Software product line engineering for consumer electronics
Hartmann, Herman

IMPORTANT NOTE: You are advised to consult the publisher's version (publisher's PDF) if you wish to cite from it. Please check the document version below.

Document Version
Publisher's PDF, also known as Version of record

Publication date:
2015

Link to publication in University of Groningen/UMCG research database

Citation for published version (APA):

Copyright
Other than for strictly personal use, it is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license (like Creative Commons).

Take-down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Downloaded from the University of Groningen/UMCG research database (Pure): http://www.rug.nl/research/portal. For technical reasons the number of authors shown on this cover page is limited to 10 maximum.
Chapter 1. Introduction

1.1 Market Trends

Since the end of the previous century, consumer electronics products have changed radically. Traditionally, consumer electronics products were stand-alone products and were used for a few dedicated tasks, such as watching televisions, listening to music or were used to watch a video. Nowadays, consumer electronics products are connected devices and are used for a variety of tasks, such as communicating with friends using a mobile phone, watching online video clips or used as a navigation device for getting driving directions.

A large part of today’s functionality is implementation through software, whereas previously it was implemented through hardware. The consequence is that the software development effort now constitutes a significant, and in many cases largest, proportion of the total development effort. Therefore nowadays the success of a consumer electronics company is highly determined whether it can successfully develop the required software.

Since the end of the previous century, the following major trends exist for the development of software for consumer electronics products:

An increasing amount and complexity of software:
The size of software has grown rapidly and has roughly followed the law of Moore [Moore 1965], i.e. the size of software is doubled every 2 years [Rooijmans 1996, Genugten 2007]. These developments have been made possible because Integrated Circuits (ICs) are available at low costs and because the performance of these ICs have grown rapidly over the last decades. Together with the size also the complexity of the software has grown. This relates to the complexity of developing software in large teams, possibly spread over the world [Jacobs 2007], as well as the complexity of the software architectures [Siewiorek 2004].

An increasing amount of variability:
A wide range of variants are needed to serve the needs of different market segments, different geographical regions and different customer’s needs [Ommering 2004]. As an example look at the variety of digital cameras that are available in the market. Manufacturers offer a wide range of cameras with different lenses, size of screens, amount of pixels. A wide range of built-in software features are available in these digital cameras. for instance to support the photographer in taking different types of pictures, such as macro, portrait or pictures at candle light and to add special effects such as using “classic” sepia colors, or to create 3 dimensional pictures. Another example is that Digital Televisions have to support different interfaces such as different standards for connecting video, e.g. Scart, DVI, HDMI, S-video, USB and have to support different encoding standards such as MPEG, DivX, H.264.
In order to develop the required variability more efficiently and deliver products to market more rapidly, many companies have adopted the concepts of product line engineering [Pohl 2005]. This approach allows the variants of the products to be created efficiently through a systematic reuse of the development artifacts. Product line engineering has been successfully applied in the development of software for consumer electronics in many different firms [PHOF 2014].

**The use of third party components**

Until the early nineties, companies like Sony, Philips and General Electric developed and manufactured the entire product, but nowadays large parts of a product is developed by specialized firms [Davenport 2005]. This trend is a result of the requirements described above: The first reason is that, due the huge amount of software, it is not possible for a single firm to employ sufficient software developers anymore [Raskin 2005]. Secondly for a single firm it is too difficult to amortize the investments, while a specialized firm could develop this software and sell this to a large number of firms, thereby creating a larger economy of scale [Porter 1985]. The third reason is that many products are now a combination of functionality that originates from other types of products. For instance mobile phones now have built in cameras and MP3 players, digital cameras are now equipped with a GPS. For a firm that wants to add a new feature to a product, it would take too much through-put-time to develop this software in house.

The use of 3rd party software components has been facilitated by the use of software components. Since software components can be developed independently, the development can be subcontracted to third parties or can be purchased as existing, commercial of the shelf components [Wallnau 2002, Greenfield 2004].

**An increasing speed of innovation and pressure on time-to-market:**

New functionality is introduced rapidly and older products become commercially obsolete within a few years [Minderhoud 2005]. For instance playing Vinyl records was the most used way to listen to music between 1930 and 1990. Vinyl records were largely replaced by CDs between 1990 and 2005 and CDs have become largely replaced by storage on flash memory on Mp3 players. In the last few years music is streamed from the internet, e.g. using Spotify, and no copies are stored locally anymore. Furthermore, functionality of products changes rapidly. This innovation is causing a high pressure on the time-to-market because usually the first company that offers the functionality, earns most of the revenues, while firms that are lagging lose market share rapidly. Furthermore the time to amortize the development investments has reduced because a feature that is considered a unique selling point at one moment in time may become mainstream functionality within in a few months [Kano 1984].

