CHALMERS

Market-Driven Framework for Guiding Optimisation Decisions in Embedded Systems

Master of Science Thesis

in the Software Engineering Programme

Sofia Charalampidou
Paschalis Tsolakidis

Chalmers University of Technology
University of Gothenburg
Department of Computer Science and Engineering
Göteborg, Sweden, August 2013

Market-Driven Framework for Guiding Optimisation Decisions in Embedded Systems

Sofia Charalampidou, Paschalis Tsolakidis

Examiner: Richard Torkar
Supervisors: Christian Berger (Chalmers), Tobjörn Mattsson (Mecel AB)

Chalmers University of Technology
University of Gothenburg
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Department of Computer Science and Engineering
Göteborg, Sweden August 2013

Market Driven Framework for Guiding Optimisation Decisions in Embedded Systems

CHALMERS UNIVERSITY OF TECHNOLOGY
Department of Computer Science and Engineering

Abstract

The recent need for web connectivity in the embedded systems domain and in particular the In-Vehicle Infotainment (IVI), has fired discussions about the integration of HTML components in these systems. This need derives from the growing market demands that could be met by more dynamic IVI systems. In this thesis we present a methodological framework that connects the different stakeholders' interests, for different software quality characteristics of a product, with the optimisation decisions taken during the product development. We validate the proposed framework with an industrial example at Mecel AB, where we extend Mecel's Populus, an industrial HMI tool suite, with an HTML component that uses the WebKit rendering engine. The purpose of this extension is to enable Populus to render HTML content which is already available through the Internet or content which can be developed and saved locally as HTML pages. In both cases the benefit is the quick and easy access to new content. After extending the product we apply a market driven approach for selecting the optimisation methods and techniques that could be used to enhance the software quality of the integrated product, based on the market needs. We implemented 3 optimisation techniques and we measured the framerate and the changes in lines of code, which were the identified metrics connected with the most important software quality attributes according to the stakeholders. The proposed methodological framework aims to aid developers in similar domains in making market driven optimisation decisions.

Acknowledgements

Apart from our own effort, the success of this project depends largely on the encouragement and guidelines of many others. We take this opportunity to express our greatest appreciation to the people who have been instrumental in the successful completion of this project. Foremost, we would like to express our gratitude to our Chalmers supervisor Christian Berger and our Mecel supervisor Torbjörn Mattsson for their useful comments, remarks and patience through the learning process of this master thesis.

Furthermore we would like to thank Mecel AB and all those who believed in us and gave us the opportunity to work with this thesis topic. Especially we would like to thank Mikael Uddberg for providing us all comforts and support regarding the administrative aspects throughout our stay at Mecel, as well as the whole Populus team for the warm welcome and the pleasant collaboration. Also, we would like thank the participants in our case study, Anders Eliasson, Ross Fitkin and Torbjörn Mattsson, who willingly shared their precious time and expertise during the process of interviewing and provided us with iterative feedback enhancing the quality of our study.

Finally we can't say thank you enough to our families, who have supported us throughout the entire process, both financially and by keeping us harmonious, being by our side through thick and thin.

Contents

List of Figures

List of Tables

Abbreviations

ARM	**A**corn **R**ISC **M**achine
B2B	**B**usiness **To B**usiness
B2C	**B**usiness **To C**ustomers
CPU	**C**entral **P**rocessing **U**nit
CSS	**C**ascading **S**tyle **S**heets
DirectFB	**Direct F**rame **B**uffer
GPU	**G**raphics **P**rocessing **U**nit
GQM	**G**oal **Q**uestion **M**etrics
GTK	**G**IMP **T**ool**K**it
HCV	**H**ierarchical **C**umulative **V**oting
HMI	**H**uman **M**achine **I**nterface
HTML	**H**yper **T**ext **M**arkup **L**anguage
IVI	**I**n **V**ehicle **I**nfotainment
LOC	**L**ines **O**f **C**ode
NFR	**N**on **F**unctional **R**equirements
ODI	**O**pen **D**isplay **I**nterface
OEM	**O**riginal **E**quipment **M**anufacturer
PPDM	**P**roduct **P**rocess **D**ependencies **M**odel
RAM	**R**equirements **A**bstraction **M**odel
QFD	**Q**uality **F**unctional **D**eployment
QUPER	**QU**ality **PER**formance
SAAM-SQ	**S**takeholder **A**lignment **A**ssessment **M**ethod for **S**oftware **Q**uality
XML	E**X**tensible **M**arkup **L**anguage

Chapter 1

Introduction

The software development process is usually influenced by factors like the market needs and the stakeholders' objectives. Thus it is considered important to gain a clear understanding about their needs and goals increasing the chances of the product to satisfy their expectations [KS07]. In our case study we are dealing with the case of developing and later on optimising a new feature for a product in the domain of the In-Vehicle Infotainment (IVI) embedded systems. Thus we are interested in identifying the several stakeholders' needs in order to proceed with the optimisation of the product accordingly. This task can often be challenging because of the large number of stakeholders to elicit requirements from and the large number of requirements that they have, with varying level of detail and quality [BD07]. For this reason we have selected to develop a market-driven framework for capturing the optimisation needs of the stakeholders.

In our case study we worked on the example of an IVI system that we wanted to extend with a new HyperText Markup Language (HTML) rendering functionality. Generally the literature captures the tendency for web connectivity in the embedded systems domain. Nowadays many mobile devices, home entertainment systems, and IVI systems are connected to the web. Most of them have integrated web browsers (rendering engines) [IGB12]. Currently the expectations of multimedia content delivery methods grow rapidly [GMM+11], thus browsers are the preferred choice for an application runtime environment [IGB12]. Such applications as well as the need for high display quality and improved processing power have introduced a need for hardware graphics accelerators optimised for mobile devices [Ols08]. For this reason many Human Machine Interfaces (HMIs) for embedded systems, ranging from smart-phones to TVs, are implemented in HTML5. Standards like HTML5 and WebGL are used to provide high performance

platforms for interactive graphics applications on most mobile devices [GMM+11, MF12].

Although there are many similarities in all the aforementioned embedded systems different application domains require special handling. For example, as [IGB12] explains, on many embedded systems the applications consume little to no resources when they are put in the background. However in IVI systems the processes and applications are assigned a predefined amount of resources to ensure the functionality of the basic features in every occasion. A browser in such a system has to be functional in this restricted environment. Thus it is obvious that in such cases further optimisation and hardware acceleration techniques are important to ensure an appealing user experience.

1.1 The Problem

The problem which triggered this study is the need for implementation and optimisation of an HTML rendering engine in a graphical user interface for ARM-based (Acorn RISC Machine) and x86 platforms. Specifically for the needs of the study the Mecel Populus Suite will be used, which is a product for developing graphical user interfaces for embedded systems of the automotive market. As already mentioned the embedded devices have shown a tendency to connect to the web and Populus is one of them in the IVI domain. The next step to achieve this need for connectivity is the integration of these systems with web browsers. Such an integration would add value to the product increasing its competence in the automotive domain.

The outcome of this integration will be focused in specific software quality attributes (e.g. portability and performance), which will be introduced in the form of a framework through an investigation both on the market needs and the experience of the software engineers working with the product. The goal of this framework will be to investigate the most important quality attributes that should be optimised based on the captured market needs. The study will be of high industrial interest for Mecel but will also stand as a proof of concept for the optimisation possibilities of rendering engines in the IVI domain, using the introduced framework.

1.2 Research Questions

The focus of this thesis is to present a framework for optimising embedded systems according to the market needs. Thus the major goals are to capture the stakeholders' needs and understand

how they affect the product integration and optimisation decisions. These goals are stated in the following research questions.

- RQ1: How can market needs be used to identify important software quality attributes?
- RQ2: How can the optimisation process of a product be driven by the software quality attributes from RQ1?

1.3 Outline

In Chapter 2 we provide the background information on Mecel's tools, tools for extending Populus with a web rendering engine, and generally an introduction to background terms which will be used as part of this thesis work. Chapter 3, provides details on the implementation of the product that we based our case study on. The thesis work required the integration of the Populus tool with a web rendering engine, that afterwards would be optimised based on our framework. So Chapter 3 presents information on the implementation process that was a prerequisite for our study. In Chapter 4 we present the framework that we suggest and we describe in details the methodology that has been followed. Chapter 5 presents the results of our case study as well as an analysis of these findings. Then in Chapter 6, we provide some related studies, explaining their similarities and differences with our work. The discussion of the paper (see Chapter 7) is divided in subsections, according to the research goals, while limitations and recommendations are also discussed. Finally in Chapter 8 we sum up the findings of our study and in Chapter 9 we make proposals for future work.

Chapter 2

Background

In this section we are going to introduce the relevant tools and the technological background for understanding this thesis' content. Due to the fact that the thesis was conducted at Mecel, Mecel's Populus Suite has been used and is introduced in this section. Additionally we provide information about rendering engines which are used in common web browsers. Moreover we define the word framework as used in our approach and we introduce the concept of market-driven development. Finally we present information about the ISO Quality Model which was used in our framework.

2.1 Mecel's Populus Suite

Mecel Populus Suite is a complete tool chain for designing, developing and deploying user interfaces for distributed embedded systems. It consists of Populus Editor, which is used to create an HMI (layout and logic) and store it in the HMI database, and Populus Engine, which renders the contents of the database and shows the outcome on the cluster or screen of the vehicle. The engine itself can be connected with other applications, both internal and external, through the Open Display Interface (ODI). The ODI is a communication protocol that allows applications to use the vehicle displays through the Populus Engine. [Mec12]

2.2 Rendering Engine

The purpose of a rendering engine is to transform specific content, such as a scene model, into a format that can be displayed (e.g. image). [HW11] In web browsers such software is being used to render pages (HTML/XML) with certain rules (CSS). Several rendering engines exist nowadays and in Figure 2.1 we can observe their mapping to the common web browsers. Moreover in Figure 2.2 we see the market share of those browsers in mobiles and tablets which are a large portion of the embedded systems domain.

Browser	Web Rendering Engine
Apple Safari	Webkit
Microsoft Internet Explorer	Trident
Mozilla Firefox	Gecko
Google Chrome	Blink (Webkit fork)
Opera	Blink (Webkit fork)

FIGURE 2.1: Browsers – Engines

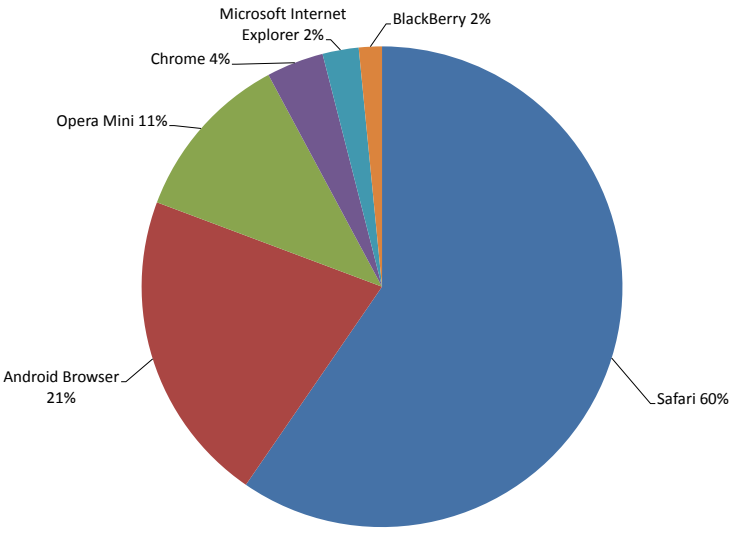

FIGURE 2.2: Mobile Browsers' Market Share [Sha13]

From Figure 2.2 we observe that more than 90% of the current browsers in mobiles and tablets are using the Webkit rendering engine or an engine based on it (Safari, Android Browser, Chrome, Blackberry and Opera Mini after February 2013 [Sof13]). Due to its clear market share advantage we decided to use the Webkit rendering engine for our implementation. Since it is an open source software, Webkit provides various ports under a GNU Lesser General Public License and the one we chose for this thesis is the WebkitGtk+. In the main page of WebkitGtk+ it is stated that most small or large scale projects can be developed using this particular port and also that there is a big community behind the port that supports and maintains it. [Tea13]

2.3 Framework Definition

As stated in [Wie96] a definition of a framework for system development, that fits well to our approach and the framework we present in this thesis, has been given in [RSBG84]. According to this definition, a framework is structured based on an aggregation hierarchy; "At each level of the hierarchy, a sequence of tasks is defined that can be constructed as a division of requirements determination and conceptual modelling into subtasks, and a merging of the resulting steps." [RSBG84]

2.4 Market-Driven Development Objectives and Stakeholders

There are two different approaches for developing a software product based on the type of the market. The customer specific development, also found as bespoke or contract driven, and the market-driven development, known also as packaged software or commercial off-the-shelf [Sve11]. The basic difference between these two approaches is their objectives. For example the objective of Market-Driven development is to attract a wide range of customers and thus is interested in gathering requirements from multiple sources than focusing on one single customer [BD07].

However the lack of a real customer has an impact on the requirements elicitation and analysis processes. The reason is that these processes need to identify new sources of input data, like potential users, who are people that are expected to fit the profile of the product users, [Pot95, SSK99] and internal stakeholders like for example from the marketing department of the company or even external sources like trade publication reviews. [CB95] A benefit of the

market-driven approach that makes it interesting for this case study is the positive influence in the development of new innovative products [Kwa96].

2.5 ISO Quality Model

In this thesis, the ISO/IEC 25010 has been used as a tool for defining the product quality. This standard is an international standard for software product quality measurement, a revision of ISO/IEC 9126 [ISO01], issued in 2011 [ISO11]. The ISO/IEC 25010 standard defines a quality model that comprises of eight product quality characteristics and 31 subcharacteristics (see Figure 2.3). [ISO10]. According to [KvST97] this kind of definition of quality characteristics can be used as part of a design contract and especially in cases of complex embedded systems, it usually contributes to a set of well worked out requirements. For the rest of this thesis whenever we refer to the ISO we mean the ISO/IEC 25010.

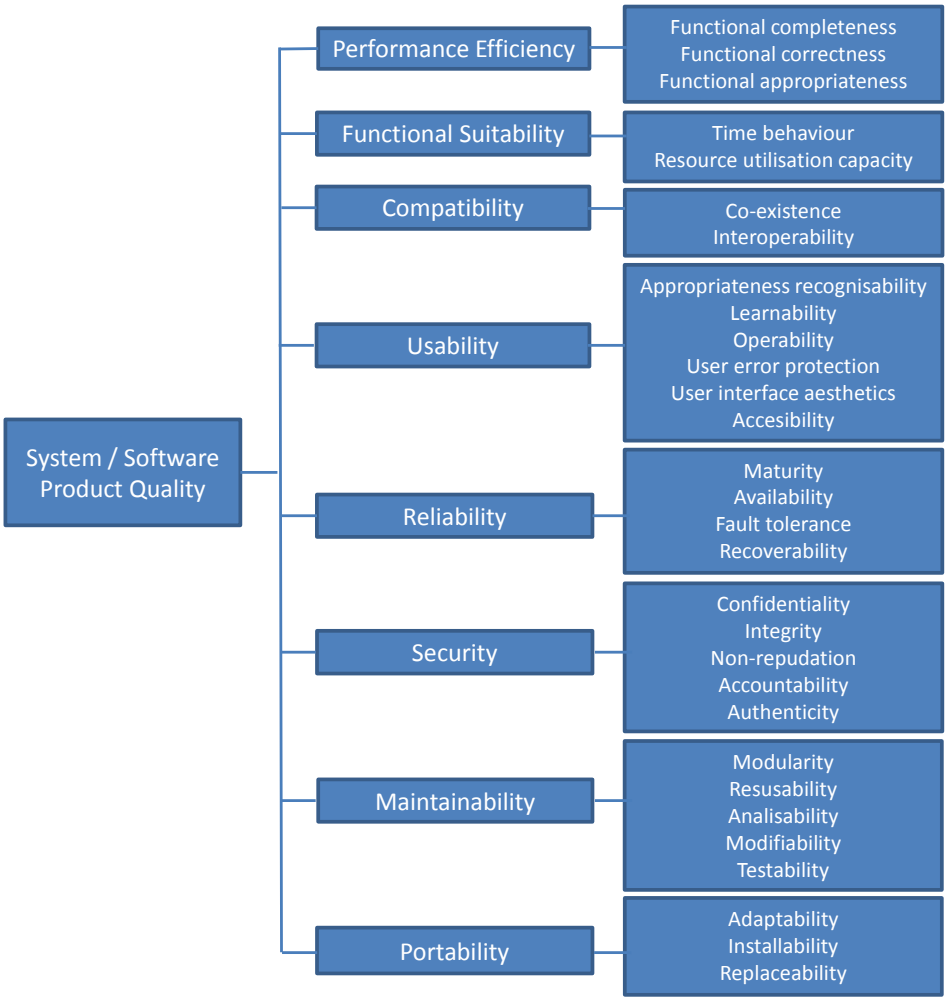

FIGURE 2.3: ISO Quality Model [ISO10]

According to [Sve11], other standards for quality requirements are described in [Lau02] and [DT90]. Also based on [Bar11, WLM05] other examples of quality models include McCall's quality model, Boehm's quality model, Dromey's quality model. However it is out of the scope of this thesis to analyse further the existing quality models.

