Master Thesis

New Software Required?

The Product Software Overhaul Framework - A Multiple Case Study Of Software Company Experiences

Author:
Martin Raub
Master student of Business Informatics
Student number: 3937836
E-Mail: m.raub@students.uu.nl

Supervisor:
Dr. Slinger Jansen
(Utrecht University)

2nd Supervisor:
Prof. Dr. Sjaak Brinkkemper
(Utrecht University)

Universiteit Utrecht

Faculty of Science
Department of Information and Computing Sciences
Princetonplein 5, De Uithof
3584 CC Utrecht

August 2014

Declaration of Authorship

I, Martin Raub, declare that this thesis titled, 'New Software Required?The Product Software Overhaul Framework - A Multiple Case Study Of Software Company Experiences' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a Master degree at Utrecht University .
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.

Signed: _____

Date: _____

Abstract

The software industry is booming. Valued at 299 billion USD in 2013 and having a growth rate of about three up to four percent per year, it is one of the most rapidly growing sectors in ICT. Within the flourishing software industry, one type of software became increasingly popular during the past decade, namely product software. Product software is highly standardized software that is developed for and traded in a specific market. Product software companies nowadays face the challenge of a constantly evolving industry, where technology and platforms are emerging rapidly. In order to stay competitive in the market and to respond to the technological changes, product software companies are continuously concerned with the improvement and renewal of their products.

A major part of such a renewal is the software product overhaul process. Companies are facing a complete overhaul of their product every once in a while. In this context, they are either concerned with a fundamental change of their product or the migration of their product towards a new technological environment. Such product overhaul processes are complex, challenging and time-consuming, however research that embeds this phenomenon within practical contexts is scarce. Thus, the aim of this thesis is to investigate the software product overhaul process from an industrial point of view in order to define its major characteristics.

A systematic literature review is conducted first in order to provide an overview of state-of-the-art literature. A multiple case study approach is then chosen to explore the phenomenon in its natural settings. Eight product software companies particpate in the multiple case study, whereas data is collected by means of semi-structured interviews on the one hand, an analysis of documents on the other hand.

The multiple case study reveals that there are different types of an overhaul such as the migration towards a new technological environment or the redevelopment of parts of the product such as the user interface. The overhaul process is mostly triggered by technological, organizational and market-related aspects. Furthermore, companies can approach this process in different manners. The right choice of such a strategy is essential, as the project success often depends on the strategy that was chosen. The software product overhaul process generally appears to be highly complex, related to large investments and tremendous delay. There are manifold challenges to overcome for software companies before, during and after the product overhaul. To support this process in software companies, the present thesis contributes several models. A *process deliverable diagram* introduces the most important phases and steps that an overhaul consists of. Several *challenges-countermeasures models* help to approach the major challenges of such a process. Finally, the *product software overhaul framework* provides an elaborate overview of what a software product overhaul is affected by.

Keywords: *Software business, Product software, Software product management, Software product overhaul, Software migration, Multiple case study, Semi-structured interview.*

Acknowledgements

First of all, I would like to thank my supervisor dr. Slinger Jansen for his support and guidance througout the thesis project. Besides him, I would also like to thank prof. dr. Sjaak Brinkkemper for being the second examiner of this thesis.

Furthermore, I would like to thank all of the case companies, who shared their valuable knowledge and insights to make this thesis happen.

I would also like to thank my friends Lars, Guillermo, Nikos, Aris, Alberto, Brenna and many more.

Finally, I would like to thank my family and especially Laura for everything they have done for me in the past.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The ICT industry is flourishing. During the last decade, global ICT spending almost doubled from 2.1 trillion US dollar (USD) in 2002 to 3.7 trillion USD in 2013 (OECD, 2002; Rivera & van der Meulen, 2014). A remarkable amount of the global market is spent on software - valued at 299 billion USD in 2013, having a constant growth rate of about three up to four percent per year (cf. figure 1.1). Hence, the software market is one of the most rapidly growing sectors in ICT (Fois & Lysonick, 2012).