**A high pressure on product quality:**

Delivering products with sufficient quality is becoming an increasing problem and consumers are increasingly unsatisfied with the reliability of the products. The field call rate, which is the percentage of products that are returned to the shops and service centers, has dramatically increased [Brombacher 2004]. This is caused by two types of failures:
Introduction

(1) Failures caused by faults in the software [Sieworek 2004], referred to as software failures [Lyu 1996, BSI 1998]. This is often caused by a mismatch between components from 3rd parties and in-house developed components [Trew 2006].

(2) Failures caused by a discrepancy between the user’s expectations and the functionality of a product, referred to as soft failures [Geudens 2005, Koca 2006]. This means that although the product functions as specified, the user is not satisfied with the functionality [Korhonen 2010], experiences usability problems or experiences problems that are caused by a poor interoperability with other devices [Norman 2002, Ouden 2006].

1.2 Problem Statement

The following problem statement is the starting point for this thesis:

*Consumer electronics products are characterized by a high degree of innovation and are containing an increasing amount of variability that is implemented through software. The consequences are that the development cost, throughput time and product quality are highly determined by software development. To reduce development costs and bring new functionality to the market faster, the recent trend is that the software is developed by multiple parties. As a result the success of a company nowadays largely depends on whether it can develop the required software together with multiple parties in a timely fashion, while ensuring sufficient reliable products.*

1.3 Research Objectives

Based on the problem statement in the previous section the following objectives for this thesis have been formulated:

- Determine the consequences and identify the problems that are caused by the increasing amount of software, variability and involvement of multiple parties for the practice of software product line engineering in the consumer electronics industry.

- Developed methods that can be used by the industry as part of their strategy, software development and test process to address the consequences and problems that have been identified, and determine the benefits and consequences of applying these methods for the software development process.

Note: While the objectives for the research in this thesis find its origin in consumer electronics, some of the results are more widely applicable, i.e. to the domain of embedded systems or software engineering in general. Section 1.7 gives an overview of the applicability of the individual research results.


Chapter 1

1.4 Research Methodology

This section describes the research methodology that is used in this thesis. Determining an appropriate research methodology is considered an important element in a research study [Wedawatta 2011]. A research methodology is defined as “the overall approach to the entire process of the research study” [Collis 2009]. This includes the process that is followed and the methods that are used to answer the research questions [Saunders 2009].

This section describes the general process and gives an overview of the methods that are used. Section 1.8 presents in detail which type of research results, process and methods have been used to obtain the individual research questions.

This section starts with the type of research results, since this determines which process and what type of methods have to be used [Runeson 2009, Saunders 2009].

1.4.1 Types of research results

Robson [Robson 2002] identifies four types of research results: (1) Exploratory, (2) Descriptive, (3) Explanatory, (4) Improving. In the literature of software engineering usually two different types of research results are identified [Lazaro 2005, Easterbrook 2008, Shaw 2002]:

A. Research that studies the practice of software engineering, including the methods and techniques used. This type is empirical research and results in a descriptive or explanatory model, the identification of concerns, reporting of experiences, answer to a specific question, or taxonomy for a problem area. This type encompasses the first three types of research results as identified by Robson.

B. Research that is aimed to solve a particular problem. The goal of this type of research is to find better ways in doing things. This type of research results in a solution, which can be a method or technique; a language or tool; or a specific solution that embodies a result. These solutions are aimed to improve the development of software e.g. to make the process more efficient, develop higher quality products, create architectures that are better maintainable etcetera. This type encompasses the last category of Robson, i.e.: improving.

Long term research programs in software engineering usually involves a mix of these two types of research results since researchers investigate a specific problem and develop a solution that works best [Wieringa 2006].

The research for this thesis also consists of these two types of research, since the objectives are both to determine the consequences, as well as to create methods to improve the practice.

1.4.2 Research process

A scientific research process is defined as follows [Bailey 2012]: “Research is a systematic process based on the scientific method that facilitates the identification of relationships and determination of differences in order to answer a question. The scientific method is a process that uses an organized structure to formulate questions and determine answers in a research project”.

4
Introduction

The research in this thesis follows a stepwise process and these steps are presented in Figure 1. Here the processes to obtain the research results that covers both the objectives are presented as connected tracks. On the left side of the figure the steps to obtain descriptive and explanatory results are shown and on the right hand side the steps to obtain the research results that are aimed to solve a particular problem.

![Figure 1 Research process](image-url)

The research methods

Software engineering science is a laboratory science which involves case studies to demonstrate the applicability of a certain method and to develop and prove or disprove theories [Basili 1996]. Easterbrook identified five classes of research methods that are most relevant in software engineering [Easterbrook 2008]: (1) controlled experiments, (2)
Chapter 1

exploratory case studies, (3) confirmatory case studies, (4) survey research, (5) ethnographies and (6) action research.