Chapter 3

HTML Component Implementation

In order to test our framework we worked with Mecels Populus Engine which needed an extension for an HTML component. There was no previous implementation of such a component except from an attempt to use the Chromium Embedded framework in a Windows based system. The goal of this extension was to enable the rendering of HTML content. The rendering engine we chose is WebkitGTK+. The task consisted of a simple implementation along with a build system similar to the one of Populus Engine.

3.1 Architecture

As mentioned above we had to create the HTML extension of the Populus Engine. For this reason we began by designing the architecture of the extension to make sure that the two rendering engines (Webkit, Populus Engine) can be combined. The Webkit rendering engine is capable of rendering HTML content and we used GTK windows to show the results of the rendering. On the other side Populus Engine is also a rendering engine that displays content on the HMI display. In order to connect the two rendering engines we used a shared pixel buffer. The overall architecture is shown in Figure 3.1.

3.2 Implementation Process

For achieving the connection between Populus and Webkit we had to run both rendering engines in separate threads. In the first thread Webkit loads the HTML page into an off-screen window.

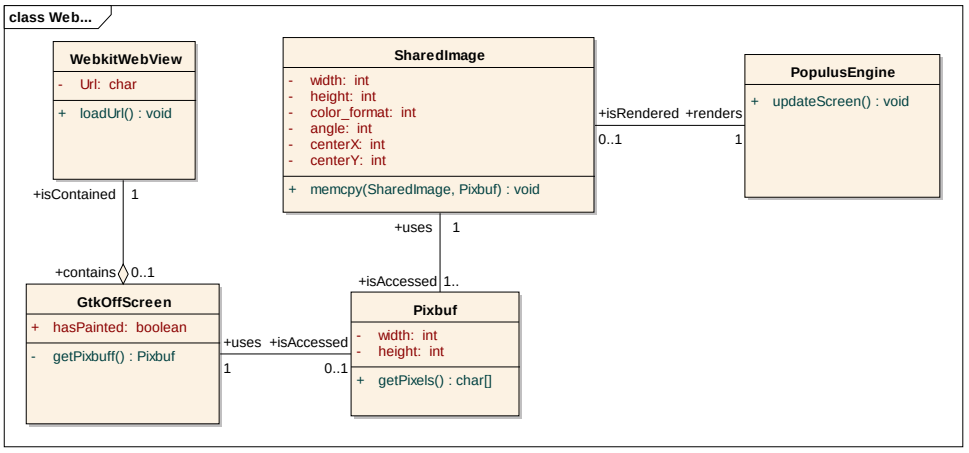

FIGURE 3.1: HTML Component Architecture

By using GTK signals whenever a change occurs in that window, a function is called to write the new pixel set into a pixel buffer. Afterwards a global flag is set to true so that it is known that the pixel buffer contains new data.

In parallel the second thread running Populus Engine is checking periodically, based on the framerate, if there is new content to be displayed. When the flag becomes true the pixels from the pixel buffer are copied in a shared memory image buffer. After this process Populus Engine renders the contents of the shared memory image on the HMI display. Since the only shared object between the two threads is the pixel buffer the implementation is free from deadlocks. The whole implementation process can be seen in Figure 3.2 below.

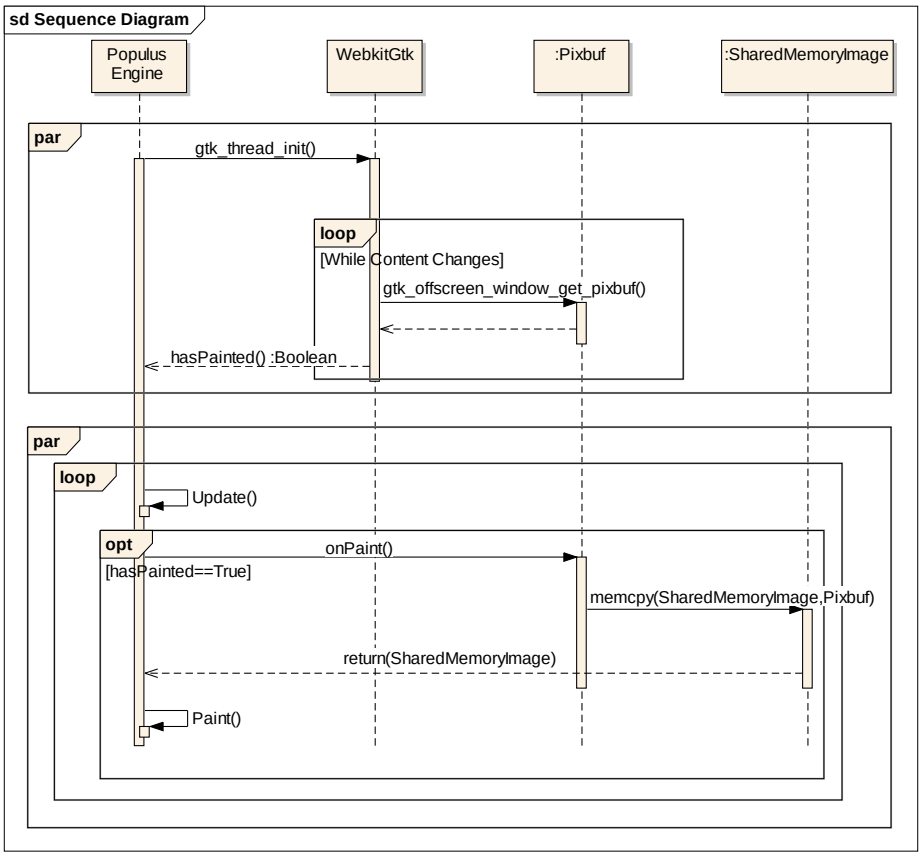

FIGURE 3.2: Implementation Process

Chapter 4

Methodology

To perform our study, we used three different research methods; a set of semi-structured interviews with stakeholders to collect input data for our study, a case study to capture the optimisation possibilities of a rendering engine into an HMI suite and a set of experiments validating that certain optimisation techniques improve the quality characteristics of our test cases, which are based on the input data. The goal for using these three research methods was to establish good knowledge on the domain and achieve triangulation of our data [Joh03, RH09]. In this chapter we will present the methodological framework that we propose and the methodology followed during our case study.

4.1 Framework

The basic idea of this framework is to capture the market needs using a solid methodology and then analyse them properly in order to come up with optimisation techniques and the importance of their application based on the market needs.

The framework starts by capturing the existing or potential market needs and the stakeholders' specific requirements. To achieve this a number of semi-structured interviews is suggested to be performed with different internal and external stakeholders [SGR+12]. Then we should analyse the data from the interviews and form a number of functional and quality requirements capturing their stated needs into an informal requirements specification document. At this point it should be taken into consideration that requirements engineering in market-driven software development is a challenging task and thus in order to achieve specification documents of good quality,

attention is required [KDR+07]. For this reason it is suggested that a well fitting template for the requirements elicitation phase should be used (for example see [GW05]), to ensure that all useful information is documented.

After the elicitation/documentation of the requirements, we should identify commonalities between them and form groups with similar characteristics. The difficulty in this grouping is the fact that in market-driven development the different stakeholders often express their requirements using different terminology and state their needs in different abstraction levels, including direct and indirect requirements. In order to group the requirements we should identify these potential differences. To do so, we use the Requirements Abstraction Model (RAM) [GW05].

The RAM methodology suggests the analysis of the original requirements into specific abstraction levels. Each requirement is classified to one level by answering a number of questions that are presented as a "how to do the placement guide" in the Appendix of the methodology. The abstraction levels that the methodology proposes are four; a *Product Level*, a *Feature Level*, a *Function Level* and a *Component Level*. The Product Level contains requirements interested on capturing the product strategies and the management goals. Thus these requirements are expected to be of high abstraction level, often differing a lot from the definition of the requirements in Software Engineering. The Feature Level contains requirements which add value to the investigation of the development effort. This abstraction level is also of quite high abstraction and the requirements specify the features of the product without any technical details on the functional characteristics. The Functional Level contains requirements describing what the user or system shall be able to do in a way that is usually testable and unambiguous enough to be used from the developers. Finally the Component Level is the lowest level of abstraction describing how something should be implemented. [GW05]

According to the methodology, after classifying a requirement in a level, it is important to work it up and transform it, creating a new requirement, in a higher or lower level of abstraction, if there are no original ones fitting this purpose (see Figure 4.1). This work is done for two reasons. First of all in order to achieve that all requirements are described in all levels of abstraction. The benefit of this process is that the requirements are understood in depth and are understandable from all types of stakeholders, while at the same time it is possible to compare them and set them against each other. So the methodology suggests that all original requirements should be connected to a Product Level requirement and all requirements should be broken down to the

Functional Level. [GW05] Secondly, the application of the RAM method groups the requirements not only based on their abstraction level but also based on their objectives. Thus under a Product Level requirement we expect to find requirements which are closely related to each other. From this point of view a benefit of applying the RAM method is the structuring of the requirements with respect to a strategic goal.

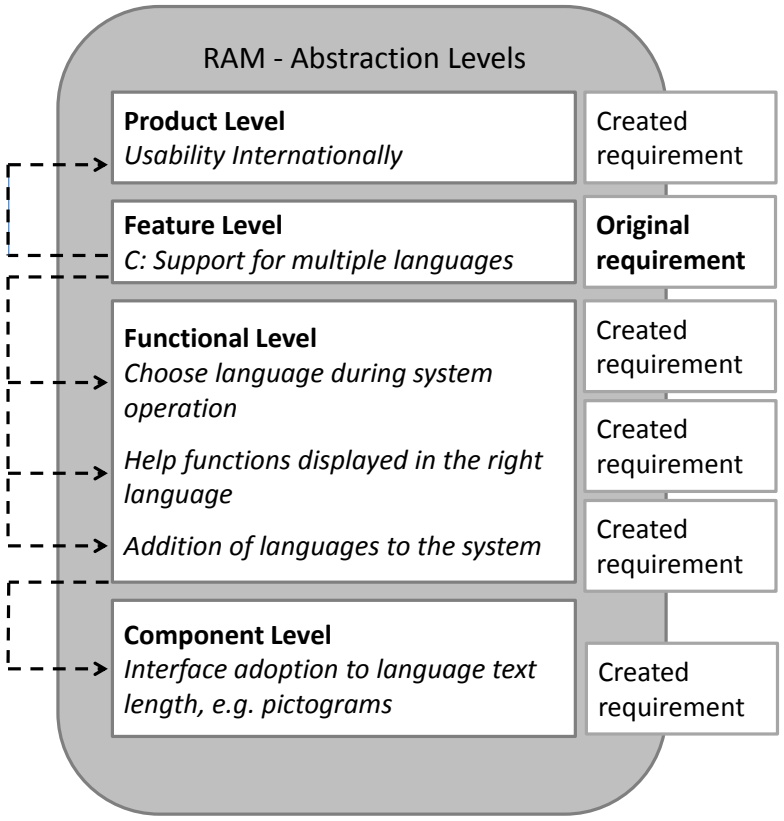

FIGURE 4.1: Abstraction and Breakdown of Example Requirement C: Support for Multiple Languages [GW05]

In our framework the reason we use the RAM methodology is not to create a well written and understandable specification document. On the contrary our goal is to achieve the creation of a requirements specification document with all the requirements organised according to their objectives in groups and linked to Product Level requirements. As we are going to explain below, this happens because our goal is to have requirements which can easily be connected with the ISO quality attributes and their sub-characteristics. Thus, we aim only to this work up of the original requirements and not to their break down to lower levels of abstraction, that the RAM method suggests.

After completing the requirements documentation phase we focus on the Product Level requirements and classify them into functional and quality requirements. These two categories of Product Level requirements will be used differently as explained in the following paragraphs. This categorisation is the last processing of the captured requirements before our requirements specification document is ready to be sent back to the stakeholders for validation. The stakeholders are expected to provide feedback on the quality and the context of the requirements and suggest additions, changes or deletions. During this process it is important to keep the stakeholders unbiased and thus it is suggested that no other information is shared with them. The validation process can require one or multiple iterations and the requirements engineering process can be continued after the validation process has been completed.

The Product Level quality requirements are expected to be abstract enough and consequently easy to connect with the software quality attributes that a quality model defines. We suggest the use of the ISO standard [ISO10] although the use of any other quality model could be applicable for the purposes of this connection. The reason why we aim at this connection of the market requirements with a quality model is that we are interested in opening a path to a solid relation of the requirements with the software metrics. This relation in turn will provide a way for measuring the quality characteristics of our systems before and after applying the optimisation techniques [BCR94]. In this way we will manage to have qualitative results showing objectively if we have succeeded or not to improve our system. Finally in order to complete the processing of the quality requirements, we ask the stakeholders to rank them using a well fitting prioritisation method [Dav05, KWR98]. The stakeholders should be able to understand the reason why they are asked to do this prioritisation and thus it is important to explain clearly to them the goals of this framework. The value added by the prioritisation of the software attributes by the stakeholders, is that as stated in [SGR+12], often there are conflicts between different internal stakeholders when it comes to prioritisation decisions. Thus it is necessary to identify all conflicting preferences and expectations of the involved stakeholders [KOR97, RHNoD+01] before taking any optimisation decisions.

On the other hand the functional requirements captured will provide input for creating test examples. Each of these test examples will test the use of a system feature (that is connected with a captured functional requirement) that we are interested to optimise. In order to make the usefulness of these test examples easier to understand, the following example based on our case is provided: In our case study that we try to optimise an HTML rendering engine which has been integrated with an HMI product, an easy test case would consist of a simple HTML page. Our

rendering engine would try to load the HTML content of the page while at the same time we would be able to measure the performance of our system.

In the beginning of the framework we stated that through the interviews we came up with a number of functional and quality requirements. However during the interviews the stakeholders were also expected to mention their needs for different versions of the product. These different versions were expected to target different market groups, i.e. customers with different needs [vdLSR07]. Thus, these needs should also be extracted when analysing the interviews because they are useful for the next step of the framework.

Until now we have mentioned the existence of several test examples and also the existence of one or several products. For each different product we are going to identify the most interesting test examples. An example would be that the full version of a product would probably be interested in the total functionality of the product and thus in all the test examples. On the other hand a free version of the same product may provide limited functionality and as a result be interested only in test examples related to basic features. Thus, our framework until now works as shown in the Figure 4.2 below.

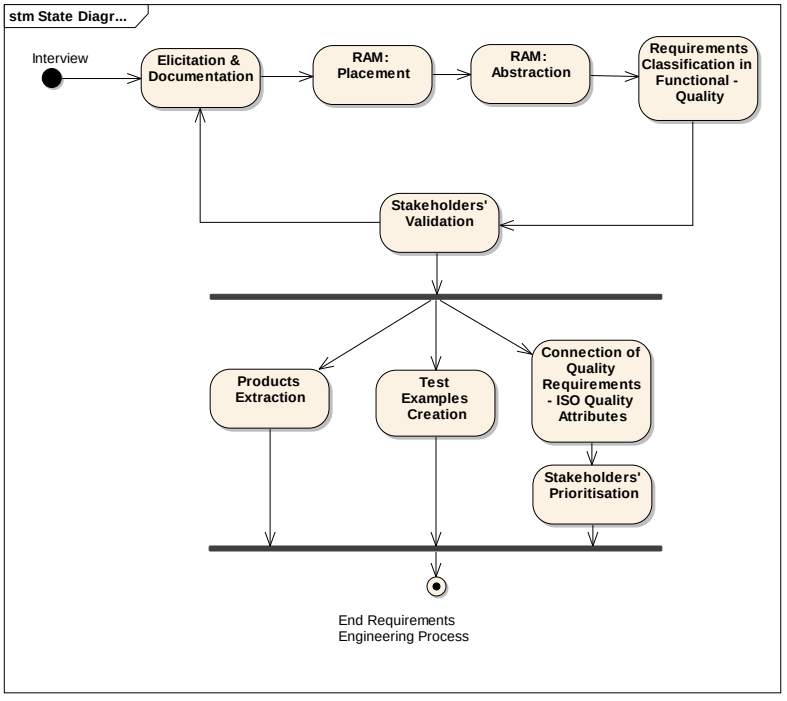

FIGURE 4.2: Requirements Engineering Process

The next step after we have prepared all these foundations of our framework is to take some initial measurements for our examples. To do so we need to come up with some useful metrics.

