for apps, telenav's html5 voice-guided gps navigation service, 27 March 2012.

[Inc13a] Amazon Web Services Inc. Amazon web services, September 2013.

[Inc13b] Amazon Web Services Inc. Amazon web services security center, September 2013.

[Inc13c] Corona Labs Inc. Corona sdk, September 2013.

[Inc13d] Google Inc. Android, the world's most popular mobile platform, September 2013.

[Inc13e] Google Inc. Google web toolkit, September 2013.

[Inc13f] Kia Motors America Inc. Kia 2013 warranty manual, September 2013.

[Inc13g] Modo Labs Inc. The technology behind kurogo, September 2013.

[Inc13h] Sencha Inc. Sencha touch, September 2013.

[Ini13] Open Source Initiative. The mit license (mit), September 2013.

[Int11] Ecma International. Ecmascript language specification, June 2011.

[Jay13] Nick Jaynes. Will outdated regulations force the u.s. to ban complex in-dash tech systems?, 16 August 2013.

[Jeo10] Myounghoon Jeon. "i-passion": A concept car user interface case study from the perspective of user experience design. *AutomotiveUI '10 Proceedings of the 2th International Conference on Automotive User Interfaces and Interactive Vehicular Applications*, 11-12 November 2010.

[jF13] The jQuery Foundation. A touch-optimized web framework, September 2013.

[JRL12] Myounghoon Jeon, Andreas Riener, and Ju-Hwan Lee. Cross-cultural differences in the use of in-vehicle technologies and vehicle area network services: Austria, usa, and south korea. *AutomotiveUI '12 Proceedings of the 4th International Conference on Automotive User Interfaces and Interactive Vehicular Applications*, 17-19 October 2012.

[Kan99] Nguyen Kaner, Falk. *Testing Computer Software*. Wiley Computer Publishing, 1999.

[KPeM+09] Andrew L. Kun, Tim Paek, Željko Medenica, Nemanja Memarović, and Oskar Palinko. Glancing at personal navigation devices can affect driving: experimental results and design implications. *AutomotiveUI '09 Proceedings of the 1st International Conference on Automotive User Interfaces and Interactive Vehicular Applications*, 21-22 September 2009.

[Kre13] Matthaeus Krenn. A new car ui, September 2013.

[KSL13] Tuomo Kujala, Johanna Silvennoinen, and Annegret Lasch. Visual-manual in-car tasks decomposed: text entry and kinetic scrolling as the main sources of visual distraction. *AutomotiveUI '13 Proceedings of the 5th International Conference on Automotive User Interfaces and Interactive Vehicular Applications*, 8 September 2013.

[Kum12] Awdhesh Kumar. Html5 websockets and coldfusion – part 1: An overview and first steps, 21 May 2012.

[Lai13] Cameron Laird. The dangers of html5: Websockets and stable standards, September 2013.

[LC13] Peter Lubbers and Frank Greco (Kaazing Corporation). Html5 web sockets: A quantum leap in scalability for the web, September 2013.

[LCDna] Gen Lu, Kevin Coogan, , and Saumya Debray. Automatic simplification of obfuscated javascript code, Department of Computer Science, The University of Arizona.

[LDS12] Joonbum Lee, John D.Lee, and Dario D. Salvucci. Evaluating the distraction potential of connected vehicles. *4th International Conference on Automotive User Interfaces and Interactive Vehicular Applications*, 17-19 October 2012.

[Lee13] Niels Leenheer. Html 5 test, September 2013.

[LK12] Annegret Lasch and Tuomo Kujala. Designing browsing for in-car music player - effects of touch screen scrolling techniques, items per page and screen orientation on driver distraction. *4th International Conference on Automotive User Interfaces and Interactive Vehicular Applications*, 17-19 October 2012.

[LLC12] Bergmans Mechatronics LLC. Html5 websocket application research, 2012.

[LLC13] Barnesandnoble.com LLC. Why nook?, September 2013.

[LS88] Barbara Liskov and Liuba Shrira. Promises: Linguistic support for efficient asynchronous procedure calls in distributed systems, 22-24 June 1988.

[Ltd13] BuiltWith® Pty Ltd. Javascript usage statistics, September 2013.

[Mar13] Peter Marwan. Lufthansa system introduces advanced concept for in-flight entertainment, 9 April 2013.

[McC13] Harry McCracken. Who's winning, ios or android? all the numbers, all in one place, April 16, 2013.

[McG13] Shane McGlaun. Volvo and neonode team up for sensus connected touch infotainment system, 12 March 2013.