In this thesis, the first 3 of these methods are used and are summarized below:

(1) Controlled Experiments: A controlled experiment is used to test a hypothesis. It is an investigation of a testable hypothesis where one or more independent variables are manipulated to measure their effect on one or more dependent variables [Easterbrook 2008]. A precondition for using controlled experiments is a clear hypothesis, and the outcome of the experiment is whether the hypothesis is rejected or accepted.

(2) Exploratory case studies: Exploratory case studies are used to explore a problem area, get an overall picture, describes current practice and to build theories [Easterbrook 2008, Eisenhardt 1989]. During theory building, a theory is created by identifying similarities and differences between different cases and therefore the resulting theory is derived from the studies [Robson 2002]. In general, case studies are preferred when the investigator has little control over the events and when the focus is on a contemporary phenomenon with some real-life context [Yin 2003]. A theory is more sound when multiple and different case studies and data collection methods are used [Eisenhardt 1989].

(3) Confirmatory case studies: Confirmatory case studies are used to test or validate an existing theory. The purpose of theory validation is to test whether a proposed theory is sound and to assess whether a method or technique actually works in practice [Easterbrook 2008]. Furthermore it is used to determine the benefits of a method. Shaw described the different validation techniques in software engineering [Shaw 2002]. She argued that a good research result requires evidence that the result is sound. In this classification, confirmatory case studies can have the form of a guided experiment, a pilot project, a toy or real life example, or by capturing the experiences of the involved persons that are using the newly developed method.

In this thesis, these various types of case studies are used and different data collection methods. In most cases the theory building and validation is based on qualitative data since quantitative data was usually not available. Where possible the qualitative evaluation was supported by quantitative data.

1.5 Research Questions

The research questions in this thesis are based on a stepwise refinement of the research objectives and follows the process described in Figure 1. Since not all the identified problems could be addressed; the decision on which research questions have been answered is based on the importance of the research questions for the involved organizations and on the availability of case studies [Robson 2002]. These selection criteria have the advantage that the research is based on existing problems in the industry and that the proposed methods have been validated using concrete case studies.

Part I is the starting point of this thesis. This part is focused on an exploration of the problem space and identification of the architectural challenges for the development of software for consumer electronics. The overall research question for this part is:
Introduction

RQ 1  What transitions have taken place in the development of software for consumer electronics products and what are the consequences for building software product lines?

In Chapter 2 the transition in the development of software development for consumer electronics products between 1990 and 2010 is analyzed and the forces that govern these transitions are identified as well as the consequences for software architectures. The research questions that will be answered in Chapter 2 are:

RQ 1.1  What transitions between industry structures have taken place? What were the forces and actors involved?
RQ 1.2  What were the forces and actors involved?
RQ 1.3  How were the architectural interfaces between the actors defined, and what were the consequences for modularity, variability and product integration?

One of the identified steps for further research in Chapter 2 is that an industry can adopt a hierarchical model where at different places in the architecture, different industry structures are used, thus leading to a more fine-grained model. Since 2010, several consumer electronics firms have create ecosystems by offering a stable interface towards 3rd party application developers and an application store to support this, but do not have a modular architecture of the software that is embedded in the device. Consequently a model is adopted where an ecosystem is used to support 3rd party applications, but different industry structures are used for the device.

Choosing an ecosystem type is a critical strategic question. When a company decides to use its own platform, it has to open up its architecture and has to attract sufficient parties. When a company decides to use an existing ecosystem by integrating an existing middleware layer that serves as application framework, it has to be confident that the adopted ecosystem is sufficiently viable. Therefore the research questions that will be answered in Chapter 3 are:

RQ 1.4  What types of ecosystems are used in consumer electronics products to support 3rd party applications?
RQ 1.5  Which of the ecosystems types is most suitable for a specific product category, from a software engineering perspective?

One of the outcomes of the research presented in chapter 2 is that for software that is embedded in consumer electronics products, software supply chain is the dominant form of co-operation between companies. In a software supply chain a company buys a set of components from several suppliers, with their variability, and integrates them into a product, possibly combined with in-house developed components. For each of its customers, i.e. the next participants in the chain, a specialized version is created. Each of the participants adds new functionality and makes configuration choices based on their role and responsibility in the supply chain.
This means that the product line now consists of different components with different sets of variability and possibly using different technologies to implement the components. Therefore the overall research question for Part II of this thesis is:

RQ 2 What are the consequences for variability management, when applied in software supply chains, and what modelling approaches can support this?