The reason for using software metrics is to provide a quantitative way of measuring certain attributes of software [TSZ09]. Another benefit of using software metrics for monitoring the quality of software is mentioned in [OH08] where the state of the art for software metrics is being analysed. Through the examination of software metric practices in the software industry the authors came to the conclusion that the use of metrics should happen early in the development cycle. By measuring, often we discover errors sooner and in the end we should make the use of software metrics a habit rather than a time overhead.

From the literature we know that there are several software quality metrics closely related to the ISO quality model [ISO10]. However the literature also suggests the existence of the term "physical metrics" that are applicable in the domain of embedded system and are also connected to the software quality metrics [CLC+10, ORC+08]. More specifically [CLC+10, ORC+08] identify a gap in embedded systems between software quality attributes like coupling and complexity and physical metrics like memory footprint and system performance. The reason for this is the close relation of the software and hardware components in embedded systems and due to this the most common metrics are the physical ones. Both papers conclude by stating that the use of software quality metrics in the early stages of the development of embedded system software can result in a significant benefit of the physical characteristics.

However an important thing to consider is what we really want to measure. In order to identify the proper metrics for the market needs we apply the Goal-Quality-Metrics (GQM) approach [BCR94], because as stated in [RH09] "the definition of what data to collect should be based on a goal oriented measurement technique". The approach is a hierarchical model that at the top of the hierarchy has a set of quality high level goals. Based on these goals certain questions are derived that can define these goals to a good extent. Finally we need to specify the metrics needed for answering the set goals [BCR94]. More specifically the basic components of the approach are defined as presented below.

Conceptual level (GOAL): "A goal is defined for an object, for a variety of reasons, with respect to various models of quality, from various points of view, relative to a particular environment. Objects of measurement are Products, Processes or Resources" [BCR94]

Operational level (QUESTION): "A set of questions is used to characterize the way the assessment/achievement of a specific goal is going to be performed based on some characterizing model. Questions try to characterize the object of measurement (product, process, resource)

with respect to a selected quality issue and to determine its quality from the selected viewpoint." [BCR94]

Quantitative level (METRIC): "A set of data is associated with every question in order to answer it in a quantitative way." [BCR94]

This framework aims at using software metrics to measure quantitatively the quality characteristics of a product, according to the captured market needs. After identifying the metrics that we want to use, we can implement test suites for measuring the performance of the system when running the test examples.

At this point the framework components are completely described. The next steps that should be done are to run the created test suits in order to take initial measurements and identify the bottlenecks. To identify the bottlenecks we suggest to decompose the system and measure each component separately if possible. Then appropriate optimisation techniques should be applied in order to improve the system's characteristics on the specific test examples. Finally the system should run again the created test suite in order to measure and get the post optimisation results. It is important to study and compare the initial and the post optimisation results of the whole test suite. This will result in realising if the optimisation method applied was beneficial for the specific test example we were trying to optimise, but also if it has affected the rest of the measurements of our test suite. A beneficial optimisation technique for one characteristic could be proven harmful for another and in such cases the discussion and approval of the stakeholders will be needed before deciding if the optimisation should be kept or should be dismissed. (see Figure 4.3) In this final decision making step it is important to investigate the potential trade-offs and consider the stakeholders' opinion in order to capture the weight of their importance. We propose a model which provides a structured way to communicate the existing trade-offs and we suggest the involvement of the expert judgement which can be proven beneficial when the market needs are not very clearly defined. However in cases that weights regarding the importance of specific factors can be applied, we suggest to include this information into the framework in order to minimise the tacit knowledge needed for the successful decision making.

At Figure 4.3 we show how the extracted information from the interviews is processed to be transformed into optimisation decisions. The figure shows the separation between the requirements and how the needs for different versions of the product are captured through the interviews with the stakeholders. For the needs of this distinction we have used lines of different length. Also the side curved arrows show the different use of the functional and quality requirements

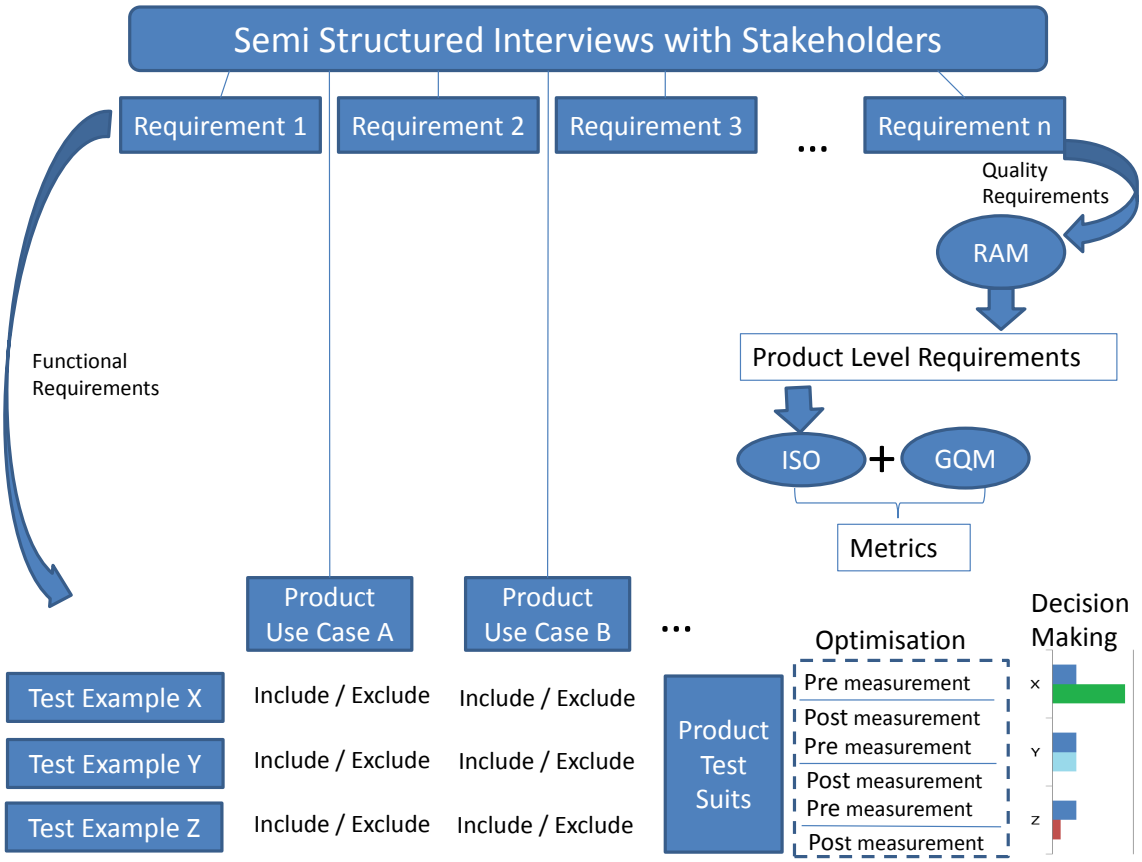

FIGURE 4.3: The Concept of the Thesis

elicited. Regarding the quality requirements, as shown in the right part of the diagram, we apply the RAM method and we come up with Product Level requirements which can easily be connected with the ISO characteristics. Then with the help of the GQM method we come up with useful metrics for our needs. On the left side of the diagram we show how the functional requirements are used in our framework to form test cases for the different versions of the product. If a test example is applicable for a version, we mark it as "include" in order to be included in the test suite for the specific version and thus get measurements about it. It is expected that all test examples will be marked as "include" for at least one product version. At the bottom right corner of the diagram we combine the findings from the functional and quality requirements processing in order to end up with an optimisation decision. To do so, we use the metrics, which we have identified with regard to the ISO characteristics, in order to measure the initial state of our system. Then according to the stakeholders prioritisation, we apply optimisation techniques and we measure again to get post optimisation results. Finally based on the pre and post optimisation results for all versions, we can take decisions about the benefits of the optimisation techniques. We use the bar charts at the bottom right corner of the diagram to show either that

the optimisation technique used on a test example was finally considered to be beneficial (green bars), harmful (red bars) or neutral (light blue bars) for the final product.

4.1.1 The Framework in 10 Steps

Aiming to make our framework easier to follow we have broken it down to a series of 10 simple steps as presented below.

Step 1 Collect data from different stakeholders, through interviews, in order to capture the market needs.

Step 2 Create a requirements specification document and analyse the requirements using the RAM method to achieve a high level of abstraction of all requirements.

Step 3 Separate the high level requirements captured in functional and quality requirements. Also extract any potential need for different products targeting to different market needs.

Step 4 Connect the quality requirements with the ISO software quality attributes.

Step 5 Ask the stakeholders to prioritise the quality attributes.

Step 6 Use the functional requirements captured to identify a set of test examples which can be used to test the quality characteristics of the product

Step 7 Use the GQM method to identify applicable metrics that can measure the quality characteristics of the product.

Step 8 Take initial measurements of the quality characteristics of the products by running the test examples while at the same time applying the selected metrics, in order to identify the bottlenecks.

Step 9 Try optimisation techniques to resolve the bottlenecks. Then, run the test suites again to get measurements showing the outcome of the optimisation.

Step 10 Compare the results of the optimisation techniques to identify the combinations of these techniques that improve the overall system's characteristics.

4.2 Case Study Design and Data Collection

For structuring our case study we followed the guidelines of [RH09, Yin09]. The study we conducted was an exploratory case study investigating a combination of qualitative and quantitative data [RH09]. The purpose of this study is to propose a framework that will be applicable for guiding market-driven optimisation decisions in the embedded systems domain. During this approach we investigate the quality software attributes that are important due to the market needs and thus should guide the optimisation decisions with respect to the improvement of the quality characteristics of the system. The study is based on a set of semi-structured interviews [RH09] (see Appendix B) and aims at the collection of information that would pinpoint the most interesting software quality attributes for the optimisation of our product, from different stakeholders' perspectives. The reason why the method of semi-structured interviews was selected as the best approach to collect input data, is that this approach leaves space for discussion with the interviewees, providing in depth understanding on the stakeholders' needs [KDR+07, RH09, SGR+12]. An alternative approach would be to use a questionnaire. However literature suggests that such an approach could have drawbacks in a study similar to ours [KDR+07, RH09], since it would increase the amount of assumptions made and would increase the chance of misunderstandings caused by the different use of terminology.

4.2.1 Planning

The planning of this study started as one of the very first tasks after a short insight on the relevant literature of the domain. We started by defining the objectives that would lead the study and we came up with research questions that during the time were updated with improvements based on the findings of our research. Also we came up with a time plan that was also updated in a weekly basis according to the flow of the research. In our research no formal protocol was used. Instead we made use of a scrum board for capturing the strategies that we would use for our research and the tasks that we had to accomplish and also for prioritising these tasks, helping us out with the weekly planning. Additionally we used Git as a collaboration tool for sharing code throughout the implementation process.

Specifically regarding the planning of the interviews we requested to meet as many stakeholders as possible, from different departments of the company and the project manager of the company responsible for this thesis arranged the interviews by selecting the most well fitting interviewees

based on two factors; their capability to provide us with useful and accurate information and their availability. Thus we did not influence the selection of the interviewees. Finally we conducted one interview with the marketing department, one with a component owner of a similar project of another product and one interview with a code developer of our product. More specifically the interviewed stakeholders were:

Marketing Department (MD)

Customers : Suppliers of HMI to OEMs and as an extension OEMs

Stakeholders: Populus team, Mecel Online Service Project team, Management Department

Source of Information: Customers, Automotive Industry Conferences, Internova Research Projects

Online Service Component Owner (CO)

Customers: Volkswagen

Stakeholders: Management Department, Marketing Department

Source of Information: Customers, Requirements Specifications, Other OEM requirements, Test groups, Conferences

Populus Code Developer (D)

Customers: Tier One (Delphi and as extension their customers), OEM

Stakeholders: Management Department, Marketing Department

Source of Information: Delphi's Advanced Research team, Marketing Department (VW), Genevi initiative (consisting of suppliers and OEMs)

4.2.2 Data Collection

Interviews

A set of semi-structured interviews was used for collecting some initial input data for our study. The interviews were conducted by both the researchers and one interviewee per time. Each interview had a duration of one hour and was recorded to prevent mistakes during the documentation and the interpretation of the data. During the interviews also hand written notes were kept, however the roles were not strict and all the participants were part of the discussion. The questions of the interview were prepared in advance and had been sorted in logical order, aiming to a structure that would promote unbiased answers. The questions were not available to the interviewees, who knew only basic information about the thesis we conduct. The list of questions

consisted of six questions occasionally split up into sub-questions. The possibility for a few follow up questions was in all cases agreed at the end of the interview.

The interviews always began with a question about the background of the candidates and their tasks and responsibilities at Mecel. Then the major part of the interview was starting with a question focusing on customers information and the communication with them. Then a discussion was done around the question about the software engineering aspects that the different groups of customers seem to be interested to. The next two questions were asking about the importance of the HTML feature both according to the interviewee's and the potential customers' opinion. Finally the last question was aiming to identify how the HTML component is expected to be used according to the market's needs. At the end of the interview a short summing up was always done to make sure that all aspects that we were interested to capture had been sufficiently discussed and correctly understood.

Software Metrics

In order to identify which metrics would be useful for our study we applied the GQM approach [BCR94]. After defining the goal we came up with a number of questions which were highly related to ISO quality attributes based on which the quality requirements had been categorised. Then we identified certain metrics that could be useful to measure our system for the needs of this thesis. Finally we decided to select the two most interesting metrics, according to the Product Level requirements prioritisation that the stakeholders did, and use it as proof of concept for our framework. In order to start taking the initial measurements we implemented scripts for the selected metrics. These scripts run for each test example before any optimisation technique was applied. The results of these measurements were automatically saved in log files. These results gave us indications about the bottlenecks of our system which helped us to decide what optimisation techniques could improve the performance of our system. A second round of the post optimisation measurements run after every optimisation technique was applied. Finally the initial and post optimisation results were analysed and compared in order to decide if the optimisation technique was beneficial or not for the total product.

This final decision making is very important and often can be very complicated since it is possible that an optimisation technique which improves the performance of a test example can be harmful for another test example. Especially in cases where the optimisation applies to more than one products, aiming to different market groups, it could be common that an optimisation

that aims to improve a specific feature of a product may introduce an overhead for the products that do not contain that feature. In such cases the decision making process should involve the stakeholders in order to identify the most beneficial set of optimisation solutions for all the related products.

4.2.3 Analysis

Since this case study involved the collection of both quantitative and qualitative data different ways of analysis were used. For the requirements' analysis the two authors collaborated for applying the RAM method on the extracted requirements. The requirements had to be worked up to higher level of abstraction as well as to be grouped together whenever possible. Finally they had to get documented in the form of a requirements specification document, in order to be discussed with the stakeholders during the feedback sessions. For validating the quality of the requirements analysis we consulted a requirements engineer at Mecel.

For the analysis of the results after the connection of the quality requirements with the ISO quality model we used the method of tabulation [RH09]. The same method was found useful for the analysis of the prioritisation results. In this way we achieved a better organisation of the results which by extension helped with the analysis of the collected data.

Finally for the analysis of our optimisation findings we used charts for visualising the change on the system's characteristics after the application of an optimisation technique.

Chapter 5

Results and Analysis

In this chapter we are going to present the results of our case study in the form of tables and figures and then analyse them further, in order to provide sufficient information for every decision taken [RH09]. The chapter is divided in six subsections; Requirements Elicitation Findings, Requirements Processing Findings, Requirements Prioritisation Findings, Software Metrics Findings, Optimisation Findings and Framework Application.

5.1 Requirements Elicitation Findings

In our case study we conducted three interviews with internal stakeholders from different departments of Mecel, a manager of the marketing department of the company, a component owner working in the same company but for another product similar to ours, and a code developer of the Populus product team.

As mentioned in [RH09] one way to decide when enough interviews have been conducted is the point of "saturation", i.e. the point when the input information starts repeating [CS08]. However in our case study we had limited number of interviews due to the limited availability of the potential candidates. As already described in the planning methodology section in Chapter 4, the number of the interviews and the interviewees themselves were selected by the project manager of the company responsible for this thesis, who was familiar with the needs of our project and was capable to identify the best possible interviewees for our needs. This was done in order to ensure the selection of the most capable to provide us useful information stakeholders. Although two of the interviewees were completely unrelated to this master thesis and thus not biased about

the expected results, the code developer was selected to be the supervisor of this thesis, as he was considered to be the most experienced developer in the domain.