[MCS+11] Sachi Mizobuchi, Mark Chignell, Junko Suzuki, Ko Koga Toyota, and Kazunari Nawa. Central executive functions likely mediate the impact of device operation when driving. *AutomotiveUI '11 Proceedings of the 3rd International Conference on Automotive User Interfaces and Interactive Vehicular Applications*, 1-2 December 2011.

[Men11] Kevin Menard. Web consistency testing, 2011.

[Mic03] Microsoft. Overview of ssl/tls encryption, 31 July 2003.

[Mic13a] Microsoft. Auto-launching apps using file and uri associations for windows phone 8, September 2013.

[Mic13b] Microsoft. Performance engineering for web applications, September 2013.

[Mic13c] Microsoft. The publish/subscribe pattern (project silk documentation, chapter 8), September 2013.

[Mot13] General Motors. General motors in-vehicle infotainment vision, September 2013.

[MozPM] Mozilla. Websockets, Apr 17, 2013 2:10:36 PM.

[MRS09] Ivan Magdalenic, Danijel Radosevic, and Zoran Skocir. Dynamic generation of web services for data retrieval using ontology. *Informatica*, Vol.20, No. 3, 2009.

[MS13] Inc. Motorola Solutions. Rhomobile suite, September 2013.

[MWG+11] Alexander Meschtscherjakov, David Wilfinger, Nicole Gridling, Katja Neureiter, and Manfred Tscheligi. Capture the car!: qualitative in-situ methods to grasp the automotive context. *AutomotiveUI '11 Proceedings of the 3rd International Conference on Automotive User Interfaces and Interactive Vehicular Applications*, 1-2 December 2011.

[NDEK09] Bernhard Niedermaier, Stephan Durach, Lutz Eckstein, and Andreas Keinath. The new bmw idrive – applied processes and methods to assure high usability. *Second International Conference, ICDHM 2009, Held as Part of HCI International 2009, San Diego, CA, USA, July 19-24, 2009. Proceedings*, 19-24 July 2009.

[New12] Telematic News. Kia showcases infotainment concept with app store, 14 March 2012.

[NH13] Thomas Nolte and Hans Hansson. Modeling and analysis of message-queues in multi-tasking systems, August 2013.

[Nie94] J Nielsen. *Usability Engineering*. Academic Press, 1994.

[OBTT12] Eshed Ohn-Bar, Cuong Tran, and Mohan Trivedi. Hand gesture-based visual user interface for infotainment. *4th International Conference on Automotive User Interfaces and Interactive Vehicular Applications*, 17-19 October 2012.

[otUoC13] Regents of the University of California. The bsd 3-clause license, September 2013.

[Oyj13] Digia Oyj. Qt 5.1 specification, September 2013.

[Par13] Parrot. Asteroid range, September 2013.

[Pho13] PhoneGap. What is the difference between phonegap and cordova? (phonegap faq), September 2013.

[Pro13] RACE Project. Race project - about us, July 2013.

[PSDK11] Bastian Pfleging, Albrecht Schmidt, Tanja Döring, and Martin Knobel. Autonui: a workshop on automotive natural user interfaces. *AutomotiveUI '11 Proceedings of the 3rd International Conference on Automotive User Interfaces and Interactive Vehicular Applications*, 1-2 December 2011.

[RBA13] Matthew Russell, David J. Buchko, and Julian Arguelles. Bmw connecteddrive: Broaden of access and expansion of services globally will include benefits for us customers, 5 June 2013.

[Rea13] RealVNC. Vnc automotive, September 2013.

[Rec13] Mark Rechtin. Average age of u.s. car, light truck on road hits record 11.4 years, polk says, 8 August 2013.

[RFB+13] A. Riener, A. Ferscha, F. Bachmair, P. Hagmüller, A. Lemme, D. Muttenthaler, D. Pühringer, H. Rogner, A. Tappe, and F. Weger. Standardization of the in-car gesture interaction space. *AutomotiveUI '13 Proceedings of the 5th International Conference on Automotive User Interfaces and Interactive Vehicular Applications*, 8 September 2013.

[RJL12] Shannon C. Roberts, William J.Horrey, and Yulan Liang. Effect of performance feedback (or lack thereof) on driver calibration. *4th International Conference on Automotive User Interfaces and Interactive Vehicular Applications*, 17-19 October 2012.

[RLH11] Barbara Rosario, Kent Lyons, and Jennifer Healey. A dynamic content summarization system for opportunistic driver infotainment. *AutomotiveUI '11 Proceedings of the 3rd International Conference on Automotive User Interfaces and Interactive Vehicular*, 1-2 December 2011.

[RMC13] Luis Rivero, Sabrina Marczak, and Tayana Conte. An approach for the elicitation of usability requirements in the development of web applications. February 2013.