Since suppliers of software develop a product line for multiple customers and multiple products, such as mobile phones, TVs and DVDs; the variability of these product lines have to be managed across different, or multiple, product lines. From this we derive the following research question which will be answered in chapter 4:

RQ 2.1 How can feature models be constructed to model multiple product lines in software supply chains?

It frequently happens that software is purchased from alternative suppliers for similar functionality, for instance because these suppliers have components with a different feature set; a different price setting, quality etcetera. This gives the following research question which will be answered in chapter 5.

RQ 2.2 How should feature models be constructed which capture the variability from the components from alternative suppliers, partly offering the same feature set?

Since software components from alternative suppliers often have different interfaces and are developed using different component technologies, also known as heterogeneous components, glue components are usually needed to bridge these components. In a product line where multiple suppliers are used, this may become a repetitive task for which an automated approach, such as Model Driven Architectures (MDA), could be suitable. This gives the following research question, which will be answered in chapter 6:

RQ 2.3 What are the challenges that arise from combining components with non-matching interfaces and what are the limitations of current practice?

RQ 2.4 How can MDA be used to bridge mismatches between heterogeneous components in software supply chains?

RQ 2.5 What are the consequences for the development process and the development roles?

One of the outcomes of the research in chapters 4 and 5 is that variability models that are used in software supply chains have become large in size and complex, especially due to the large amount of dependency relations. Therefore alternative languages should be considered to capture variability. Such an alternative language is TVL, a textual language which abbreviation stands for Textual Variability Language. This brings us the following research question that will be answered in chapter 7.
Introduction

RQ 2.6  What are the benefits of TVL for modeling product line variability, as perceived by model designers?

RQ 2.7  What are the product line variability modeling requirements that are not fulfilled by TVL?

One of the outcomes of part I of this thesis is that the amount of software faults is increasing which is caused, amongst other reasons, by an increasing amount and complexity of software and because of the use of components from 3rd parties, as shown in part II. Because of the pressure on time-to-market it is not possible anymore to remove all software faults prior to introduction.

One of the methods to increase the test efficiency is to improve the effectiveness of the test process by focusing on those areas that have the largest impact on the reliability, as perceived by the end-user, and by focusing on the risks to the user and business. Therefore the overall research question for part III of this thesis is:

RQ 3  How can test efficiency be improved when the risks and user product interaction are taken into account?

Operational profile driven testing and risk based testing are widespread, and related, methods to improve the test efficiency. In operational profile driven testing the number of test cases is allocated to the different functions and operations according to the quantitative use in practice. Risk based testing is used to prioritize test effort to those functions that pose the greatest risk, which includes several aspects. However the benefits of these methods have not been established with statistical evidence. This gives the research questions that will be answered in chapter 8.

RQ 3.1  What are the benefits for the test efficiency when using operational profiles?

RQ 3.2  Which dimensions in risk based testing require most attention to reduce the overall risks?

One of the outcomes of chapter 8 is that risk based testing improves the test efficiency. However, this method does not take into account that different members of the product line contain different feature sets. Furthermore, in product line engineering the development and testing of software product lines is usually separated into testing during domain engineering and during application engineering. And, as shown in part II, product line engineering relies heavily on variability management tools. This gives the research questions which will be answered in chapter 9.

RQ 3.3  How can risk based testing be applied in product line engineering?

RQ 3.4  What are the consequences for testing during domain and application engineering?

RQ 3.5  What tool support is required?
Chapter 1

The use of operational profiles is a widespread method for software products and the benefits have been determined in chapter 8. However, operational profiles are used for reducing the technical failures caused by faults in the software. For highly innovative consumer electronics products, failures often occur because an unexpected product use and because of a mismatch between the functionality of a product and the user’s expectations. This leads often to non-technical failures, also known as “soft” failures. Chapter 10 determines the suitability of operational profiles for highly innovative consumer electronics products and therefore the research questions that will be answered in this chapter are:

RQ 3.6 To what extent can an operational profile be applied to consumer electronics products to model (unexpected) product use for highly innovative consumer electronics products?

RQ 3.7 What adjustments need to be made?

1.6 Thesis Outline and Article Overview

The main body of this thesis consists of a number of articles that have been peer reviewed and have been published in major journals (chapters 2, 6, 10) and conferences (chapters 3, 4, 5, 7, 8, 9). The articles have been mainly left as is. The exceptions are chapter 3, 8 and 9 for which an extended version is included.

In this thesis, the articles have been published in a logical order, rather than the order in which they have been published. The reasons for this are as follows:

- Part I contains descriptive and explanatory research and describes the context and challenges, while the part II and III contain solutions for a selection of the challenges that are identified in the part I.
- It is based on the order of activities in a business context as described in the BAPO (Business Architecture Process Organization) model for software product line engineering [Linden 2004]. Part I describes the choices of a firm, e.g. the role it wants to play in an ecosystem or supply chain. This is followed part II that covers the creation of a product, while part III contains solutions for the last phase in a development project, namely that of testing.