Additionally in our case study we conducted interviews only with internal stakeholders. The reason for this decision was the fact that the company has no real customers interested in this extended version of the product yet and according to the internal interviewees the need for this product has only been discussed as an indirect potential need by the parent company or during conferences as a hot topic for discussion for the IVI domain.

5.2 Requirements Processing Findings

From the interviews we conducted, a total of 33 functional and quality requirements in various abstraction levels were extracted. After applying the RAM method we came up with a total of 49 requirements which were grouped into 13 groups. 8 of these groups consisted of quality requirements. These 8 groups are presented in Table 5.1.

As shown in Table 5.1 in our case study only the three higher abstraction levels were applicable (Product Level, Feature Level and Function Level) during the placement process of the RAM method. This was not a decision we made but it came up from the requirements we collected. We assume that the reason why there were no Component Level requirements captured, is that the interviews were quite general, focused on the potential optimisation needs of a new product without known specific customer demands, and thus the "how" was out of the scope of the discussion.

In Appendix A we provide also the specification document that was used during the elicitation process for documenting the extracted requirements and basically for analysing them using the RAM method. However it should be underlined that our aim was just to capture basic requirements giving us information about the optimisation needs for the integrated HTML component to the Populus tool and that it was considered to be out of scope for this thesis to create a very detailed specification document. For this reason several fields of the proposed template in [GW05] have been omitted. More specifically the fields *Requirement Owner*, and *Requirements Manager* were omitted because the two authors of these thesis were the ones collaborating for this tasks. Additionally to this information omitted the fields *State*, *Reject reason*, *Due date*, *Version*, *Date of creation*, and *Last changed* were not used because in our case, as it is described above, the

Product Level Requirements	Feature Level Requirements	Functional Level Requirements
CR2.PL: The product shall be responsive	**CR2.1.FeL:** The product shall provide quick HTML content rendering **CR2.2.FeL:** The product shall provide links/buttons which respond quickly **OR2.3.FeL:** The product shall be able to read quickly from external devices **CR2.4.FeL:** The product shall provide a quick page scrolling feature **CR2.5.FeL:** The product shall provide a smooth UI **CR2.6.FeL:** The product shall have a quick start up time	**OR2.1.FuL:** The rendering engine shall render the HTML content of a page in less than 100ms. Such a performance is expected when loading a small page (fitting in the screen) containing only text and up to two buttons provided that the HTML is on disc so that it does not include network load times. **OR2.2.FuL:** The rendering engine shall respond in a click of a button or link within 100ms. In this time we expect that an event will be triggered and the user will get some feedback showing that the button has been pressed successfully (ex. change of link color) **OR2.3.FuL:** The system shall be able to read from the disk/ flash in less than 100ms **OR2.4.FuL:** The rendering engine shall be able to scroll the content of a web page with framerate higher than 10 f/s. **OR2.5.FuL:** The UI of the rendering engine shall have a minimum framerate 10f/s. To consider the UI as smooth the framerate shall be higher than 30f/s **OR2.6.FuL:** The system shall be able to start up in less than 10s
OR3.PL: The product shall be lightweight	**OR3.FeL:** The memory consumption of the product shall be low.	**OR3.FuL:** The memory consumption shall be limited to lower than 100MB.
CR4.PL: The product shall be portable.	**OR4.1.FeL:** The product shall be able to run on multiple Operating Systems **OR4.2.FeL:** The product shall be able to run on multiple hardware platforms	
CR6.PL: The product shall be able to operate with other systems	**OR6.1.FeL:** The rendering of the HTML context shall not interfere with any other system in the car **OR6.2.FeL:** The user shall not be allowed to browse uncontrolled web content for safety reasons. Only connections provided by the customer backend shall be allowed.	
OR7.PL: The supplier wants to use the product for creating quickly HMIs or applications using HTML which will be rendered by the integrated rendering engine of the product.		
CR8.PL: The product shall be configurable.	**OR8.1.FeL:** The features of the product shall be easily configurable based on the needs of different customers.	**OR8.1.FuL:** The user shall be able to configure the functionality of the HTML of the product based on the existing needs. **OR8.2.FuL:** The user shall be able to configure the appearance of the HTML of the product based on the existing needs and their personal taste.
CR9.PL: The product shall promote driving safety.	**OR9.FeL:** The content displayed in the browser window shall not be interactive when the car is in motion, so that to eliminate the chances to distract the driver's attention while driving.	
OR10.PL: The product cost shall be preserved low		

TABLE 5.1: Quality Requirements Grouping

overhead for capturing this information would be extremely high compared to the benefits that we could gain.

Also regarding the application of the RAM method, as already explained in our framework in Chapter 4, we are interested only in the bottom-up work up of the requirements, aiming to connect them with requirements on a higher level of abstraction. Thus the break down of the requirements, which is suggested in the RAM method, has been omitted because it was decided

that it would not add any value to the goals of this framework. However, in cases that a complete and well analysed requirements specification document is considered to add value in the process, we suggest the full application of the RAM method.

Moreover four groups of functional requirements were extracted, one of which contained requirements applicable in the version of the product used in our case study. This group of requirements was dealing with the rendering of simple HTML pages and formed the test example used in our study. Finally a requirement for two different products needed for different customer needs was extracted. More specifically a product using an HTML component in the form of multiple widgets and a full functional browser were captured as interesting products to be implemented. For the needs of this thesis and due to time restrictions, we implemented a first simple version of the widget product. For a more detailed specification document see Appendix A.

5.3 Requirements Prioritisation Findings

After the requirements elicitation, the stakeholders were asked to prioritise the extracted quality requirements based on the 100$ prioritisation technique [Dav05]. Thus we asked the interviewees to imagine that they had 100$ and that they had to distribute them among the eight quality requirements in a given table. We explained them that they had to try to distribute the money in such a way that if requirement i is x times more important than requirement j, then they had to give it x times more money. We also asked them to avoid giving the same amount of money to all requirements since this would have no effect to the outcome of this prioritisation. This process was selected to be used as a simple method that would not be difficult for the stakeholders to apply, while it also applies weights to the prioritised items showing the importance of these items when compared to each other. The prioritisation gave us the following results as shown in Table 5.2.

Then these Product Level quality requirements were mapped to the ISO quality attributes as shown in Table 5.3.

As shown in Table 5.3 we have linked the quality Product Level requirements to the ISO quality attributes. This linking was done after the creation of the requirements specification document

Source	Product Level Quality Requirement	Prioritisation MD	CO	CD	Total
Populus Code Developer (CD) Marketing Department (MD)	**CR2.PL** The product shall be responsive	30	15	40	**85**
Populus Code Developer (CD)	**OR3.PL** The product shall be lightweight	17	5	10	**32**
Online Services Component Owner (CO)	**CR4.PL** The product shall be portable.	8	10	50	**68**
Populus Code Developer (CD) Marketing Department (MD) Online Services Component Owner (CO)	**CR6.PL** The product shall be able to operate with other systems	10	10	0	**20**
Marketing Department (MD) Online Services Component Owner (CO)	**OR7.PL** The supplier wants to use the product for creating quickly HMIs or applications using HTML which will be rendered by the integrated rendering engine of the product.	5	0	0	**5**
Marketing Department (MD) Online Services Component Owner (CO)	**CR8.PL** The product shall be configurable.	12	0	0	**12**
Online Services Component Owner (CO)	**OR9.PL** The product shall promote driving safety.	15	30	0	**45**
Marketing Department (MD) Online Services Component Owner (CO)	**OR10.PL** The product cost shall be preserved low	3	30	0	**33**

TABLE 5.2: Prioritisation of Product Level Quality Requirements

and was discussed with the stakeholders who approved the categorisation of the quality requirements under these ISO characteristics, as shown in the table. This table is the input for forming the questions needed when applying the GQM method.

Finally from the Tables 5.2 and 5.3 we present in Table 5.4 the importance of the ISO quality attributes according to the prioritisation of the stakeholders.

According to Table 5.4 the most interesting aspect for optimisation is the performance efficiency of the product, and more specifically its responsiveness as shown in Table 5.2. This result is not surprising since the bad responsiveness of a product can easily have an impact to the first opinion of a customer and is important for a good user experience of the product. Second comes the portability aspect. This result can be explained by the fact that at the moment Populus is used on various platforms and different operating systems and thus it is important that the extended version of the product can also be portable respectively.

Product Level Quality Requirement	ISO Quality Attribute
CR2.PL The product shall be responsive	**Performance Efficiency** (Time Behaviour)
CR4.PL The product shall be portable.	**Portability** (Adaptability)
OR9.PL The product shall promote driving safety.	**Usability** (Appropriateness recognisability)
OR3.PL The product shall be lightweight	**Performance Efficiency** (Resource Util. Capacity)
OR10.PL The product cost shall be preserved low	**Performance Efficiency** (Resource Util. Capacity)
CR6.PL The product shall be able to operate with other systems	**Compatibility** (Coexistence)
CR8.PL The product shall be configurable.	**Portability** (Replaceability)
OR7.PL The supplier wants to use the product for creating quickly HMIs or applications using HTML which will be rendered by the integrated rendering engine of the product.	**Usability** (Operability)

TABLE 5.3: Quality Attributes Connection with Product Level Quality Requirements

ISO Quality Attribute	Prioritisation Percentage
Performance Efficiency	50%
Portability	27%
Usability	16%
Compatibility	7%

TABLE 5.4: Importance of Quality Attributes According to the Prioritisation Outcomes

5.4 Software Metrics Findings

For the needs of the case study we used the GQM method to identify some quality metrics that could be applicable for measuring the quality attributes (which have already been linked to the extracted quality requirements) in the domain of embedded systems.

To do so, we started by stating the questions that would lead us to the metrics we would use for measuring the characteristics of our system. Then we conducted an extensive research of the literature, which introduced us a large number of papers dealing with software metrics. Among them we found some metrics related to the four ISO attributes, and more specifically their six sub-characteristics, which were found to be relevant with our Product Level requirements. Below we provide some references to these papers. However it is important to underline that our aim was just to provide some insight into a few existing metrics and we did not intent to identify and suggest the best quality metrics, since this domain is very broad and out of the scope of this master thesis.

Performance Efficiency

In [KBS+09] it is stated that in the automotive industry there is a need for optimising the resource usage and the time-behaviour (scheduling) due to the increasing number of multi-core systems. The first proposed step is to identify the critical operations of a system where optimisation is needed. In order to measure the timing behaviour they introduce the use of performance indicators which are positive values related with specific tasks of the system. High values indicate a need for optimisation and a slower performance compared to a low value. They also define response time with the Equation 5.1 below:

$$R = f - a \tag{5.1}$$

Where R is the response time of a task and is equal to the difference between the finishing time (f) and the activation time (a). We are going to use this formula in our implementation to measure the response time of certain parts of our system. For interpolating the final data it is proposed to calculate the maximum, minimum and average values.

In [Yuy05] a method is proposed to measure certain physical attributes of embedded systems such as the response time and the throughput. It is stated that the factor that drives the embedded systems are the hardware resource limitations that they have. The embedded system domain has recently begun growing however the software metrics proposed usually only focus on general software. The paper focuses on time behaviour and resource utilization and proposes a metrics model to measure them.

Portability

In [Moo97] the authors propose a framework to enhance the portability of a product by making

sure to consider it in every step of the development cycle. Although portability is one of the highly desired characteristics of a software product there are no generally accepted metrics for it. As a metric an equation is proposed to measure the degree of portability for a product. The Function 5.2 is the proposed equation:

$$DP = 1 - (cost\ to\ port\ /\ cost\ to\ redevelop) \tag{5.2}$$

Where DP is the degree of portability, which is equal to the difference between 1 and the ratio of the cost to port the product to a different system with the cost of redeveloping the product for the new system. The closer the degree of portability is to 1 the more "portable" a product is.

Usability

In [BFB91] the authors propose a usability metric specifically for hypertext systems. The metric takes into account the accessibility, the orientation and the user interaction to measure the overall usability score. The Equation 5.3 is the equation suggested in this study:

$$US=c1*A+c2*O+c3*I \tag{5.3}$$

Where US is the usability score, A is the accessibility parameter, O the orientation parameter, I the interaction parameter, c1,c2 and c3 are the weights representing the importance of the each parameter to the users. The paper provides formulas to calculate every parameter before trying to find the usability score of a system.

Compatibility

Due to the fact that this quality attribute is a new addition to ISO 25010 our initial search for literature did not capture a formal way of measuring it. The reason for this might be the change in terminology that occurred together with the changes of the new ISO standard. However, in our case study still it would be infeasible to apply a method for measuring the compatibility, since we only run Populus Engine in our system.

Finally as shown in Table 5.5 we selected some metrics that were applicable for the needs of this thesis and could answer the stated questions.

Goal	Purpose Issue Object (process) Viewpoint	Identify the quality characteristics that should be optimised based on the internal stakeholders' needs.
Question	Q1	Is the performance of our product efficient?
Metrics	M1 M2 M3	Framerate of specific product parts Changes in lines of code Memory footprint of the entire product
Question	Q2	Is the product portable?
Metrics	M3	Degree of portability
Question	Q3	Is the product compatible with other systems?
Metrics	-	Cannot be measured with our current implementation
Question	Q4	Is the product usable?
Metrics	M4	Usability score

TABLE 5.5: Quality Attributes Connection to Metrics

5.5 Optimisation Findings

Based on the outcome of the prioritization that was presented in the previous section, we began the extraction of the initial measurements for the highest ranked quality attribute. The stakeholders decided that the most important aspect of the product was the performance efficiency and thus we focused on improving our implementation with regard to their decision. The quality requirements categorised under the performance efficiency attribute focus on the responsiveness, the preservation of low cost and the development of a lightweight product, presented in such order as ranked by the stakeholders. In Table 5.5 above, it was stated that in order to improve the response time of the product we could focus on measuring its framerate. Since our product is composed by two different rendering engines we decided to measure the framerate of each rendering engine independently, in order to identify the existing bottleneck in the implementation and realise which of them had room for improvement. Additionally we used the Lines of Code (LOC) metric to measure the size of the initial implementation, in order to identify the number of changes after the optimisation techniques would be applied. The number of changes could give us an indication regarding the cost required for applying each optimisation technique.

We decided to measure the framerate by using timestamps in specific parts of the code. In the Populus engine code we measured the difference in time between the display updates when there

was content to be rendered. Since Populus engine uses a multi-buffer technique, utilizing pre-render content and fast buffer swapping, we only measured the swaps when the buffers contained new content. In order to measure the Webkit framerate we just calculated the difference in time between two sequential renderings as it only uses one buffer. Then this time difference was transformed into framerate (frames per second) using the Equation 5.4 below:

$$Framerate = 1000 / time\ difference\ in\ milliseconds \tag{5.4}$$

In Table 5.1 the average initial framerates of Webkit and Populus are presented respectively.

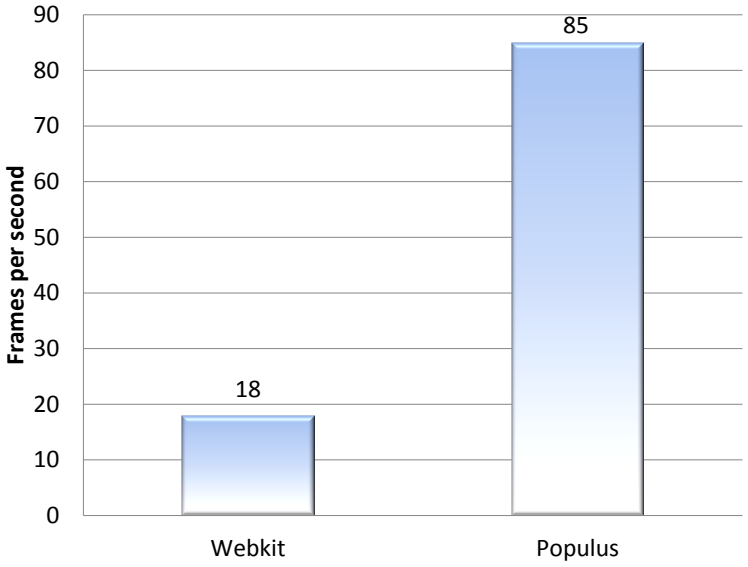

FIGURE 5.1: Initial Framerates of Populus and Webkit

The outcome of this comparison drove us at the conclusion that from the two rendering engines, Webkit is the one with the lowest framerate on which our optimisation should be focused. Thus, afterwards we investigated the possibilities for optimising Webkit.