[Ros05] Lawrence Rosen. Academic free license ("afl") v. 3.0, 2005.

[RS09a] Andry Rakotonirainy and Steinhardt. In-vehicle technology functional requirements for older drivers. *AutomotiveUI '09 Proceedings of the 1st International Conference on Automotive User Interfaces and Interactive Vehicular Applications*, 21-22 September 2009.

[RS09b] Andry Rakotonirainy and Dale Steinhardt. In-vehicle technology functional requirements for older drivers. *AutomotiveUI '09 Proceedings of the 1st International Conference on Automotive User Interfaces and Interactive Vehicular Applications*, 21-22 September 2009.

[Rub13] Adam Rubenfire. Group sues ford, claims touchscreen systems defective, July 17, 2013.

[SAA+11] Helena Strömberg, Pontus Andersson, Susanne Almgren, Johan Ericsson, Mari-Anne Karlsson, and Arne Nåbo. Driver interfaces for electric vehicles. *AutomotiveUI '11 Proceedings of the 3rd International Conference on Automotive User Interfaces and Interactive Vehicular Applications*, 1-2 December 2011.

[SFG99] Helen Sharp, Anthony Finkelstein, and Galal Galal. Stakeholder identification in the requirements engineering process, 11/12/1999.

[She12] Sam Shead. Symantec: Data-stealing hackers use ddos to distract from attacks, October 2012.

[Soc99] Network Working Group (The Internet Society). Hypertext transfer protocol – http/1.1 (rfc2616), 1999.

[Soc13] SockJs. Sockjs-client, June 2013.

[Son10] Jan Sonnenberg. Service and user interface transfer from nomadic devices to car infotainment systems. *AutomotiveUI '10 Proceedings of the 2th International Conference on Automotive User Interfaces and Interactive Vehicular Applications*, 11-12 November 2010.

[SPB+13] Stefan Schneegass, Bastian Pfleging, Nora Broy, Frederik Heinrich, and Albrecht Schmidt. A data set of real world driving to assess driver workload. *AutomotiveUI '13 Proceedings of the 5th International Conference on Automotive User Interfaces and Interactive Vehicular Applications*, 8 September 2013.

[SRF12] Ronald Schroeter, Andry Rakotonirainy, and Marcus Foth. The social car: New interactive vehicular applications derived from social media and urban informatics. *AutomotiveUI '12 Proceedings of the 4th International Conference on Automotive User Interfaces and Interactive Vehicular Applications*, 17-19 October 2012.

[Sys07] GNU Operating System. Gnu lesser general public license, 29 June 2007.

[Tea13] Webkit Team. Webkit sunspider javascript test, September 2013.

[Tec13a] TechEmpower. Web framework benchmarks, 2 July 2013.

[Tec13b] Unity Technologies. Unity3d showcase, September 2013.

[TGH+11] Nora Broy Technische, Sebastian Goebl, Matheus Hauder, Thomas Kothmayr, Michael Kugler, Florian Reinhart, Martin Salfer, Kevin Schlieper, and Elisabeth André. A cooperative in-car game for heterogeneous players. *AutomotiveUI '11 Proceedings of the 3rd International Conference on Automotive User Interfaces and Interactive Vehicular Applications*, 1-2 December 2011.

[TN00] Hans Hansson Thomas Nolte. Modeling and analysis of message-queues in multitasking systems, 2000.

[TSG04] Inc. The SCO Group. How rpc works, 27 April 2004.

[TSZ+11] Sergej Truschin, Tobias Schlachtbauer, Andreas Zauner, Michael Schermann, and Helmut Krcmar. Content matters: towards handling e-mail while driving safely.

AutomotiveUI '11 Proceedings of the 3rd Int6ernational Conference on Automotive User Interfaces and Interactive Vehicular Applications, 1-2 December 2011.

[Val11] Jeferson Valadares. ree-to-play revenue overtakes premium revenue in the app store, July 07, 2011.

[Vol13] Volvo. *SenSuS 3.0 Seven Quick Start Guide*, 27 June 2013.

[WCN+13] Jibo He Wichita, Alex Chaparro, Bobby Nguyen, Rondell Burge, Joseph Crandall, Barbara Chaparro, Rui Ni, and Shi Cao. Texting while driving: is speech-based texting less risky than handheld texting? *AutomotiveUI '13 Proceedings of the 5th International Conference on Automotive User Interfaces and Interactive Vehicular Applications*, 8 September 2013.

[Wes13] Linda Westfall. Software requirements engineering: What, why, who, when, and how, September 2013.

[Wik13] Wikipedia. Multiple phone web-based application framework, 6 August 2013.