The main relations between the articles is shown in the figure below, and follows the same structure as the stepwise refinement of the research questions:
1.6.1 Part I: Industry structures for software development

The article presented in Chapter 2 provides an analysis of the transition of software development for the consumer electronics industry between 1990 and 2010 using case studies from digital televisions and mobile phones. This article introduces a model consisting of five industry types, describes the forces that govern the transitions and describes the consequences for software architectures and software product line engineering. It is shown that the software supply chain is the dominant structures because the modularization of the architecture is limited. The model and forces presented serve the decision making process for companies that consider a transition to a different industry type.

The article presented in Chapter 3 describes three types of ecosystems that are used for consumer electronics products that support 3rd party applications: Vertically integrated hardware/software platforms, closed source software platforms and open source software platforms. A first step towards multi-criteria decision support method is introduced to select the most suitable ecosystem type for a product category from a software engineering perspective. This method is used to analyze traditional consumer electronics products, such as gaming consoles, digital cameras and digital televisions, as well as smart phones, tablets and personal computers, and newly introduced product categories, such as smart watches. Using the analysis method, the main challenges are identified and the article shows how the different players address these challenges.
Chapter 1


1.6.2 Part II: Variability management in software supply chains

The article presented in Chapter 4 introduces the concept of context variability, which contains the primary drivers for feature selection, such as geographic regions or different customer types. The context variability model constrains the feature model which makes it possible to model multiple product lines. The article shows how this approach can be used in software supply chains and that the approach is intuitive and straightforward to use with existing feature modeling tools.

The article presented in Chapter 5 introduces an approach to combine feature models from alternative suppliers, i.e. suppliers that offer partly the same functionality. The article shows that the introduced method facilitates an efficient selection of features and suppliers during application engineering, while preserving the links between the features and the development artifacts.

The article presented in Chapter 6 describes the challenges that arise from combining components using different component technologies and with non-matching interfaces and in particular the challenges that arise from software supply chains and for developing resource constrained devices. An approach is introduced that uses Model Driven Architectures (MDA) to generate the required glue components efficiently and only then when they are required. The article addresses the implications on the development process and identifies the development roles that are associated with this approach.

The article presented in Chapter 7 describes an evaluation of a textual feature modeling language using four industrial case studies. These case studies represent examples of large feature models which are cumbersome to model with the traditional graphical feature modeling languages, e.g. based on the FODA notation. The article shows that TVL, the textual feature modeling language that was evaluated, combines advantages of textual languages as well as graphical languages and that practitioners can benefit from TVL.


1.6.3 Part III: Software testing

The article presented in chapter 8 gives a statistical analysis of the benefits of test time allocation based on an operational profile and test time allocation based on a the risks for the user and the business. An operational profile is a quantification of the operations that in its turn is derived from a quantification of functional profiles and user profiles. Risk based testing is used to prioritize test effort to those functions that pose the greatest risk. The analysis shows that using operational profiles improves the test efficiency, however not when a high reliability is required. For risk based testing a quantification is introduced, and the analysis shows that an efficiency improvement can be realized when the usage frequency is treated as a separate dimension of the risk matrix.

The article presented in chapter 9 presents risk based testing for product line engineering. This method is based on risk based testing for single system engineering which is extended with a dimension that captures the percentage of product variants that use a particular development artefact. Based on the risk of development artifacts, the priorities for testing during domain and application engineering are determined. It is shown that the basic ideas behind risk-based testing product lines are intuitive, pragmatic in nature, and provide the means for practitioners for guiding the test effort for different test levels.

The article presented in chapter 10 discusses the limitations of using operational profiles for highly innovative consumer electronics products since operational profiles are focused on reducing the technical risks while an increasing proportion of product returns is caused by non-technical failures, so called soft-failures. The article presents an enhanced user-product interaction framework that takes into account the use-process, the environment and the different types of users.

Chapter 8 H. Hartmann: “A Statistical Analysis of Operational Profile Driven Testing and Risk Based Testing”. This article is submitted and is an extended version of:
H. Hartmann, J. Bokkerink, V. Ronteltap: ”How to reduce your test process with 30%; The application of operational Profiles at Philips Medical Systems”. In: Supplementary proceedings of the 17th IEEE International Symposium on Software Reliability Engineering.
Chapter 9 H. Hartmann, F. van der Linden, J. Bosch: Risk based testing for software product line engineering. SPLC 2014: 227-231, DOI 10.1145/2648511.2648535

This thesis concludes with Chapter 11. This chapter answers the research questions, gives recommendations for future research and describes the contributions of this thesis.
1.7 Applicability of the Research Results

While this thesis has found its starting point in consumer electronics, many of the results of these articles are applicable to a wider domain. Here we identify three levels (1) Consumer electronics (2) Embedded systems and (3) Generally applicable in software engineering. The table below presents this classification and includes the domain in which the case studies where executed.