As described in Chapter 3 Webkit renders its content in an offscreen window and saves the pixels in a buffer which then is transferred to Populus Engine. We realized that a different approach would be to have Webkit rendered in a separate window, stripping our implementation of the intercommunication overhead. By displaying the outcome of each rendering engine separately we succeeded in a better overall framerate. As opposed to our initial implementation, where user interaction needed to pass from one rendering engine to the other, by having Webkit rendering in its own window, there was no need for such an action. In addition most Webkit features like CSS3, Adobe Flash and XML technologies [Fea09] were available without further effort. Another optimisation method was to change the Webkit rendering target from the X11

window system [Fou13] to the Direct Frame Buffer (DirectFB) [Buf13]. Both were tested for our implementation as surfaces for Webkit to render its content.

Figure 5.2 presents a chart which shows the average Webkit framerate results of all implementations described above. The results presented is a collection of the average framerate, after 50 runs of each optimisation method. As already described above, this means that for each run we measured the difference in time between all the pairs of sequential display updates of Webkit, when there was content to be rendered.

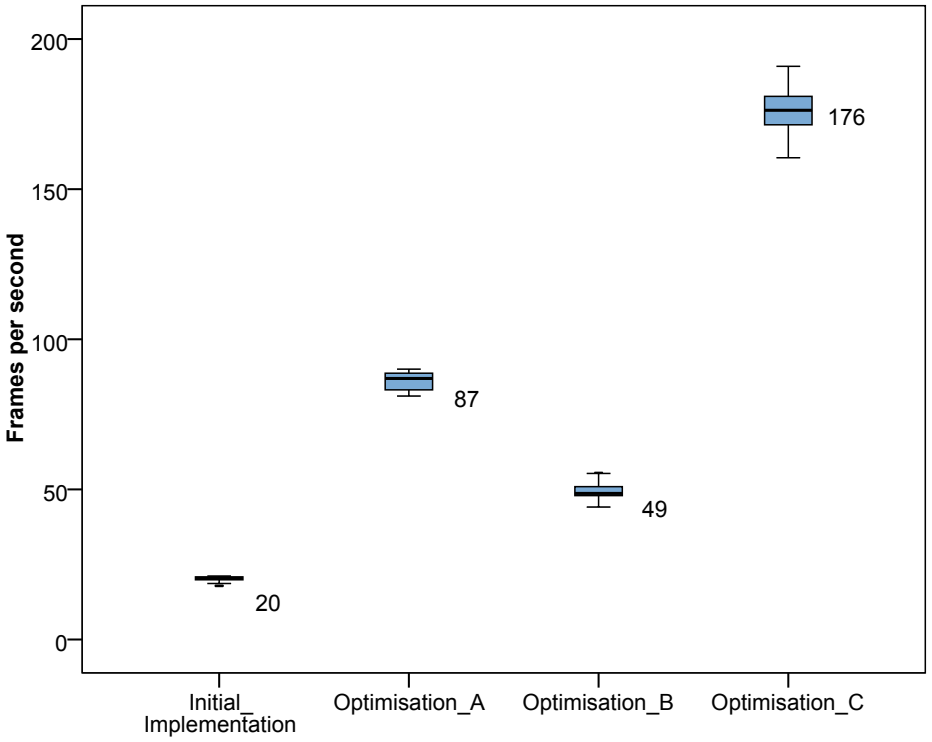

FIGURE 5.2: Post Optimization Webkit Framerate - 50 Samples

In Figure 5.2 it is shown that all three optimisation techniques are beneficial with respect to the Webkit framerate, when compared to the initial implementation. Also a first observation is that there is no overlapping between the results of the initial implementation and the optimisation techniques, neither with regard to their quartiles, nor their maximum and minimum values. This finding is interpreted as a significant optimisation improvement. The same is observed among the results of the different optimisation approaches, and thus the greater the average, the better the optimisation approach is considered.

More specifically, the optimisation C, using a DirectFB in 2 windows, appears to score the highest framerates with an average of 176 frames per second, while in the second place comes

the other DirectFB implementation, with an average of 87 frames per second. The fact that we get higher framerate when using the DirectFB as the target surface for Webkit is explained by the lighter nature of the DirectFB, operating on top of the Linux Frame Buffer, which is part of the Linux Kernel, in contrast to the more complex X11 window system.

In Figure 5.2 we observe an increase of the average framerate when Webkit uses its own window (Optimisations B and C) independently of the rendering target (X11 or DirectFB). As explained previously this is a result of the necessary communication between Populus and Webkit in the initial implementation and optimisation A.

Finally it is important to mention that a known issue that affected our results was the Linux process scheduler. This occurred because we got the time measurement from the timestamps in the various parts of our implementation in milliseconds before converting them into framerate. The task of the process scheduler is to assign a period of time for each running process. These periods of time are called timeslices and specify how long a process can utilize the CPU. In Linux timeslices vary from 10ms to 230ms, depending on the priority level of the process. In our initial implementation the influence from the process scheduling can have an impact on our measurements between 0.18% and 4.18%. The same influence in optimisation C ranges from 1.82% to 41.82%. As a result optimisation C is 10 times more affected by the process scheduling than our initial implementation. A delay of a few milliseconds due to process scheduling can have a much higher impact in high framerate scenarios. [BC05]

To validate the reliability of our results with respect to the sufficient number of samples used, we doubled the sample of the collected data and we created Figure 5.3. Thus this figure shows the Webkit framerate results, based on 100 samples.

From the results shown in Figure 5.3 we realise that the higher the framerate is the bigger the variance on the results we get. This, as already explained above, can be interpreted by the high impact that a delay of a few milliseconds, caused by the Linux process scheduler, can have in cases of high framerate scenarios. Our initial implementation as well as sation B show a quite stable behaviour with slightly lower average framerates. On the other hand the results for optimisations A and C show a more significant change at their lower and upper quartile values, that could be interpreted as a need for further sample collection. However, for the needs of our study the observation that Figure 5.3 preserves the same layout like Figure 5.2,as well as the significant difference between the average framerates of the three optimisation techniques

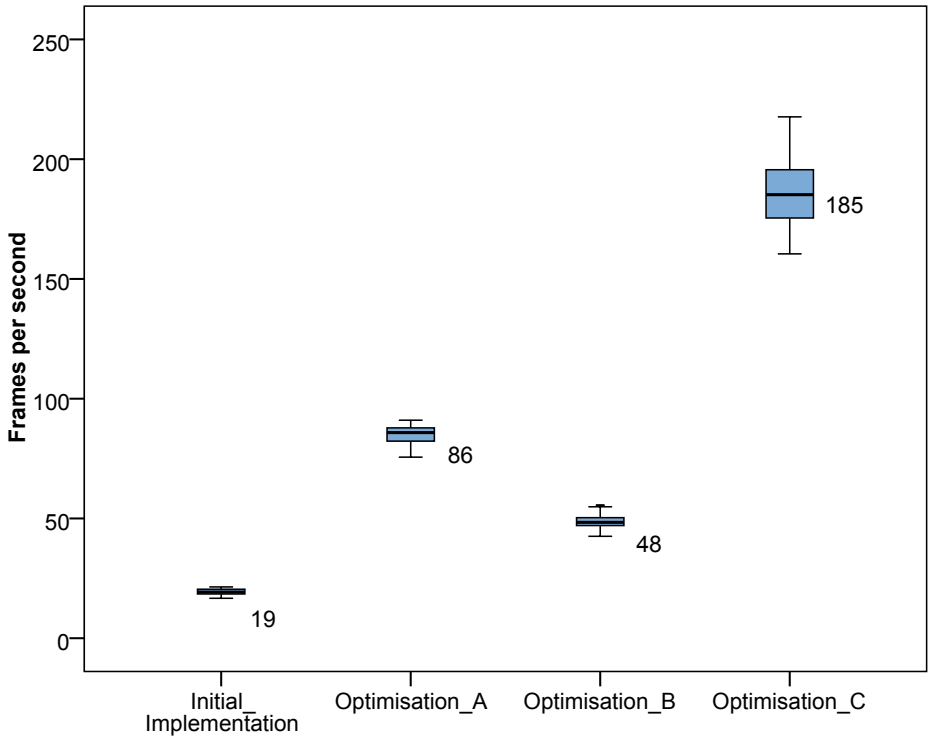

FIGURE 5.3: Post Optimization Webkit Framerate - 100 Samples

applied, validates the reliability of our sample data, with respect to the sufficient number of samples used.

Additionally to the framerate results, we investigated the cost impact of each optimisation technique. To do so,we tracked the lines of code that were changed from the initial product. Since we were working on a branch of Mecel's Git repository, we used the "*git diff –stat*" command, which calculates the difference between commits or trees of a repository. In our case we calculated the difference between the working tree of our implementation and an empty one. The LOC value for our initial implementation was 2,668,006 lines. While comparing this number to the LOC value for optimisation A, which uses two separate windows instead of a shared memory, we observed a change in 59 lines (0,0022%). Moreover the difference of LOC between the Xserver version of our implementation and the DirectFB version of optimisation B was 25 lines (0,0011%). Additionally for the transition from Xserver to DirectFB, we had to build manually 7 libraries which in turn increase the overall cost. Furthermore in optimisation C we have changes in 84 lines of code (0,0031%) in addition to the 7 aforementioned libraries.

According to these results we observe that optimisation B is the one which requires the minimum number of changes, however all the three optimisation techniques require a relatively small number of changes compared to the initial implementation. Thus considering the trade-off between the cost and the framerate improvement, one could claim that optimisation C is the best optimisation method to apply. Although optimisation C seems to be the most beneficial for our implementation, increasing more than 8 times the performance of the system with respect to the framerate, it could be the case that the stakeholders would prefer optimisation A. Optimisation A improves the average framerate only about 4 times, but it does not require the manual installation of these 7 libraries mentioned above and additionally requires less changes in the code. Another relevant observation that should be considered for taking this decision, is the framerate limitations of each optimisation technique with regard to the limits introduced by the Populus framerate. For example, in the case of optimisation A, the applicable framerate is dependent to the framerate that Populus Engine can utilise, while in the case of optimisation B and C, the use of a second window make the rendering framerate of Webkit independent from Populus. Thus it is suggested that in such cases the stakeholders should be responsible for taking the final trade-off decisions regarding the most beneficial optimisation techniques for the product.

5.6 Framework Application

Finally we present all aforementioned results of our case study combined together as our framework suggests. Figure 5.4 can be used as an example of how the framework is expected to look when real data are applied.

Starting from top to bottom and from left to right we see that in our case study we conducted three interviews with internal stakeholders and we extracted a total of 33 requirements, as well as two products that would cover different customer needs. During this thesis we implemented and used for our case study a version of the one of the products and specifically a basic widget functionality of the integrated HTML component. However in this figure we present the "Full Functional Browser" product too since this information was extracted by the interviews.

After extracting the requirements we separate them into functional and quality requirements. As shown in the figure the quality requirements will be processed according to the RAM method as described in Chapter 4, resulting in a total of 49 requirements, grouped into 13 groups. The functional requirements will form text examples, which are actually test cases that we should

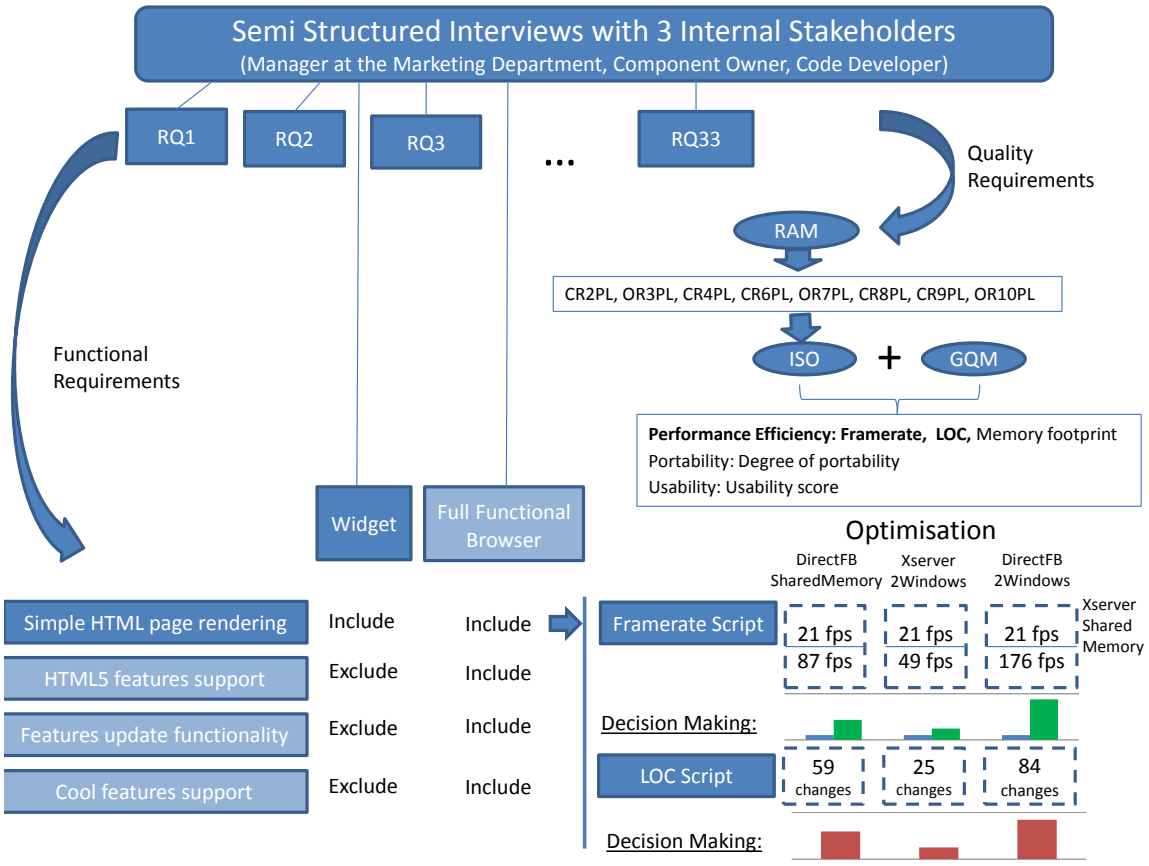

FIGURE 5.4: Framework Application in our Case Study

run while measuring the characteristics of our system. In our case study one test example was of interest for this initial implementation of the product. By the term "Simple HTML page rendering" we mean the rendering of a page that does not contain complicated web content like heavy graphics, but only text and a couple of buttons. In our case study we used the ACID3 web page [Pro13], which was considered to be a good example of a web page with non-static content which also tests the compliance of the rendering engine with web standards. The non-static content would help us measure the framerate later on, since we would be able to get more than one measurements for each page load.

After applying the RAM method for the quality requirements, we came up with eight high level quality requirements, which could be easily connected with certain ISO quality attributes. These ISO characteristics were afterwards connected to certain metrics that were considered to be useful for the needs of our case study. This connection was performed using the GQM method. The selected metrics are the framerate, the changed Lines of Code, the Memory Footprint, the Degree of Portability and the Usability Score. In this thesis we used the framerate and the LoC as proof of concept for the use of our framework.

At the bottom right corner of the figure we see the pre and post optimisation results for the two metrics applied. Regarding the framerate results, both the numbers and the bar charts indicate that all three optimisation techniques had positive impact to the initial implementation. The height of the bars shows that the solution using a DirectFB with 2 windows gave the best results compared to the other optimisation solutions with regard to the framerate. On the other hand optimisation B requires the fewest changes in the code, which is beneficial with respect to the preservation of low cost, while optimisation C requires the most. For the final decision making the involvement of the stakeholders is necessary, in order to combine the results of the framework with their experience and select the most beneficial optimisation technique for the specific case.

Chapter 6

Related Work

Our research started with a process of collecting relevant literature, aiming to capture the state of the art. In this chapter we will present the methodology we used for collecting some related studies and we will present them mentioning their similarities and differences compared to our work.

The literature research began from the very first days of this project and grew iteratively. Since our focus was not to conduct a complete systematic literature review, but to find relevant previous works on the domain of our interest, we studied certain well-known methods for systematic literature reviews in software engineering [BKB+07, KC07]. However we only used parts of these structured methods which would be helpful according to our needs. After completing the first round of this literature research, we also applied the backward snowballing method as discussed in [KC07, WW02], searching for interesting references of the selected related works.

First of all we started our research by specifying the sources in which we would search, the keywords we would use and the inclusion and exclusion criteria for our research.

The selection of the sources was made based on [KC07]. Thus we searched in eight digital libraries; IEEExplore, ACM Digital library, SpringerLink, SCOPUS, Google scholar, Citeseer library, EI Compendex and Inspec. The keywords we used were the phrases: "market driven" AND "embedded systems" AND "quality attributes" AND optimisation, and they were extracted from the title of the thesis. The basic inclusion criteria were that the studies had to be written in English language and they had to deal with introducing a framework for eliciting and handling quality requirements deriving from different stakeholders. Thus papers dealing generally with requirements engineering processes and models, or metrics and quality models were excluded.