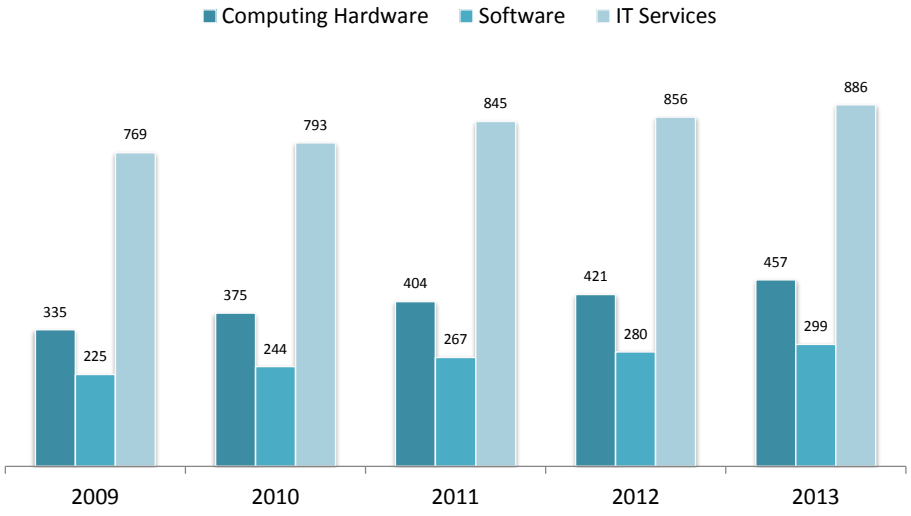


FIGURE 1.1: Worldwide IT spending in billion USD by sector between 2009 and 2013 (adapted from Fois and Lysonick (2012); Rivera and van der Meulen (2014)).

Nowadays, an increasing amount of software is developed for the needs of an open marketplace rather than for one specific customer (Regnell & Brinkkemper, 2005). Both scholars and practitioners developed different terminologies to refer to this specific type of software, such as standard software, packaged software or product software. Latter term has gained most attention by researchers during the last decade, thus it is also used in the context of the present thesis. Xu and Brinkkemper (2005) define product software as follows:

> *"A software product is defined as a packaged configuration of software components or a software-based service, with auxiliary materials, which is released for and traded in a specific market."*

A recent survey with 498 German CIOs, conducted by Buxmann, Diefenbach, and Hess (2011), shows that nowadays almost two third of German software companies rely on standardized software solutions (cf. figure 1.2). Furthermore, most of the companies believe that the demand for standardized solutions will further grow (cf. figure 1.3). Considering these two developments, the products software industry appears to be an attractive business domain for existing companies as well as for startups.

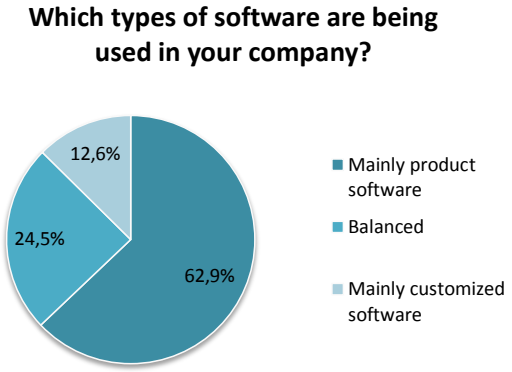


FIGURE 1.2: Use of different types of software in German software companies (adapted from Buxmann et al. (2011)).

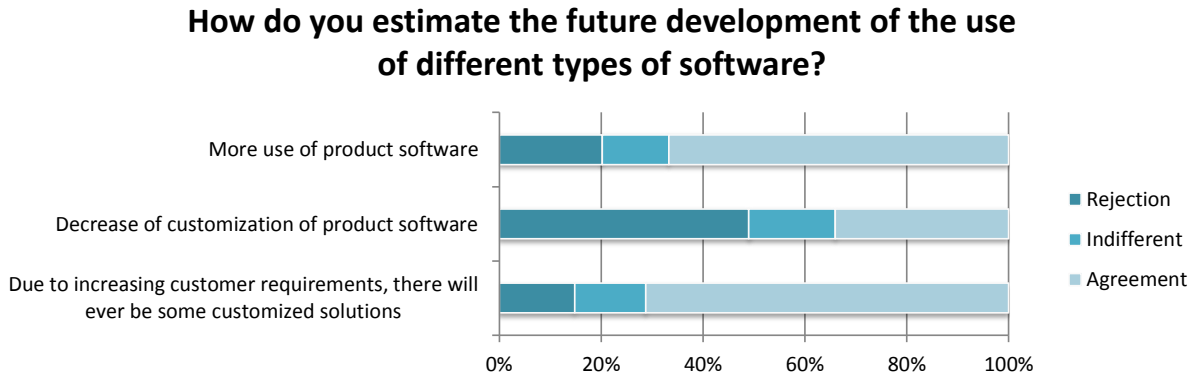


FIGURE 1.3: Estimation of 498 German CIO's of the future development of different software types (adapted from Buxmann et al. (2011)).