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Domain</th>
<th>Case studies</th>
</tr>
</thead>
<tbody>
<tr>
<td>2: The changing industry structure of software development for consumer electronics</td>
<td>Consumer Electronics</td>
<td>Consumer Electronics</td>
</tr>
<tr>
<td>3: Consumer electronics software ecosystems</td>
<td>Consumer Electronics</td>
<td>Consumer Electronics</td>
</tr>
<tr>
<td>4: Using context variability to model multiple product lines</td>
<td>Embedded Systems</td>
<td>Consumer Electronics</td>
</tr>
<tr>
<td>5: Supplier independent feature modelling</td>
<td>Embedded Systems</td>
<td>Consumer Electronics</td>
</tr>
<tr>
<td>6: Using MDA for integration of heterogeneous components in software supply chains</td>
<td>Embedded Systems</td>
<td>Consumer Electronics</td>
</tr>
<tr>
<td>7: Evaluating a textual feature modelling language</td>
<td>Generally applicable</td>
<td>4 different domains</td>
</tr>
<tr>
<td>8: A statistical analysis of operational profile driven testing and risk based testing</td>
<td>Generally applicable</td>
<td>Healthcare</td>
</tr>
<tr>
<td>9: Risk based testing for software product line engineering</td>
<td>Generally applicable</td>
<td>Healthcare</td>
</tr>
<tr>
<td>10: Towards a more systematic analysis of uncertain user-product interactions</td>
<td>Consumer Electronics</td>
<td>Consumer Electronics</td>
</tr>
</tbody>
</table>

1.8 Detailed Overview of Research Processes and Methods

In Section 1.4 the overall research methodology was described. The current section provides details on how this was applied for each of the research questions that have been answered in the individual chapters. This section covers the type of research results, the process that was applied, in relation to the process shown in figure 1, and the research methods and
validation techniques. Here the types of research results, methods, and the validation techniques from Shaw [Shaw 2002] are highlighted using an italic font.

**Chapter 2: The changing industry structure of software development for consumer electronics**

**Type of research result:** A descriptive model that presents a taxonomy of different types of industry structures. Furthermore it contains an explanatory model that presents the forces that govern the transitions between these types.

**Research process:** To arrive at the research results, steps 1 through 4 have been executed twice: At first the current practice from digital televisions was observed and described. This was compared with related art, i.e. the PC industry and others, and a theory was formulated. Then the practice of mobile phones, was observed and described. Based on these observations the model of industry structures was refined. Finally the problems and consequences, i.e. step 5, were identified and described.

**Research methods:** Exploratory case studies have been used to build the theory, using case studies from two different product types. Multiple data sources are used, i.e. the experiences from the authors, data obtained through interviews, and information from literature. The empirical predictive model is evaluated using these case studies and information from literature.

**Chapter 3: Consumer electronics software ecosystems**

**Type of research result:** A descriptive model that presents the different types of ecosystems. Furthermore it contains a first towards a multi-criteria decision support method which is a method to select between the alternative types of ecosystems. This method is based on an explanatory model of the strengths and challenges of each ecosystem type.

**Research process:** To arrive at the research results, step 1 through 4 have been executed a number of times. To obtain the descriptive and explanatory model, multiple cases from mobile devices were studied, which iteratively lead to these theories. This result was published as separate article. Then the practice of other consumer electronics products was studied and described, which confirmed the theory. The creation of the decision support method was started with the observations from the challenges, i.e. step 5, and the comparison with literature, i.e. step 6, did not reveal an existing method that can determine the most suitable type of ecosystem. The procedure to select between different types of ecosystem was then developed iteratively which started with an initial method, i.e. step 7, and was exercised and evaluated, i.e. steps 8 through 10, using the case studies from 7 different consumer electronics products.

**Research methods:** Exploratory case studies have been used to build the theory. For validating the theory and the model, the case studies where used to confirm and refine the theories. The case studies are based on qualitative data from real life examples of a wide range of consumer electronics products, where literature and the experiences from the authors are the main data sources. The method to determine the most suitable ecosystem type was evaluated using real life examples from a wide range of products, using data from literature.
Chapter 4: Using context variability to model multiple product lines

Type of research result: This research contains a technique to improve the practice of variability modelling in software supply chains.