During the first round of our literature research we found 47 results, the majority of which can be categorized in three categories; those investigating the requirements management process and the difficulties of capturing requirements correctly in a variety of domains, those dealing generally with decisions related to software product lines engineering and those focusing on architectural decisions. Finally, five of these studies are closely related to our approach and our research interests. By researching the references of the selected papers we came up with a total of 15 related studies dealing with the elicitation of requirements from different stakeholders and the alignment of their opinions for taking decisions about the final product, that we are going to present in the rest of this section.

In [KvST97] the authors present a model called multi-party chain model, which charts the different views of internal and external stakeholders regarding the quality characteristics of a product. Their aim is to trigger a communication channel between the stakeholders in order to achieve mutual understanding and consensus of the different points of view. The model investigates the various interpretations of quality among the different roles in order to define quality metrics that will bridge the understanding of quality requirements between customers and engineers. On the contrary to our approach, the multi-party chain model is more interested in the alignment of stakeholders and thus focuses in capturing the parties and the roles involved as well as the common or conflicting interests. In our approach this information is not of high importance. At the final steps of our framework, when the company is about to make the optimisation decisions, we are also presenting the conflicting benefits of the applied optimisation techniques among different products, but at that point the stakeholders' interest is only indirectly connected with the quality improvement decision that has to be made. Another worth mentioning difference is that this study describes the development process of already existing products with real customer needs and thus the model starts by measuring the quality of the product before triggering the discussion between the specific needs of the stakeholders. On the other hand, our approach is applicable to a more abstract software development environment where there are no real customers but only potential market needs. In such a case the multi-party chain model would be more difficult to be applied since the quality of the system is more loosely defined in such circumstances.

In [KvST99a] the same multi-party chain model is compared to a questionnaire based approach aiming to the identification of important quality attributes. However that approach does not involve all the stakeholders in the elicitation process of the quality requirements but makes

use of a questionnaire where the ISO terminology is used to extract requirements. Both these characteristics of this approach differentiate it from our approach.

In [KvST99b] the authors compare two strategies for capturing the needs of software quality in the embedded domain. They state that ensuring software quality in embedded systems is difficult due to the reason that the most important goals are time-to-market and cost. This makes software characteristics like reliability, usability and maintainability second priority targets. As software quality they define a set of quality characteristics which can be found in standards like ISO 9126 and the IEEE software engineering standards collection. The benefit of using quality standards is that you can provide well defined quality characteristics to the different stakeholders and achieve the mitigation of having different interpretations. The common points between our approach and theirs are the collection of data through interviews with different stakeholders, the matching of data with ISO quality characteristics and lastly the prioritization of those characteristics. In our case we proceed by using the outcome of the prioritization to drive our optimisation decisions.

In [HJBP98] the authors present another model aiming to improved decision making after capturing the knowledge of experienced personnel both in managerial and technical positions. However this model focuses on the product/process dependencies (PPDMs) in cases of product-driven software development in contrast to our market-driven approach aiming to help the stakeholders' decision making with respect to optimisation needs. Although the goal of the model is different than the goal of our framework, we can identify some similarities. For example the PPDM approach investigated the dependencies of the product quality attributes, as they are defined in the ISO 9126. In our framework we link the captured product quality requirements with the quality attributes as defined in ISO 25010, which is an update of ISO 9126 [ISO11]. Also both frameworks use the Goal-Question-Metric (GQM) approach. In our framework the approach is suggested in order to identify useful metrics for taking initial and post optimisation measurements of the system, while in the case of PPDMs is used for generating the product/process dependency hypothesis as well as analysing and validating it.

In [Bar11] the author has two major objectives. First of all to define a method which determines the alignment between stakeholders when they are asked to prioritise software quality aspects and secondly to identify the factors which affect the results of the method. The method proposed is called "Stakeholder Alignment Assessment Method for Software Quality (SAAM-SQ)" and consists of seven steps; Selection of a company and a product, identification of success-critical stakeholder groups, development of a quality model, development of a questionnaire, conduct

the questionnaire analysis of the results and organisation of a workshop on the results. The major differences of this approach with our framework is first of all that this study basically focuses on the stakeholders' alignment while our major focus is on the software quality aspects that should be optimised based on the market needs. Additionally the SAAM-SQ method suggests the use of a questionnaire, which is related to the important software quality attributes for the study, for the prioritisation of the quality aspects. On the contrary, in our approach we extract the quality aspects that the stakeholders are interested in through the process of interviews, we capture them in the form of quality attributes and we analyse them using the Requirements Abstraction Model (RAM) method [GW05] making it easier to connect them to the ISO software quality model and consequently to a number of relevant software metrics. Our process requires more advanced analysis of the input data, since they are loosely related to the quality attributes, but it is expected to be easier for the stakeholders since their participation to a semi-structured interview and the prioritisation of quality Product Level requirements is expected to be an easier process than the application of the HCV (Hierarchical Cumulative Voting) method selected from the authors in the related case studies applied [BWCA11].

According to [Bar11] there are several methods for merging perspectives on software quality. Four of these methods are the expert judgement, the Non-Functional Requirements (NFR) Framework, the Quality Functional Deployment (QFD) and the Theory-W. The expert judgement involves one or more professionals and suggests decision making based on their experiences, without any modelling or numerical assessment. The NFR Framework [CNY00] uses a structured graphical representation of quality requirements for simplifying their management, refinement and interrelations, aiming to enhance the decision making process in the design and operation of a system. The QFD method focuses on the opinion of the customers and users aiming to achieve a good prioritisation of the system's goals [HSP03]. Also in [KOR97] the authors present a QFD method for requirements management. The method focuses on the customers and the users in order to provide an effective way for prioritizing and communicating quality software requirements. However the basic difference between this approach and our framework is that this approach can be applied only when specific customers and users exist and thus it cannot be used in new projects or market-driven development cases like ours. In [MH98] the QFD approach is combined with customer satisfaction to provide a more systematic connection of the two. Finally the Theory-W [BR89] has as principal to make all the stakeholders feel "winners" in the software process. The method includes success-critical stakeholder identification and requirements elicitation as well as negotiations between the manager and the various

constituencies to create a win-win situation for all groups. The resulting benefit is the mutual understanding of the existing needs between groups with common goals.

In [Sve11] the author is focusing on the management of quality requirements in the case of market-driven software development. The goal of the research is to enhance high level decision making by achieving early accurate estimations on aspects like performance. The QUPER (Quality Performance) method is proposed for cost-benefit analysis of quality requirements and is used to improve the requirements prioritization and quality requirements road-mapping, during the initial release planning phases. However although the study is proposing a framework for handling quality requirements in the context of market-driven software development the main research interest of the study is focusing towards the cost-benefit scale, rather than the optimisation needs that is the interest of our thesis.

In [SGR09, SGR+12] the authors investigate the challenges of quality requirements management in market-driven development cases. Their approach suggests the elicitation of requirements through semi-structured interviews and the identification of the most important quality aspects, which are derived after the connection of the requirements with the ISO quality attributes. However, although there are many similarities between this approach and the first steps of our framework, the objective of these studies differ a lot from ours. These studies are interested in capturing the differences between companies working in B2B and B2C markets with respect to the handling of quality requirements, the examination of the interdependencies between requirements and generally the quality requirements management in down-stream development activities.

In [DKK+05] the authors propose an NFR method to elicit, document, and analyse quality requirements through a series of workshops with stakeholders. The paper suggests that initially the important quality attributes shall be elicited during workshops while at the same time quality requirements should be gathered about them. These quality attributes bridge the gap between requirements and existing quality models, which also provide a set of metrics useful for measuring the quality of a system. However a difference to our approach, is that in this study it is suggested that the quality requirements shall be gathered based on the identified important quality attributes, while in our framework we apply the connection with the quality attributes independently and after the requirements elicitation has been completed, in order to avoid any biased answers. Also this difference is connected with the focus of our framework to capture the different needs between different stakeholders or even different versions of a product and their

connection to the decision making process regarding the optimisation methods that should be applied. The focus of this paper is only up to the point of analysis of the quality requirements.

Chapter 7

Discussion

In this chapter we will discuss the major findings of this thesis answering each research question. Additionally we will present the threats to validity for our study. Finally we will elaborate on the lessons learned as well as some recommendations for future related work.

7.1 Discussion per Research Question

Until now we have presented and analysed all the results and findings of this case study according to the structure of the proposed framework. This chapter aims to interpret the findings per research question, evaluate the findings in the light of previous research and address how this report adds to the research being done in the field. In this way we will ensure the clear understanding of how the research questions are answered as well as the contribution of this thesis to the academic community.

7.1.1 RQ1: How can market needs be used to identify important software quality attributes?

In this thesis we propose a market-driven framework for guiding the optimisation decisions in the domain of embedded systems and we test its application in the case of the Populus tool extension with an HTML component in Mecel AB.

The first steps of the framework require the extraction of the market needs from stakeholders, through semi-structured interviews. After the extraction and documentation of the requirements,

the framework provides a structured way for connecting the requirements with the ISO quality attributes aiming to set a bridge for a connection with applicable, useful metrics, that will be able to provide us information about the state of interesting characteristics of our system. This process involves the application of the RAM method [GW05] for working up the requirements in a higher level of abstraction that will be processed and linked to the ISO quality attributes.

According to the literature research that we performed during this study, there are several works on the domain of requirements elicitation from different stakeholders with focus on the alignment of their opinions for taking decisions about a final product [Bar11, BR89, BWCA11, CNY00, DKK+05, HJBP98, HSP03, KOR97, KvST97, KvST99a, KvST99b, MH98, SGR09, SGR+12, Sve11]. The basic benefit of our approach is "the enhancement of communication and the definition of a common ground as basis for high quality requirements engineering" as also stated in [DKK+05] about their NFR method. However, although we identify some similarities with previous works, as analytically discussed in Chapter 6, the combination of already existing widely approved methods and techniques (like the RAM and GQM methods, the ISO quality model and the use of documented metrics) is the strong characteristic of this framework.

7.1.2 RQ2: How can the optimisation process of a product be driven by the software quality attributes from RQ1?

Our framework aims at the connection of the market needs with the optimisation process. As described in the previous section, its first steps aim at transforming the captured market needs into ISO quality attributes. The connection of the ISO quality attributes with the concept of metrics is well known almost by definition, as it is introduced in the ISO 9126 [ISO01]. However, our framework suggests the use of the GQM method for identifying applicable metrics for the needs of each case study. A key factor for driving the optimisation process based on the market needs of RQ1 is the prioritisation of the identified quality attributes by the stakeholders. This prioritisation indicates what is considered to be the most important optimisation aspects and can help handling any potential decision regarding the existing trade-offs, after the optimisation techniques have been applied and the pre and post optimisation results have been collected.

During our literature research we did not manage to find any related work investigating the connection of software quality attributes with the optimisation process in the domain of market-driven development. Thus the contribution of our framework is a complete guideline from the

requirements elicitation process to the decision making process after the optimisation process has been completed.

7.2 Limitations and Threats to Validity

Construct validity:

"This aspect of validity reflects to what extent the operational measures that are studied really represent what the researcher have in mind and what is investigated according to the research questions." [RH09]

In our case study the sample of interviewees was small (3 people) due to the availability of people with respect of time and relevant knowledge to the topic. As mentioned in Chapter 5 it is suggested to conduct enough interviews until the point of "saturation" for optimum results [RH09]. However, in our case an effort was made to conduct interviews with the most applicable candidates, expecting that they would be able to give us a complete picture about potential needs.

During the elicitation and prioritisation process we tried to keep the stakeholders unbiased. For example we extracted their prioritization results without them knowing the initial measurements of the implementation. However our supervisor at Mecel was one of the stakeholders and due to his close relation with the project we cannot consider his opinion unbiased. Additionally the fact that only internal stakeholders were involved increases the risk of capturing limited points of view, since all stakeholders could be influenced by the company's policy.

Since there was no previous HTML extension of the product, the stakeholders were often confused during the interviews when asked questions about it. To mitigate this threat we had to frequently remind them that the purpose of the interviews was to capture the requirements for such a potential product. In addition to this lack, there was also no customer interested for such an extension. For these reasons we consider the environment of this study as an extreme case for applying our framework, although it could be frequent in cases of market-driven development.

Finally the metric for the framerate of the implementation was created by us. Since we used no third party tools except from the documentation of WebkitGtk we based all of our calculation on how the documentation describes the methods and objects.

Internal validity:

"This aspect of validity is of concern when causal relations are examined. When the researcher is

investigating whether one factor affects an investigated factor there is a risk that the investigated factor is also affected by a third factor." [RH09]

The environment we used for developing our product was a virtual machine. As such, it was influenced by the work load and the processes of the host operating system. We know that this issue had an impact on the measurements we took and thus could have affected the results for the implementation/optimisation.

Also, during the measurement of the framerate we noticed that unintentional movements of the mouse affected the results. To mitigate the problem we tried to create a stable environment to run our measurements. Thus we not only gathered the measurements when running the same HTML page, but also we tried to minimise the external factors that could influence the results. Firstly we measured the framerate of each implementation just after restarting the virtual machine. The reason was that we wanted all implementations to run under a fresh environment since it was noticed that running the virtual machine for a long duration could affect our results. Additionally we made sure that no other processes, that would increase the workload of the virtual machine, were started by us during the extraction of the measurements. We also ensured that no interactions from our side, like mouse movements or typing, were taking place during the measuring process.

External validity:

"This aspect of validity is concerned with to what extent it is possible to generalize the findings, and to what extent the findings are of interest to other people outside the investigated case." [RH09]

The techniques we used for the optimization of the initial implementation were very specific to our case and thus we do not expect them to be universally applicable for optimising products in different cases. Moreover in our case we had a simple implementation. We only had one product, one test example and two metrics. Thus further investigation is needed for finding out if our framework would be applicable to more complex cases or to other domains too, rather than only for IVI embedded systems.

Reliability:

"This aspect is concerned with to what extent the data and the analysis are dependent on the specific researchers." [RH09]

According to [KvST99b] a threat arising from the interview method we applied is the high level of expertise needed by the interviewers. This is necessary not only to avoid using the ISO terminology directly, but in the same time capture the needs in a way that they can be correctly interpreted later on. To mitigate the risk of incomplete requirements elicitation we involved the stakeholders into a validation process, giving them the opportunity to provide us with feedback regarding the captured requirements. Additionally another risk was our inexperience in the requirements elicitation and documentation process. For this reason we also asked an experienced employee of Mecel, to validate that the captured requirements were sufficiently documented.

7.3 Recommendations and Lessons Learned

One of the risks of this thesis is the fact that no initial implementation of HTML extension existed. This situation, as mentioned in the previous section, is considered an extreme scenario to test the validity of our framework. The fact that the framework is focused on driving optimisation decisions makes the lack of an implementation a very important factor. Thus we recommend the completion of the implementation in advance to the use of the framework, since this would increase the stakeholders' understanding regarding the optimisation needs of the product.

Another issue is the boundaries between the implementation and the optimisation. When having to do both, it is often unclear to the developer when the implementation is considered complete and when the optimisation should begin. This issue affects mostly the decision making process e.g. how many features to include initially, how many systems to support.

Moreover we learned that for such a project, that provides an implementation of a product and a framework for optimising it, much time is needed for testing complex use cases. In our thesis we had time to explore only one simple scenario which included HTML content rendering and to measure only one quality attribute, that of performance.

As a recommendation for similar projects we propose the continuous and close communication with the stakeholders. The feedback from the stakeholders helped us during every step of the framework, mainly during the requirements elicitation and the implementation phase.

Chapter 8

Conclusion

This thesis presents a market-driven framework for guiding optimisation decisions on the embedded systems domain. It is described how such a framework can be used by people in the domain, pinpointing the benefits and the risks. As part of our case study we validated it against Mecels Populus, where we extended the current HMI tool suite with an HTML component and optimised it based on the market needs. By using the proposed framework we achieved the increase of the software quality of the product in terms of Performance Efficiency. Summing up the findings of the study we conclude that the use of the framework is of value in market-driven cases of the embedded systems domain and that the market needs can successfully be connected with software quality attributes in order to influence the optimisation decisions.