Product software on the one hand has the main advantage that a product - once built - can be reproduced with almost zero costs, enabling product software companies to offer millions of copies to the market (Buxmann et al., 2011; Cusumano, 2004). On the other hand, product software does have significantly shorter life cycles than traditionally manufactured goods, which implies a real entrepreneurially-oriented challenge for product

software companies, when it comes to managing such products (Lippoldt & Stryszowski, 2009). Building and delivering a software product and its subsequent versions is thus not only a challenge for software developers, but also particularly for software product managers, who are constantly concerned with portfolio management, product life cycle management and various other tasks (Ebert, 2007; Fricker, 2012; Haines, 2009; Sawyer, 2000).

According to Kittlaus and Clough (2009), the software producing industry is growing faster than any other industry. Technology and thus platforms are emerging rapidly. Product software companies are constantly concerned with the improvement of their product portfolio. Improving the portfolio can range from updating a current product version and adding features to simply fixing bugs over different releases. Once in a while, however, companies need to respond to major changes in the environment of the product by migrating to a new technological level (Malton, 2001). Besides opening up new opportunities, new technologies have the advantage that maintaining the product becomes less expensive (Haines, 2009). In information technology, migration is often referred to as *"[. . .] the process of moving from the use of one operating environment to another operating environment that is, in most cases, thought to be a better one"* (Rouse, 2005). Migration can also involve changes in hardware platforms, architecture as well as the client interface, which can lead to a completely new version of a product (Torchiano et al., 2008).

Besides responding to platform-related changes, product companies need to constantly review the performance of their current products in order to stay competitive in the market. It is important to question, whether a product still meets the organization's business objective and, if not, to rebalance and rationalize the portfolio (Jansen, Popp, & Buxmann, 2011; Haines, 2009). Rebalancing and rationalizing a portfolio amongst others means to replace a product by a follow-up product. Furthermore, it is relatively easy and cheap to enter the product software market. This forces product software companies to permanantly innovate their products in order not to forfeit market shares.

As previously described, not only the rapid technological development and increasing maintenance costs, but also general market pressure should encourage software organizations to reconsider the introduction of a new product version instead of constantly updating the old one (Haines, 2009). In the present thesis, every process that is related to a major change of a current software product that leads to a new product (version) is in the following referred to as a *software product overhaul*.

1.1 Problem statement and research objective

Every few years, a product overhaul is an essential task for product software companies, mostly entailing a large-scale transition from one technological platform to another such as from Windows to client-server or from client-server to the web. With a product overhaul, companies can improve their product using state-of-the-art technology, possibly enter new market segments and thus reach new customers - leading to an increase in market shares and profits - as well as decrease maintenance costs.

> *"The more you struggle - that is, the more time, money and people you pour into product development [...] in the hope of a turnaround - the deeper you sink and the quicker you die" (Cusumano, 2004).*

Apart from the benefits and opportunities, a product overhaul can also be a heavy burden for companies in terms of investments and loss of time (Kittlaus & Clough, 2009). K. H. Bennett and Rajlich (2000) constitute that *"[...] changing to another [software product] is proving extremely expensive, technically difficult, and time consuming."* A survey of the Standish Group conducted in 1995 revealed that 84 percent of 8000 US-based software companies did not finish new product introductions on time, on budget, or with all features implemented (Hoch, Roeding, Punkert, Lidner, & Muller, 2000). A current survey of the Standish Group showed that these numbers did not change much within the last decade, as general software projects still have a failure rate of 61 percent (Standish Group, 2013). Regarding the development of software products, also Michael Cusumano, professor of management at the MIT Sloan School of Management, makes a similar observation. He states that 75 up to 80 percent of such projects are late and over budget (Cusumano, 2004). These observations are alarming as the late introduction of a follow-up product can cost a company its position in the market as the delay of new product introduction can decrease a firm's market value by 5.25 percent on average (Hoch et al., 2000).