Research process: This research started with step 5, a result of the research in chapter 2, that variability is managed in a supply chain. Furthermore, an existing example from NXP showed that a method was needed to model multiple product lines in a single feature model. The comparison with related art, i.e. step 6, revealed that a method did not exist that could be used within a supply chain, nor with standard feature modelling tools. The improved method was developed, i.e. step 8, and validated with cases from literature and the earlier mentioned example from NXP. This showed clear improvements, i.e. step 9, but also some shortcomings of the used tools, related to merging of feature models. Since merging was not regarded as primary objective, the method was not further improved.

Research method: Exploratory case studies from the practice of NXP and in literature were used to identify the problem. An example is used to demonstrate the applicability of this technique, and experiences from several persons were used, using pilot projects, to validate whether this technique is easy to apply and can be used with existing variability management tools.

Chapter 5: Supplier independent feature modelling

Type of research result: This research contains a technique to improve the practice of variability modelling in software supply chains.

Research process: This research started with the results from chapter 4, i.e. step 5, which identified the problem of merging overlapping feature models. This problem was confirmed by a concrete case study, i.e. that of ZigBee. The literature was studied, i.e. step 6, but revealed no existing solutions. To arrive at the improved solution, multiple alternatives were developed, compared and exercised, i.e. step 7 through 10.

Research method: Case studies from the practice of NXP were used to identify the problem. A confirmatory case study consisted of using a pilot project that demonstrates the applicability of this technique. Experiences from several persons were used to validate whether this approach can be used with existing variability management tools.

Chapter 6: Using MDA for integration of heterogeneous components in software supply chains

Type of research result: This research contains a technique to improve the practice of creating glue components.

Research process: This research started with the results from chapter 4, i.e. step 5, which identified the problem of combining heterogeneous components. This problem was confirmed by a concrete case study, i.e. that of ZigBee. The literature was studied, which revealed no existing efficient solutions. To arrive at the improved solution, multiple alternatives where considered, developed and exercised, and the improvements determined i.e. step 7 through 10. The final solution still revealed some shortcomings, but no additional improvement cycles where executed due to a lack of funding.

Research method: Case studies from the practice of NXP were used to identify the problem and a confirmatory case study consisted of using a pilot project that demonstrates the
applicability of this technique. *Experiences* from several persons were used to validate whether this approach has efficiency gains in comparison with current practice.

**Chapter 7: Evaluating a textual feature modelling language**

**Type of research result:** This research provides an *answer* to the question what the benefits of TVL are and provides an *answer* to the requirements that are not fulfilled.

**Research process:** This evaluation fulfils step 10: *determine improvements*, and step 7: *analyze shortcomings*, to validate the improved method of using a textual variability language.

**Research method:** *Confirmatory case studies* were used to validate the research results. The data was obtained using a questionnaires and through interviews. In this way the *experiences* from several persons were used based on *real life examples* using TVL and these experiences were compared with using the tools that were used by the participants as part of their day-to-day work.

**Chapter 8: A statistical analysis of operational profile driven testing and risk based testing**

**Type of research result:** This research provides *answers* to the questions what the efficiency improvements of using operational profiles are and comparison of which dimensions in risk based testing require most attention. Furthermore, it contains an improved *method* for risk based testing.

**Research process:** This evaluation fulfils step 10: *determine improvements* of using operational profiles and risk based testing. Based on the analysis the shortcomings were analyzed, step 7, and improved methods were proposed, step 8, namely to use operational profiles unless a very high reliability is required and for risk based testing to use three dimensions, rather than two. The improvements were determined using a statistical analysis.

**Research method:** A *quantitative analysis* was done using a number of examples and a *confirmatory case study* which consisted of using a *pilot project*.

**Chapter 9: Risk based testing for software product line engineering**

**Type of research result:** This research contains a *method* to improve the practice of testing in software product line engineering.

**Research process:** This research followed the steps 5 through 10: First the problems from current practice were identified that revealed that in current practice an improvement can be realized. The comparison with related art revealed no existing methods. Based on the shortcomings of current practice, an improved method was developed and the improvements were determined. Also the shortcomings of the improved method were analyzed, related to quantified feature modelling. A further improvement cycle has not been started but is left for further research.

**Research method:** The method was developed using an *exploratory case study* from a *real life project* and interviews where used to obtain the *experiences* from the practitioners. An *example* was used to validate the use and modelling of quantified feature models.
Chapter 10: Towards a more systematic analysis of uncertain user-product interactions

**Type of research result:** This research contains a *qualitative model* and *framework* that describes which aspects are needed to analyze unexpected user product interaction.