Chapter 9

Future Work

As a continuation of our work the framework could be evaluated by being used in more complex use cases or even in use cases of different domains. It would also be interesting to use the framework in a situation where a larger number of stakeholders is available for prioritizing the requirements, in order to examine how potentially controversial opinions could affect the optimisation decisions. Alternatively an other field for investigation could be a case of Software Product Lines Engineering, were it would be expected that more than one similar products would be extracted by the interviews and thus would have to be handled by the framework. In such a case, the decision making would have to handle the trade-offs, regarding all the different products and how they are affected by the optimisation techniques applied. For example a full version of a product could be optimised by an optimisation on feature X, while the same time the free version of the product which would not provide the specific feature would have a negative impact, as a result of the overhead of the optimisation method. In such a case the stakeholders would have to decide which product is more important based on the marketing strategy of the company. If this decision is possible to be captured as a weight of importance between the product, this information could be added in the framework, so that to simplify the stakeholders' task. Finally, another interesting aspect for Software Engineering could be the automation of the optimisation process when a new product is introduced e.g. in the Software Product Lines example above, it would be interesting to investigate the scenario of identifying automatically the commonalities of a new product with the already processed products of the same product family in order to reuse the existing test examples and test scripts for the common requirements. Such an automation would be expected to reduce in many cases the time and effort needed to apply the suggested framework.

Appendix A

Requirements Specification Document

Requirements Specification Document

The requirements have been extracted from the interviews with internal stakeholders and aim to specify what the market would expect from the extension of Populus with an HTML component. The fact that there are no real customers for the specific product triggered our interest to investigate what the different stakeholders inside Mecel consider of big importance for a product like this.

Although there are no real customers for this case study, we know from the interviews that the potential customers can be of two types, either the end users or the HMI developers of the supplier companies. Thus in this specification document we will refer to users or developers respectively.

The extracted requirements have been processed according to the Requirements Abstraction Model[1] in order to achieve same level of abstraction for all requirements. The levels of abstraction are three; Product level, Feature level and Function level starting from the higher to the lower level of abstraction.

Each requirement has a unique code consisted of four parts:

1. The first part of the code specifies if the requirement is an original requirement (OR) or has been created based on the needs of the methodology (CR).

2. The second part of the code is the identification number of the requirement (1-14). This number introduces a category of requirements in different abstraction levels that are under the same Product Level.

3. The third part of the code is a number differentiating the requirements of the same abstraction level, in cases that in the same requirement category there are more than one requirements in the same abstraction level. In cases that there is only one requirement in an abstraction level, this number is omitted.

[1] Tony Gorschek and Claes Wohlin. 2005. Requirements Abstraction Model. *Requir. Eng.* 11, 1 (December 2005), 79-101.

4. Finally the fourth part of the requirement specifies the abstraction level of the requirement; Product level (PL), Feature level (FeL) and Function level (FuL).

The requirements below are presented according to this abstraction hierarchy. The Original requirements are further analysed to assure that they have been captured and understood correctly, while the Created ones consist only of a short description.

CR1.PL – Product Level

The product shall provide a basic functionality.

OR1.1.FeL – Feature Level

1. Title

HTML rendering

2. Description

The system shall be able to render HTML content.

3. Reason / Benefit / Rationale

WHY: In order to show the content of online/offline HTML pages

BENEFIT: Extension of the product functionality

4. Restriction/ Risks

Introduces web security risks.

5. Requirement Source

Marketing Departement

Populus Code Developer

6. Dependency

No

OR1.2.FeL – Feature Level

1. Title

Display the content of a URI

2. Description

The system shall be able to navigate to specific predefined URIs.

3. Reason / Benefit / Rationale

WHY: In order to allow the rendering of content from different pages

BENEFIT: Allow navigation from one page to another

4. Restriction/ Risks

-

5. Requirement Source

Populus Code Developer

6. Dependency

No

OR1.3.FeL – Feature Level

1. Title

Reading functionality

2. Description

The product shall be able to read from a disk/flash.

3. Reason / Benefit / Rationale

WHY: The HTML content to be rendered shall be possible to be stored in a flash or disk and the product shall be able to read that saved content.

BENEFIT: No need for constant conection to the internet. Also quicker way to show predefined content (loading from the disk is usually quicker than loading from the internet).

4. Restriction/ Risks

-

5. Requirement Source

Populus Code Developer

6. Dependency

No

CR1.4.FeL – Feature Level

The user shall be able to navigate in the rendered HTML content of a page.

OR1.4.FuL – Function Level

1. Title

Interaction with page content

2. Description

The user shall be able to interact with the HTML content of a given page, using buttons or links.

3. Reason / Benefit / Rationale

WHY: In cases that the content of a page has components that require interaction with the user

BENEFIT: Allows interaction

4. Restriction/ Risks

It should be assured that the predefined content is not connected (through links or buttons) with any untrustworthy pages or content that is not supported by our HTML component.

5. Requirement Source

Populus Code Developer

6. Dependency

No

OR1.5.FeL – Feature Level

1. Title

Page scrolling

2. Description

The product shall provide page scrolling functionality

3. Reason / Benefit / Rationale

WHY: In cases that the HTML content is larger than a page the user wants to be able to access the whole content.

BENEFIT: Allows access to larger HTML content, without any trade off regarding the legibility of the content.

4. Restriction/ Risks

The scrolling functionality shall be responsive so that it does not affect negatively the sense of good usability.

5. Requirement Source

Populus Code Developer

6. Dependency

No

CR2.PL – Product Level

The product shall be responsive

CR2.1.FeL – Feature Level

The product shall provide quick HTML content rendering

OR2.1.FuL – Function Level

1. Title

Rendering load time specification

2. Description

The rendering engine shall render the HTML content of a page in less than 100ms. Such a performance is expected when loading a small page (fitting in the screen) containing only text and up to two buttons provided that the HTML is on disc so that it does not include network load times.

3. Reason / Benefit / Rationale

WHY: Long loading time would make the use of the rendering engine time consuming and would create a feeling of bad responsiveness of the system resulting to unsatisfied customers.

BENEFIT: Good responsive behavior of the system

4. Restriction/ Risks

-

5. Requirement Source

Populus Code Developer

6. Dependency

OR1.1.FeL

CR2.2.FeL – Feature Level

The product shall provide links/buttons which respond quickly

OR2.2.FuL – Function Level

1. Title

Link or button response specification

2. Description

The rendering engine shall respond in a click of a button or link within 100ms. In this time we expect that an event will be triggered and the user will get some feedback showing that the button has been pressed successfully (ex. change of link color)

3. Reason / Benefit / Rationale

WHY: A slow response of a button that has been pressed would give false impression to the user that the button has not been successfully pressed. As a result the user would probably try to press the button again slowing down even more the processing.

BENEFIT: Good responsive behavior of the system

4. Restriction/ Risks

-

5. Requirement Source

Populus Code Developer

6. Dependency

OR1.4.FuL

OR2.3.FeL – Feature Level

1. Title

Disk/Flash reading response

2. Description

The product shall be able to read quickly from external devices

3. Reason / Benefit / Rationale

WHY: The rendering latency of stored content will have an impact on the overall performance of the product

BENEFIT: By increasing the reading speed we increase the overall performance of the product

4. Restriction/ Risks

-

5. Requirement Source

Marketing Department

6. Dependency

OR1.3.FeL

OR2.3.FuL – Function Level

1. Title

Disk/Flash delay specification

2. Description

The system shall be able to read from the disk/ flash in less than 100ms

3. Reason / Benefit / Rationale

WHY: The rendering latency of stored content will have an impact on the overall performance of the product

BENEFIT: By increasing the reading speed we increase the overall performance of the product

4. Restriction/ Risks

-

5. Requirement Source

Populus Code Developer

6. Dependency

OR1.3.FeL

CR2.4.FeL – Feature Level

The product shall provide a quick page scrolling feature

OR2.4.FuL – Function Level

1. Title

Scrolling framerate specification

2. Description

The rendering engine shall be able to scroll the content of a web page with framerate higher than 10 f/s.

3. Reason / Benefit / Rationale

WHY: Slow scrolling performance would make the use of the renedering engine time consuming and would create a feeling of bad responsiveness

BENEFIT: Good responsive behavior of the system

4. Restriction/ Risks

-

5. Requirement Source

Populus Code Developer

6. Dependency

OR1.5.FeL

CR2.5.FeL – Feature Level

The product shall provide a smooth UI

OR2.5.FuL – Function Level

1. Title

UI smoothness specification

2. Description

The UI of the rendering engine shall have a minimum framerate 10f/s. To consider the UI as smooth the framerate shall be higher than 30f/s

3. Reason / Benefit / Rationale

WHY: To provide the feeling of a well responsive UI

BENEFIT: Enhance user experience

4. Restriction/ Risks

-

5. Requirement Source

Populus Code Developer

6. Dependency

No

CR2.6.FeL – Feature Level

The product shall have a quick start up time

OR2.6.FuL – Function Level

1. Title

Start up time specification

2. Description

The system shall be able start up in less than 10s

3. Reason / Benefit / Rationale

WHY: A slow start up time would give the false impression to the user that the system has not been successfully started.

BENEFIT: Good responsive behaviour of the system

4. Restriction/ Risks

We should ensure that the product dependencies can also start in the same time

5. Requirement Source

Populus Code Developer

6. Dependency

No

OR3.PL – Product Level

1. Title

Lightweight product

2. Description

The product shall be lightweight

3. Reason / Benefit / Rationale

WHY: Good performance on low cost, low end platforms is one of the most advertised features that characterise the product

BENEFIT: Market competitive advantage

4. Restriction/ Risks

A lightweight product supports fewer features

5. Requirement Source

Marketing Department

Populus Code Developer

6. Dependency

No

OR3.FeL – Feature Level

1. Title

Low memory consumption

2. Description

The memory consumption of the product shall be low.

3. Reason / Benefit / Rationale

WHY: The hardware platforms which are usually used for this purpose do not provide high memory resources.

BENEFIT: Compatibility with low cost hardware platforms

4. Restriction/ Risks

The product would be able to support heavy memory consumption features

5. Requirement Source

Online Services Component Owner

6. Dependency

No

OR3.FuL – Function Level

1. Title

Memory consumption specification

2. Description

The memory consumption shall be limited to lower than 100MB.

3. Reason / Benefit / Rationale

WHY: The hardware platforms which are usually used for this purpose do not provide more resources.

BENEFIT: Compatibility with low cost hardware platforms

4. Restriction/ Risks

-

5. Requirement Source

Populus Code Developer

6. Dependency

No

CR4.PL – Product level

The product shall be portable.

OR4.1.FeL – Feature level

1. Title

Portability between Operating Systems

2. Description:

The product shall be able to run on multiple Operating Systems.

3. Reason / Benefit / Rationale

WHY: There are many platforms with different operating systems used for this purpose.

BENEFIT: The integrated rendering engine will be functional on all the Operating Systems used.

4. Restriction/ Risks

At the moment the Operating Systems of interest are only Linux and Windows and thus we will restrict the portability test only for these two.

5. Requirement Source:

Populus Code Developer

6. Dependency

No

OR4.2.FeL – Feature Level

1. Title

Portability between hardware platforms

2. Description:

The product shall be able to run on multiple hardware platforms

3. Reason / Benefit / Rationale

WHY: There are many platforms with different characteristics and costs used for this purpose.

BENEFIT: The integrated rendering engine will be functional on all the hardware platforms used.

4. Restriction/ Risks

At the moment the hardware platforms used are only x86 and ARM and thus we will restrict the portability test only for these two.

We should ensure that all changes applied will work for all platforms.

5. Requirement Source:

Populus Code Developer

6. Dependency

No

CR5.PL – Product Level

The user shall be able to update the product features by themselves

OR5.1.FeL – Feature Level

1. Title

Search new applications

2. Description:

The user shall be able to search for new applications in the supplier's web page after the product is sold

3. Reason / Benefit / Rationale

WHY: The user should always be able to search for applications that the car supplier provides and are interesting for him/her to use.

BENEFIT: Easy and quick information about existing applications provided by the supplier

4. Restriction/ Risks

Searching for applications shall be allowed only in the supplier's web page for safety reasons.

Internet connection will be required for searching.

Secure internet connection shall be ensured.

5. Requirement Source:

Marketing Department

Populus Code Developer

6. Dependency

No

OR5.2.FeL – Feature Level

1. Title

Download new applications

2. Description:

The user shall be able to download new applications from the supplier's web page after the product is sold

3. Reason / Benefit / Rationale

WHY: The user should have access to acquire new applications that the supplier provides. These new IVI applications provided by the supplier can be downloaded on the car quickly and easily by the user.

BENEFIT: The functionality of the product is extended by those applications.

Potential profit.

4. Restriction/ Risks

Applications shall be downloaded only from the supplier's web page for safety reasons.

Internet connection will be requires for downloading.

Secure internet connection shall be ensured.

5. Requirement Source:

Marketing Department

Populus Code Developer

6. Dependency

No

OR5.3.FeL – Feature Level

1. Title

Delete applications

2. **Description:**

The user shall be able to delete existing applications.

3. **Reason / Benefit / Rationale**

WHY: The product has limited storage space

BENEFIT: Storage space management provision.

4. **Restriction/ Risks**

Only non-core applications shall be possible to uninstall

5. **Requirement Source:**

Marketing Department

6. **Dependency**

No

CR6.PL – Product Level

The product shall be able to operate with other systems

OR6.1.FeL – Feature Level

1. **Title**

Safe systems coexistence

2. **Description**

The rendering of the HTML context shall not interfere with any other system in the car

3. **Reason / Benefit / Rationale**

WHY: Using shared resources can have negative impact on the functionality of critical systems

BENEFIT: Protect stability of critical systems and enhance safety.

4. Restriction/ Risks

We should reserve and limit resources for the rendering process in advance and these resources will be constantly reserved even when no rendering is performed.

The static allocation of resources increases the chances of crashing while rendering heavy context.

5. Requirement Source

Marketing Department

6. Dependency

No

OR6.2.FeL – Feature Level

1. Title

Safe browsing

2. Description

The user shall not be allowed to browse uncontrolled web content for safety reasons. Only connections provided by the customer backend shall be allowed.

3. Reason / Benefit / Rationale

WHY: To control the content that the user can access.

BENEFIT:Increase security and stability

4. Restriction/ Risks

The user is not allowed to access any unauthorised web page.

Limited content

5. Requirement Source

Online Services Component Owner

6. Dependency

No

OR7.PL – Product Level

1. Title

Development time reduction

2. Description

The supplier wants to use the product for creating quickly HMIs or applications using HTML which will be rendered by the integrated rendering engine of the product.

3. Reason / Benefit / Rationale

WHY: The supplier wants the product to help reduce development time

BENEFIT: Existing third party HTML applications and HMIs will be possible to be reused

4. Restriction/ Risks

-

5. Requirement Source

Marketing Department

6. Dependency

No

CR8.PL – Product level

The product shall be configurable.

OR8.1.FeL – Feature level

1. Title

Feature configuration capability

2. Description:

The features of the product shall be easily configurable based on the needs of different customers.

3. Reason / Benefit / Rationale

WHY: There are different customers with different customer needs.

BENEFIT: Capability to meet the customer needs quickly through code reusability.

4. Restriction/ Risks

This can happen only to some extent and often it may require changes that are difficult to be met for new customer needs.

5. Requirement Source:

Online Services Component Owner

6. Dependency

No

OR8.1.FuL – Functional Level

1. Title

Behaviour Configuration

2. Description:

The user shall be able to configure the functionality of the HTML of the product based on the existing needs.

3. Reason / Benefit / Rationale

WHY: There are different customers which require different functional characteristics at the HTML component. For example, the scrolling feature shall be able to be configured as line by line or per page.

BENEFIT: The product will provide customisation capabilities, making it usable for different needs.

4. Restriction/ Risks

-

5. Requirement Source:

Online Services Component Owner

Populus Code Developer

6. Dependency

No

OR8.2.FuL – Functional Level

1. Title

Appearance Configuration

2. Description:

The user shall be able to configure the appearance of the HTML of the product based on the existing needs and their personal taste.

3. Reason / Benefit / Rationale

WHY: There are different customers which require different visual characteristics at the HTML component.

BENEFIT: The product will provide customisation capabilities, making it usable for different needs.

4. Restriction/ Risks

-

5. Requirement Source:

Online Services Component Owner

6. Dependency

No

CR9.PL – Product level

The product shall promote driving safety.

OR9.FeL – Feature level

1. Title

Handling of distractive HTML content.

2. Description:

The content displayed in the browser window shall not be interactive when the car is in motion, so that to eliminate the chances to distract the driver's attention while driving.

3. Reason / Benefit / Rationale

WHY: Interactive HTML content can distract the driving concentration and become dangerous for the driver's physical integrity

BENEFIT: Driving Safety

4. Restriction/ Risks

In order to achieve non distractive content the original content should be altered. There is a risk of loss of information

5. Requirement Source:

Online Services Component Owner

6. Dependency

No

OR10.PL – Product Level

1. Title

Low cost focus

2. Description

The product cost shall be preserved low

3. Reason / Benefit / Rationale

WHY: There are customers whose priority is to preserve the cost of the product low when extending its features.