One of the participants (Appendix D, Company E, 130) of the multiple case study, which has been conducted in the course of this thesis (cf. section 2.4), summarizes the dilemma of the product overhaul process as follows:

> *"[...] I have been in [the] software [business] for 23 years, I have been trough this process at least six or seven times in the last 15 years with different companies, and it's always a painful process, always. [...] for a lot of companies that's just the end of the company very often even."*

Considering the previously mentioned facts, it is surprising that only little research has been done regarding the process of a product overhaul. Many researchers have focused on corporate legacy system migration, mainly investigating the technological aspects of such a migration (Andrikopoulos, Binz, Leymann, & Strauch, 2013; Carrière, Woods, & Kazman, 1999; Cordy, 2006; Correia, Matos, Heckel, Koutsoukos, & Andrade, 2006; Hunold et al., 2008; Khadka, Batlajery, Saeidi, Jansen, & Hage, 2014; Khadka, Reijnders, Saeidi, Jansen, & Hage, 2011; Khadka, Saeidi, Idu, Hage, & Jansen, 2012).

The process of a software product overhaul however is not only a technical endeavor but also a significant entrepreneurial challenge for product software companies (Khadka, Saeidi, Jansen, Hage, & Haas, 2013). First, it concerns the questions: *"what do we keep, what do we evolve, what do we stop?"* (Ebert, 2007). Furthermore, it involves decisions regarding investments (how much do we invest at the right moment in time?), positioning of the product in the market (do we keep on serving the same market segment?), internationalization (shall we go international?), the personnel (what are we doing with the developers, who are not aware of the new technology?) as well as the business or delivery model (is it worth to consider a software-as-a-service solution?) (Dubey & Wagle, 2007). These are just some examples revealing that a product overhaul involves several organizational challenges and also requires elaborate decision-making before actually moving to a

new product. Staudenmayer et al. (1998) revealed that, if a company is targeting another market segment with its new product, organizational changes in structure and composition such as the reorganization of the development team are not rare. This means that a company should be aware of the changes that are caused by the decisions they make. Besides all these decisions, product software companies face the challenge of overhauling their product without dissatisfying or even losing any customers.

Summarizing the previous argumentation, the formal problem statement of this thesis can be stated as follows:

"A product overhaul is a crucial part of a product software company's life cycle. However, it also appears to be a heavy burden for software organizations in terms of investments and loss of time. Despite those facts, there is little knowledge available about the process of a software product overhaul. This lack of knowledge leads to increased risk and failures for software firms."

To my best knowledge, there is no study, which has explored the phenomenon 'software product overhaul' from an organizational point of view. The main purpose of this research is to include multiple example case companies to gain insight on how different product software companies deal with that process. The investigation covers furthermore, what strategies exist to approach the overhaul process. One main part of the practical outlook of this thesis is to discuss the challenges and to determine countermeasures that help to approach these challenges. Since research about this phenomenon is scarce, this thesis aims to increase the awareness and knowledge regarding the product overhaul process in practice. Furthermore it serves as a source of inspiration for those companies, who are concerned with a software product overhaul as well. This appears to be highly relevant, since software companies tend to base their decision-making on tacid knowledge, experience and subjective judgements (Chan, Chan, & Tang, 2000). With this thesis, implicit knowledge about the process is made available, which shall help product software companies to successfully finish their product overhaul processes.

Summarizing, the formal objective of this thesis can be stated as follows:

"The main goal of this thesis is to investigate the product overhaul process from an organizational point of view, gaining insights on how different example case companies deal with that process."

1.2 Academic and societal contributions

This thesis has the main objective to investigate the product overhaul process from an industrial perspective. In this context, it offers both academic and societal contributions, which are listed in the following.

Academic contribution

The examined phenomenon *software product overhaul* has not gained much attention in academic contexts. Torchiano et al. (2008) conducted research about migration projects in practice. However, the researchers focused on investigating the migration process of software companies from a technological point of view. Hence, the research revealed valuable information about the technological change that is linked to the migration of software. Any other aspects or challenges were however not considered.