**Research process:** This research followed the steps 5 through 10: First the problems from current practice were identified that revealed that operational profiles cannot be used for detecting non-technical failures. The comparison with related art revealed that there are no existing models to analyze user product interaction and do not provide a framework that can be used for highly innovative consumer electronics. Based on the shortcomings of operational profiles, an improved framework was developed and the improvements were determined. The analysis revealed that the framework could not be fully validated, i.e. related to the environment. A further validation and improvement cycle has not been started but is left for further research.

**Research method:** *Exploratory case studies* were used to develop the framework, and a *controlled experiment* was executed to test the hypotheses using a quantitative analysis.

### 1.9 Related Publications

A few related publications are not included in this thesis. Earlier versions for chapter 6 have been published as a workshop and conference articles.

On the relation between software reliability and soft reliability, a publication was made that describes different classifications that were found in the academia and the industry. This has led to a better understanding of the differences between Fault-not-Found and No-Fault-Found and has provided supportive material for modeling different types of users and user product interaction as described in chapter 9.

A publication was made on the application of TRIZ for software development. This publication discusses the increasing size and complexity of software architectures and argues that TRIZ might become useful for dealing with conflicting requirements in developing software architectures. These related publications are the following:

1.10 Contributions to the Articles

This thesis consists of research that is published in articles for which multiple authors have contributed. This section describes my personal contributions to the research in these articles.


**Chapter 3:** H. Hartmann, J. Bosch: “Orchestrate Your Platform: Architectural Challenges for Different Types of Ecosystems for Mobile Devices”, ICSOB 2014: 163-178 I did the research for this paper and wrote the paper. Jan Bosch acted as reviewer. This also holds for the extension that has been submitted and is included in this thesis.

**Chapter 4:** H. Hartmann, T. Trew: “Using Feature Diagrams with Context Variability to Model Multiple Product Lines for Software Supply Chains”. SPLC 2008: 12-21 I did the research for this paper and wrote the paper. Tim Trew acted as reviewer.

**Chapter 5:** H. Hartmann, T. Trew, Aart Matsinger: “Supplier independent feature modelling”. SPLC 2009: 191-200 I did the research for this paper and wrote the paper. Aart Matsinger analyzed the variation points of the case study. Both Aart Matsinger and Tim Trew acted as reviewer.

**Chapter 6:** H. Hartmann, M. Keren, A. Matsinger, J. Rubin, T. Trew, T. Yatzkar-Haham: “Using MDA for integration of heterogeneous components in software supply chains. Sci. Comput. Program. 78(12): 2313-2330 (2013)” The research for this paper was a joint effort between the six authors over a long period of time. My primary contributions to the research were the analysis of the different approaches in current practice, the identification of the requirements from software supply chains, the overall process, the connection to variability management tooling, the analysis of the tasks of the different development roles and the evaluation of the approach. Furthermore I acted as corresponding author of this paper, including earlier versions for workshops and conferences, I wrote most parts of this paper and acted as editor for those parts that I didn’t write.

**Chapter 7:** A. Hubaux, Q. Boucher, H. Hartmann, R. Michel, P. Heymans:” Evaluating a Textual Feature Modelling Language: Four Industrial Case Studies”, SLE 2010, 3rd International Conference on Software Language Engineering, (2010). The research for this paper was initiated by the University of Namur and I contributed as expert from the industry on complex feature modelling and my experience as researcher on user product interaction. I contributed by the identification of suitable cases studies and I
analyzed and described these two case studies. Furthermore I identified the feedback process and contributed by analyzing the differences between the case studies. I acted as reviewer for the parts that I didn’t write.


I did the research for this paper and wrote the paper. Jarmila Bökkerink was the industry partner at which the case study was performed. Vincent Ronteltap acted as reviewer. The extensions that has been submitted and is included in this thesis was done solely by myself.


I did the research for this paper and wrote the paper. Frank van der Linden identified the case study. Frank van der Linden and Jan Bosch acted as reviewer.


This paper was the result of joint research over a long period of time. The starting point for this article was my research on operational profiles, described in chapter 8. I identified that operational profiles has less value benefits when applied to highly innovative products, by recognizing that (1) the use process of consumer electronics products is not captured in operational profiles; (2) the user profiles, as defined by software reliability engineering are different that those used for highly innovative products and (3) that the environment is not explicitly modeled. Based on this analysis I contributed to the developing of the proposed interaction framework. I contributed with setting up the analysis method, and executed two of the high contrast consumer tests and analyzed the results of these tests.
Part I: Industry Structures for Software Development

“For when I construct lines and colour combinations on a flat surface, it is with the aim of portraying ‘universally beauty’ as consciously as possible. Nature inspires me, but I want to approach the truth as closely as possible, abstract everything until I come to the foundation of things.”

Piet Mondriaan