BENEFIT: Market competitive advantage

4. Restriction/ Risks

-

5. Requirement Source

Marketing Department

Online Services Component Owner

6. Dependency

No

OR11.PL – Product Level

1. Title

HTML5 feature support

2. Description

The product shall support HTML5 features in order to be able to render HTML5 content

3. Reason / Benefit / Rationale

WHY: It brings HTML up to date with extended functionality (new tags and enhancements for a wide range of features) making it easier for developers to deliver what the users want better and faster.

BENEFIT: Marketing advantage for attracting suppliers who are interested to use the new HTML5 features

4. Restriction/ Risks

Necessary overhead to support the HTML5 features

5. Requirement Source

Marketing Departement

6. Dependency

No

OR12.PL – Product Level

1. Title

Cool features support

2. Description

The product shall contain cool infotainment features

3. Reason / Benefit / Rationale

WHY: The user wants to experience the wow effect on the product

BENEFIT: Maket competitive advantage

4. Restriction/ Risks

The Wow features shall be in accordance with the other functional and non-functional requirements (e.g. shall not cause the driver distraction, shall not interfere with the other IVI systems etc.)

5. Requirement Source

Marketing Department

Online Services Component Owner

6. Dependency

No

OR12.1.FeL – Feature Level

1. Title

Multiple units use

2. Description

The product shall be able to be shown in multiple units

3. Reason / Benefit / Rationale

WHY: The user wants to show the product features in screens different than the car cluster

BENEFIT: Utilisation of multiple screens allows more content to be shown that visually impresses the user

4. Restriction/ Risks

-

5. Requirement Source

Online Services Component Owner

6. Dependency

No

OR12.2.FeL – Feature level

1. Title

Media format support

2. Description:

The product shall support a variety of media formats

3. Reason / Benefit / Rationale

WHY: The user shall be able to receive content in many different formats.

BENEFIT: Increase the amount of content that can be rendered by the product.

4. Restriction/ Risks

This can happen only to some extent and often it may require changes that are difficult to be met for new customer needs.

5. Requirement Source:

Online Services Component Owner

6. Dependency

No

CR12.3.FeL – Feature Level

1. Title

3D rendering

2. Description

The product shall support rendering of 3D animation

3. Reason / Benefit / Rationale

WHY: The HTML content of a page may require 3D rendering functionality

BENEFIT: Increased capability to render pages with such HTML content.

4. Restriction/ Risks

-

5. Requirement Source

Marketing Department

6. Dependency

No

CR13.PL – Product level

The product shall have two versions.

OR13.1.FeL – Feature Level

1. Title

Widget support

2. Description

The product shall provide an advanced rendering and scripting engine, for supporting third party content in the form of multiple widgets.

3. Reason / Benefit / Rationale

WHY: To enhance security and ensure that the rendered content follows specific standards

BENEFIT: Standards compliant predictable behavior

4. Restriction/ Risks

The available content is restricted to third party content only.

Standards overhead to ensure that the available content follows the standard.

5. Requirement Source

Populus Code Developer

6. Dependency

No

OR13.2.FeL – Feature Level

1. Title

Full functional browser

2. Description

The product shall provide web browsing functionality to web sites that are compatible to the HTML web standards.

3. Reason / Benefit / Rationale

WHY: The product would gain value by allowing the users to navigate to their favorite web sites from the vehicle system.

BENEFIT: Increase the connectivity of the product

4. Restriction/ Risks

Safety restrictions shall be taken into consideration

5. Requirement Source

Marketing Departement

Populus Code Developer

6. Dependency

No

Appendix B

Interview Questions

1. Can you give us a small introduction to your job tasks and your background? (How strong managerial, marketing or technical background and what responsibilities does the job have?)
2. Which are your customers?
    - Which company, what department?
    - How are they interested to your products?
    - Can you define split them in different categories according to their needs?
3. What is the main interest of the customers for the HMI suite (and in extend to the HTML component)?
    - High performance?
        - Time?
        - Quality of graphics?
    - Low energy consumption?
    - Robustness?
        - Loose coupling?
    - Low cost?
        - Memory usage?
        - Disk space?

- other?

4. Why do you think HTML could be useful to be integrated with Populus?
5. Who mentioned the need for HTML components in Populus? (customers, developers, management,...)
    - How many stakeholders have been asked?
    - How many stakeholders mentioned the need?
6. Why did they want HTML components?
    - Stand alone HTML components (Browser,SOA service provider etc...)?
    - Complementary to another component?
    - Other ideas?

Bibliography

[Bar11] S. Barney. *Software Quality Alignment: Evaluation and Understanding*. PhD thesis, Blekinge Institute of Technology, Department of Computer Science, Sweden, July 2011.

[BC05] D. Bovet and M. Cesati. *Understanding The Linux Kernel*. Oreilly & Associates Inc, 2005.

[BCR94] V. R. Basili, G. Caldiera, and H. D. Rombach. The goal question metric approach. In *Encyclopedia of Software Engineering*. Wiley, 1994.

[BD07] J. Bergström and A. Dahlqvist. Besmart - a framework for shifting from bespoke to market-driven requirements engineering. Master's thesis, Blekinge Institute of Technology, Department of Computer Science/Software Engineering, Sweden, 2007.

[BFB91] E. M. Babiker, H. Fujihara, and C. D. B. Boyle. A metric for hypertext usability. In *Proceedings. 9th annual international conference on Systems documentation.*, SIGDOC '91, pages 95–104, New York, NY, USA, 1991. ACM.

[BKB+07] P. Brereton, B. A. Kitchenham, D. Budgen, M. Turner, and M. Khalil. Lessons from applying the systematic literature review process within the software engineering domain. *Journal of Systems and Software*, 80(4):571–583, April 2007.

[BR89] B. W. Boehm and R. Ross. Theory-w software project management principles and examples. *IEEE Transactions on Software Engineering*, 15(7):902–916, 1989.

[Buf13] Direct Frame Buffer. http://directfb.org/, July 2013.

[BWCA11] S. Barney, C. Wohlin, P. Chatzipetrou, and L. Angelis. Offshore insourcing: A case study on software quality alignment. *IEEE 7th International Conference on Global Software Engineering*, 0:146–155, 2011.

[CB95] E. Carmel and S. Becker. A process model for packaged software development. *IEEE Transactions on Engineering Management*, 42(1):50–61, 1995.

[CLC+10] U. B. Correa, L. Lamb, L. Carro, L. Brisolara, and J. Mattos. Towards estimating physical properties of embedded systems using software quality metrics. In *IEEE 10th International Conference on Computer and Information Technology (CIT).*, pages 2381–2386, 2010.

[CNY00] L. Chung, B. A. Nixon, and E. Yu. *Non-Functional Requirements in Software Engineering*. The Kluwer International Series in Software Engineering. Kluwer Academic, 2000.

[CS08] J. M. Corbin and A. L. Strauss. *Basics of qualitative research*. Sage Publ., Los Angeles [u.a.], 3. ed. edition, 2008.

[Dav05] A. M. Davis. *Just Enough Requirements Management: Where Software Development Meets Marketing*. Dorset House Publishing Co., Inc., New York, NY, USA, 2005.

[DKK+05] J. Doerr, D. Kerkow, T. Koenig, T. Olsson, and T. Suzuki. Non-functional requirements in industry - three case studies adopting an experience-based nfr method. In *Proceedings. 13th IEEE International Conference on Requirements Engineering.*, pages 373–382, 2005.

[DT90] M. Dorfman and R. H. Thayer. *Standards, guidelines, and examples on system and software requirements engineering*. IEEE Computer Society Press tutorial. IEEE Computer Society Press, 1990.

[Fea09] WebKit Features. http://trac.webkit.org/wiki/webkitfeatures, April 2009.

[Fou13] Xorg Foundation. http://www.x.org/wiki/, July 2013.

[GMM+11] D. Golubovic, G. Miljkovic, S. Miucin, Z. Kaprocki, and V. Velisavljev. Webgl implementation in webkit based web browser on android platform. In *19th Telecommunications Forum (TELFOR).*, pages 1139–1142, November 2011.

[GW05] T. Gorschek and C. Wohlin. Requirements abstraction model. *Requirements Engineering*, 11(1):79–101, December 2005.

[HJBP98] D. Hamann, J. Järvinen, A. Birk, and D. Pfahl. A product-process dependency definition method. In *Proceedings. 24th Euromicro Conference.*, volume 2, pages 898–904 vol.2, 1998.

[HSP03] G. Herzwurm, S. Schockert, and W. Pietsch. Qfd for customer-focused requirements engineering. In *Proceedings. 11th IEEE International Requirements Engineering Conference.*, pages 330–338, 2003.

[HW11] B. Hujun and H. Wei. *Real-Time Graphics Rendering Engine*. Advanced Topics in Science and Technology in China. Zhejiang University Press, Hangzhou and Springer-Verlag Berlin Heidelberg, 2011.

[IGB12] S. Isenberg, M. Goebl, and U. Baumgarten. Is the web ready for in-car infotainment? a framework for browser performance tests suited for embedded vehicle hardware. In *14th IEEE International Symposium on Web Systems Evolution (WSE).*, pages 35–43, 2012.

[ISO01] International Organization for Standardization ISO. *ISO/IEC 9126. Software engineering – Product quality*. ISO/IEC, Geneva, Switzerland, 2001.

[ISO10] International Organization for Standardization ISO. *ISO/IEC 25010. Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models*. ISO/IEC, Geneva, Switzerland, 2010.

[ISO11] International Organization for Standardization ISO. Iso standards catalogue, http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue _detail.htm?csnumber=35733, 2011.

[Joh03] R. Johansson. Case study methodology. A key note speech at the International Conference âĂIJMethodologies in Housing ResearchâĂİ organised by the Royal Institute of Technology in cooperation with the International Association of PeopleâĂŞEnvironment Studies, Stockholm, 22 to 24 September 2003., September 2003.

[KBS+09] F. Konig, D. Boers, F. Slomka, U. Margull, M. Niemetz, and G. Wirrer. Application specific performance indicators for quantitative evaluation of the timing behavior for embedded real-time systems. In *Design, Automation Test in Europe Conference Exhibition (DATE '09).*, pages 519–523, 2009.

[KC07] B. Kitchenham and S. Charters. Guidelines for performing Systematic Literature Reviews in Software Engineering. Technical Report EBSE 2007-001, Keele University and Durham University Joint Report, 2007.

[KDR+07] L. Karlsson, G. Dahlstedt, B. Regnell, J. Natt och Dag, and A. Persson. Requirements engineering challenges in market-driven software development - an interview study with practitioners. *Information Software Technology*, 49(6):588–604, June 2007.

[KOR97] J. Karlsson, S. Olsson, and K. Ryan. Improved practical support for large-scale requirements prioritising. *Requirements Engineering*, 2(1):51–60, 1997.

[KS07] C. Kessler and J. Sweitzer. *Outside-in software development: a practical approach to building successful stakeholder-based products*. IBM Press, first edition, 2007.

[KvST97] R.J. Kusters, R. van Solingen, and J.J.M. Trienekens. User-perceptions of embedded software quality. *International Workshop on Software Technology and Engineering Practice*, 0:184, 1997.

[KvST99a] R. J. Kusters, R. van Solingen, and J. J. M. Trienekens. Identifying embedded software quality: two approaches. *Quality and Reliability Engineering International*, 15(6):485–492, 1999.

[KvST99b] R. J. Kusters, R. van Solingen, and J. J. M. Trienekens. Strategies for the identification and specification of embedded software quality. In *Proceedings. Software Technology and Engineering Practice (STEP '99).*, pages 33–39, 1999.

[Kwa96] A. G. Kwaku. Market orientation and innovation. *Journal of Business Research*, 35(2):93 – 103, 1996.

[KWR98] J. Karlsson, C. Wohlin, and B. Regnell. An evaluation of methods for prioritizing software requirements. *Information and Software Technology*, 39(14âĂŞ15):939–947, 1998.

[Lau02] S. Lauesen. *Software Requirements: Styles and Techniques*. Addison-Wesley, 2002.

[Mec12] AB Mecel. http://www.mecel.se/, December 2012.

[MF12] M. M. Mobeen and L. Feng. Ubiquitous medical volume rendering on mobile devices. In *International Conference on Information Society (i-Society).*, pages 93–98, June 2012.

[MH98] K. Matzler and H. H. Hinterhuber. How to make product development projects more successful by integrating kano's model of customer satisfaction into quality function deployment. *Technovation*, 18(1):25–38, 1998.

[Moo97] J. D. Mooney. Bringing portability to the software process, 1997.

[OH08] M. J. Ordonez and H. M. Haddad. The state of metrics in software industry. In *Proceedings. 5th International Conference on Information Technology: New Generations.*, ITNG '08, pages 453–458, Washington, DC, USA, 2008. IEEE Computer Society.

[Ols08] T. J. Olson. Hardware 3d graphics acceleration for mobile devices. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2008).*, pages 5344–5347, 2008.

[ORC+08] M. F. S. Oliveira, R. M. Redin, L. Carro, L. da Cunha, and F. R. Wagner. Software quality metrics and their impact on embedded software. In *5th International Workshop on Model-based Methodologies for Pervasive and Embedded Software (MOMPES 2008).*, pages 68–77, 2008.

[Pot95] C. Potts. Invented requirements and imagined customers: requirements engineering for off-the-shelf software. In *Proceedings. 2nd IEEE International Symposium on Requirements Engineering.*, pages 128–130, 1995.

[Pro13] The Web Standards Project. Acid tests, http://http://www.acidtests.org/, March 2013.

[RH09] P. Runeson and M. Höst. Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering*, 14(2):131–164, 2009.

[RHNoD+01] B. Regnell, M. Höst, J. Natt och Dag, P. Beremark, and T. Hjelm. An industrial case study on distributed prioritisation in market-driven requirements engineering for packaged software. *Requirements Engineering*, 6:51–62, 2001.

[RSBG84] G. Roman, M. J. Stucki, W. E. Ball, and W. D. Gillett. A total system design framework. *Computer*, 17(5):15–26, 1984.

[SGR09] R. B. Svensson, T. Gorschek, and B. Regnell. Quality requirements in practice: An interview study in requirements engineering for embedded systems. In Martin Glinz and Patrick Heymans, editors, *Requirements Engineering: Foundation for Software Quality*, volume 5512 of *Lecture Notes in Computer Science*, pages 218–232. Springer Berlin Heidelberg, 2009.

[SGR+12] R. B. Svensson, T. Gorschek, B. Regnell, R. Torkar, A. Shahrokni, and R. Feldt. Quality requirements in industrial practice: An extended interview study at eleven companies. *IEEE Transactions on Software Engineering*, 38(4):923–935, 2012.

[Sha13] Net Market Share. Market share statistics for internet technologies, http://www.netmarketshare.com/, June 2013.

[Sof13] Opera Software. http://my.opera.com/odin/blog/300-million-users-and-move-to-webkit, July 2013.

[SSK99] P. Sawyer, I. Sommerville, and G. Kotonya. Improving market-driven re processes. In *Proceedings. International Conference on Product Focused Software Process Improvement.*, pages 222–236, 1999.

[Sve11] R. B. Svensson. *Supporting Release Planning of Quality Requirements: the Quality Performance Model*. PhD thesis, Lund Institute of Technology, Lund University, Department of Computer Science, Sweden, October 2011.

[Tea13] WebKitGTK+ Team. http://webkitgtk.org/, July 2013.

[TSZ09] H. Tu, W. Sun, and Y. Zhang. The research on software metrics and software complexity metrics. In *International Forum on Computer Science-Technology and Applications (IFCSTA '09).*, volume 1, pages 131–136, 2009.

[vdLSR07] F. J. van der Linden, K. Schmid, and E. Rommes. *Software Product Lines in Action: The Best Industrial Practice in Product Line Engineering*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2007.

[Wie96] R. J. Wieringa. *Requirements engineering: frameworks for understanding*. John Wiley & Sons, Inc., New York, NY, USA, 1996.

[WLM05] C. Wohlin, L. Lundberg, and M. Mattsson. Special Issue: Trade-off Analysis of Software Quality Attributes. *Software Quality Journal*, 13(4):327–328, 2005.

[WW02] J. Webster and R. T. Watson. Analyzing the past to prepare for the future: writing a literature review. *MIS Quarterly*, 26(2):xiii–xxiii, June 2002.

[Yin09] R. K. Yin. *Case Study Research: Design and Methods*. Applied Social Research Methods. SAGE Publications, 2009.

[Yuy05] Y. Yuyu. Efficiency metrics model for component-based embedded application software. In *2nd International Conference on Embedded Software and Systems.*, page 7, 2005.