Consequently, crucial information is missing about the overall characterization of the product overhaul process, leaving a gap in current literature. Most of the currently existing literature is not specifically focused on the overhaul process of product software, but rather interested in the migration of legacy systems within large organizations. In comparison to legacy systems, which need to be replaced because of their partly obsolete condition, the overhaul process of product software is different as it involves a multitude of other factors. It is for example a big challenge to innovate and replace a product, while the customer is constantly being served with the best solution.

As the overhaul process of product software appears to be an essential task for companies every now and then, this thesis aims to take a first step to fill the gap, which is currently existing in literature. There are several parts of this thesis that contribute to the knowledge base of scientific literature:

- First of all, a systematic literature study is conducted, which presents an overview of state-of-the-art literature about the overhaul process of software products.
- Besides, several artifacts are derived from a multiple case study. A comparison table provides an overview of the companies, whose product overhaul processes had been analyzed.
- Moreover, the most important phases of a product overhaul in software companies is translated into a process deliverable diagram (PDD), which has been proposed by van de Weerd and Brinkkemper (2008) to facilitate the understanding of complex processes.
- Another essential artefact of this thesis is represented by several *challenges- countermeasures models*, which reflect on the challenges that appear during a software product overhaul as well as the countermeasures that are suitable to approach these challenges.
- Finally, the *product software overhaul framework* constitutes a model, which illustrates a high-level overview of a software product overhaul.

Societal contribution

By means of a multiple case study, which is performed in the course of this thesis, this thesis contributes to a better understanding of the product overhaul process in practical contexts. Even though it can be a challenging task for product software companies to move from one product to another - particularly in terms of investments and loss of time - it seems to be inevitable due to manifold reasons such as changing technology or market pressure. Many companies are struggling with replacing their products or introducing new product versions (Hoch et al., 2000). Thus, it is even more surprising that there is no study yet that investigates the overhaul process from an industrial perspective. Due to this lack of knowledge about structured, standardized approaches, many companies lack a structured plan to start their overhaul projects, resulting in a high risk of failing. Even experienced organizations make ad-hoc decisions about the phase-out process of a software product as well as its replacement by a follow-up product (Jansen et al., 2011). Hence it seems to be of big importance to investigate what software companies typically experience while overhauling their product or product line.

This research is based on a rather practical approach, aiming to identify the characteristics of the overhaul process in practice. The artifacts, which were mentioned previously, do not only contribute to the scientific knowledge base, but they are also aligned to support software companies in practice. More specifically, this thesis contributes to societal contexts as follows:

- **Comparison table:** A comparison table provides a clear summary of the analyzed case studies. This table serves as a source of information, which can be used by other companies when concerned with any of the product overhauls that were analyzed. More specific information is furthermore provided by the analyis of the individual case studies.
- **PDD of major phases:** A PDD of the major phases of the software product overhaul process is established. This model is valuable for other software companies in such a way that they can consider any possible step of a product overhaul in their planning processes. This shall help companies to gain clarity about their overhaul process right from the start.
- **Challenges-countermeasures model:** By means of the outcomes of the multiple case study analysis, several *challenges-countermeasures models* are established. Software companies shall benefit from these models as the models both list the challenges of product overhaul processes and suggest countermeasures in order to approach these challenges.
- **Product software overhaul framework:** This model serves as a super-model of this thesis, summarizing the findings of the the multiple case study. This model represents a guideline for software companies that makes clear what aspects need to be considered during the software product overhaul process.

1.3 Outline of the thesis

The remainder of this thesis is structured as depicted in figure 1.4.

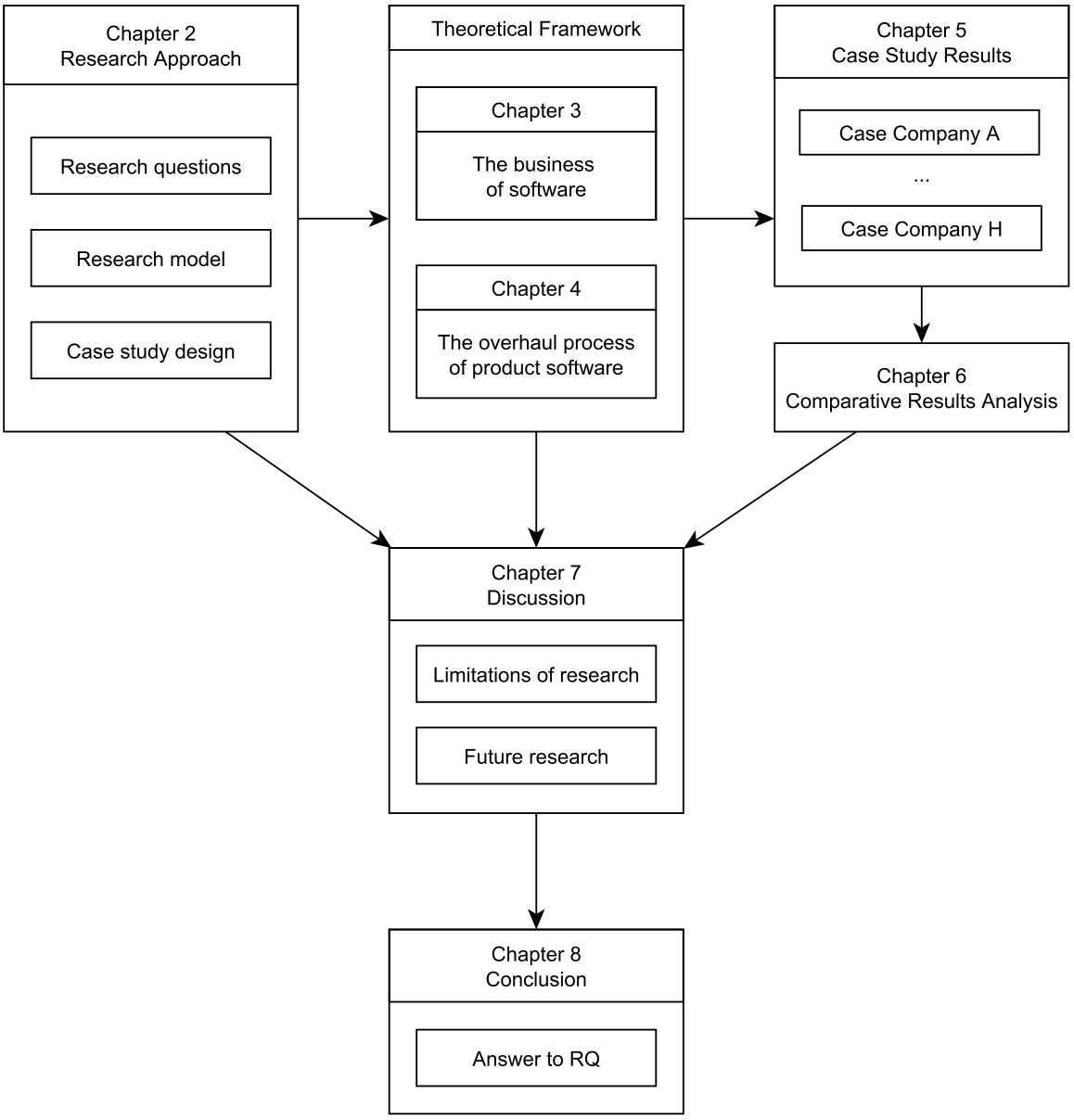


FIGURE 1.4: Outline of the remainder of this thesis.

Chapter 2 provides a detailed explanation of the methodological approach of the present thesis. First, the research question and its sub questions are stated. These central questions are answered in the course of this thesis. Besides, the research model is presented, which represents a project plan entailing each step and result of the thesis project. Finally, details about the case study design, data collection procedures as well as data analysis

are revealed. These sections are carefully elaborated as it is crucial that the reader can comprehend how the practical part of this thesis was executed.

Chapter 3 and chapter 4 represent the theroretical framework of this thesis. Theoretical information concerning the research objective is essential, as it also provides important information for the execution of the practical part. Chapter 3 on the one hand is providing a high-level overview of the software business in general, introducing its characteristics and some specific terms, which are relevant for the remainder of the thesis. Chapter 4 on the other hand examines the overhaul process from a theoretical point of view.

In Chapter 5 the results of the multiple case study are presented for each of the individual case companies. This chapter reveals relevant practical insights on the software product overhaul process. Furthermore, the individual case studies provide detailed input for the comparative results analysis in chapter 6. This analysis focuses on elaborating similarities and differences between the individual case studies. Moreover, several models are presented that contribute practical knowledge.

Chapter 6 contains a discussion of the results of this thesis. The limitations of this research as well as options for future research are discussed.

Chapter 7 concludes this thesis by presenting a summary of the most important findings of this thesis. Moreover, the research questions of this thesis are being answered.

Chapter 2

Research Approach

In the following sections, the research approach of this thesis is elaborated. First, the main research question and several related sub research questions are stated and explained. Besides, a simplified version of the research model is shown, which illustrates the methodology of the research. One of the main sections of this chapter deals with the explanation of the qualitative research approach, which is designed as a multiple case study. Eventually, it is depicted how validity concerns are being handled in the course of the research.

2.1 Research questions

In order to approach the problem, stated in section 1.1, an overall research question is provided to be solved in this research. This question is answered by considering a certain number of sub research questions. Answering these questions provides input for academic contexts, where the phenomenon of a product overhaul has only gained little attention so far. Furthermore, providing answers for the research questions can help software product companies, which are concerned with the replacement of legacy products.

The main research question is stated as follows:

RQ: *"What are the characteristics of the major product overhaul process in software companies?"*

The process of a product overhaul has not been widely investigated by researchers so far. However, it is proven to be an inevitable task for product software companies within their life cycle. Most companies have to fundamentally renew their product at least once every decade. As many of them are not aware of the specific factors they have to consider during a product overhaul, a more practical research approach seems to be required, aiming to derive a broad framework for the overhaul process. To reach this purpose, the following sub research questions need to be considered.

SQ1: *"How can a product overhaul process be defined and where can it be positioned in the domain of product software?"*

With this first sub research question, the product overhaul process is introduced and generally explained. Thinking of the term software product overhaul as a synonym for the renewal of a software product, one might pose the question: Does a product overhaul entail a large transition to a new operating environment, is it a slight alteration of the graphical user interface or even just the addition of features? Thus, this sub research question is about defining the nature of a product overhaul, in which context the term is suitable and what it consists of. The question is approached by a literature study (cf. section 4.2), providing the current state of the art of academic literature. Eventually, it is also essential to clarify, where the software overhaul process can be positioned in the domain of product software. Therefore, a thorough introduction of the software business in general is provided in chapter 3, which is the first part of the theoretical framework of this thesis.

SQ2: *"What are the reasons for product software companies to fundamentally overhaul their products?"*

Product software companies do not just overhaul their products without having any specific reason. This sub question aims to determine the triggers and reasons that induce software companies to start an overhaul project. Previous literature presented some explanations, why some products need to renewed (cf. section 4.3). Most information that leads to a solution of this question is however provided by product software companies themselves. It is of interest to find out, whether companies do specifically know these reasons or whether these processes were just randomly initiated. The triggers for the product overhaul process are important to understand for this research as well as for practitioners as one should be aware that a product overhaul is a challenging task to accomplish and it would be irresponsible to start this process without having any specific reasons.

SQ3: *"What are the strategies that product software companies apply to approach the overhaul process?"*

The third sub research question is focused on clarifying what different paths or strategies exist that product software companies can apply to approach the product overhaul process. Besides it is revealed, which strategies are chosen by different product software companies and why. There it is also crucial to find out, whether one specific strategy was successfully applied or whether it led to failure. This question is answered by both literature (cf. section 4.4) as well as the multiple case study, the more practical part of this thesis.

SQ4: *"What aspects affect the product overhaul process in software companies and what are the most significant challenges?"*

The last sub research question has a more practical outlook and is answered by a qualitative research approach (cf. section 2.4). It aims at investigating what aspects do affect the product overhaul process in software companies. Furthermore, the major challenges that can occure during the overhaul process are elaborated.

All in all, the overall research question of this thesis is solved by considering four sub research questions. Each of these sub research questions receives input from different sections of this research. While the first sub research questions (SQ1-SQ3) are answered both by means of a literature study and the multiple case study, the last sub research question (SQ4) is solely approached by applying a qualitative research method. An overview of the research questions as well as the applied methods to solve the questions and the referring sections is provided by table 2.1.

Number	**Research question**	**Referred section**
RQ	*What are the characteristics of the major product overhaul process in software companies?*	Overall thesis
SQ1	*How can a product overhaul process be defined and where can it be positioned in the domain of product software?*	Chapter 3, section 4.1
SQ 2	*What are the reasons for product software companies to fundamentally overhaul their products?*	Section 4.2, chapter 5, chapter 6
SQ3	*What are the strategies that product software companies apply to approach the overhaul process?*	Section 4.3, chapter 5, chapter 6
SQ4	*What aspects affect the product overhaul process in software companies and what are the most significant challenges?*	Chapter 5, chapter 6

Table 2.1: Research questions of this thesis with referred sections.

2.2 Research model

In this section, both a simplified (cf. figure 2.1) as well as a more detailed research model (cf. figure 2.2) is presented. Both research models illustrate the methodological approach that was chosen to answer the research questions of the present thesis. Hence, it provides an overview of the steps that were taken in the course of the overall thesis project, allowing for intersubjective comprehensibility of the research.

The simplified version of the research model visualizes the basic components of the present research (cf. figure 2.1). As an essential contribution of this thesis, a systematic literature review forms the theoretical fundament of both chapter 3 and chapter 4. Additionally, it also provides input for the integral part of this thesis: the case studies. Before starting the data collection and data analysis processes of the case studies, however, the development of an interview guideline was necessary. This guideline contains the main questions that are asked during the expert interviews. Next to the interviews, a document study served as an additional data collection method. Thus the data collection process is based on using multiple sources of evidence. Collected data was further analyzed using the coding

principle known from Grounded Theory (Corbin & Strauss, 1990), which required several steps from transcribing the interview until the refinement of the interview guideline. Eventually, the results of the individual case studies were validated in order to enhance the trustworthiness and the quality of the findings.

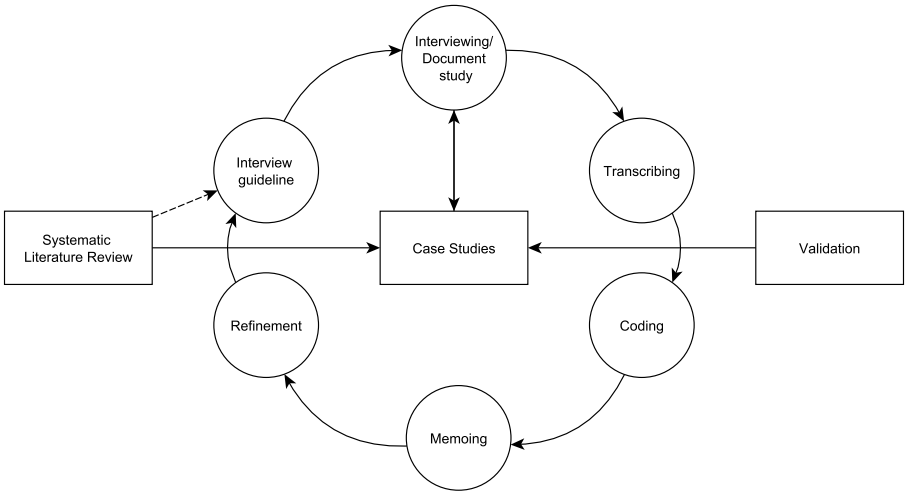


FIGURE 2.1: Simplified version of the research model.

Process deliverable diagram

A process deliverable diagram (PDD) serves as a more elaborate research model, providing a clear overview of the activities and deliverables of this research. The construction of the PDD is based on the rules and notations of van de Weerd and Brinkkemper (2008). They define a PDD as a meta-modelling technique, which consists of two integrated diagrams. On the left-hand side of the diagram, the meta-process modelling of the activities is represented, which is based on a UML activity diagram. On the right-hand side of the diagram, the meta-deliverable modelling of the deliverables is described, which is based on a UML class diagram. The deliverables on the ride-hand side are called concepts, which are derived from the activities on the left-hand side. These connections are specified by the dotted arrows.

The PDD representing this research is illustrated in figure 2.2. The first organizational task of this research project was to create a short and a long proposal, which were both explained in a presentation. The main purpose of these proposals was to define how this research is being approached in terms of the methodology. The second main step after writing the proposals was the establishment of a literature review. After conducting the review and selecting relevant studies, the results of the selected studies needed to be incorporated into the theoretical framework. With the completion of the theoretical framework, the more practical part of the thesis began. A multiple case study approach was chosen as an appropriate research method. After preparing the case studies and selecting suitable case companies, both data collection as well as data analysis followed.

After having analyzed the data, practical conclusions could be drawn, which in combination with the research approach and the theoretical framework represent the main artifacts of this thesis.

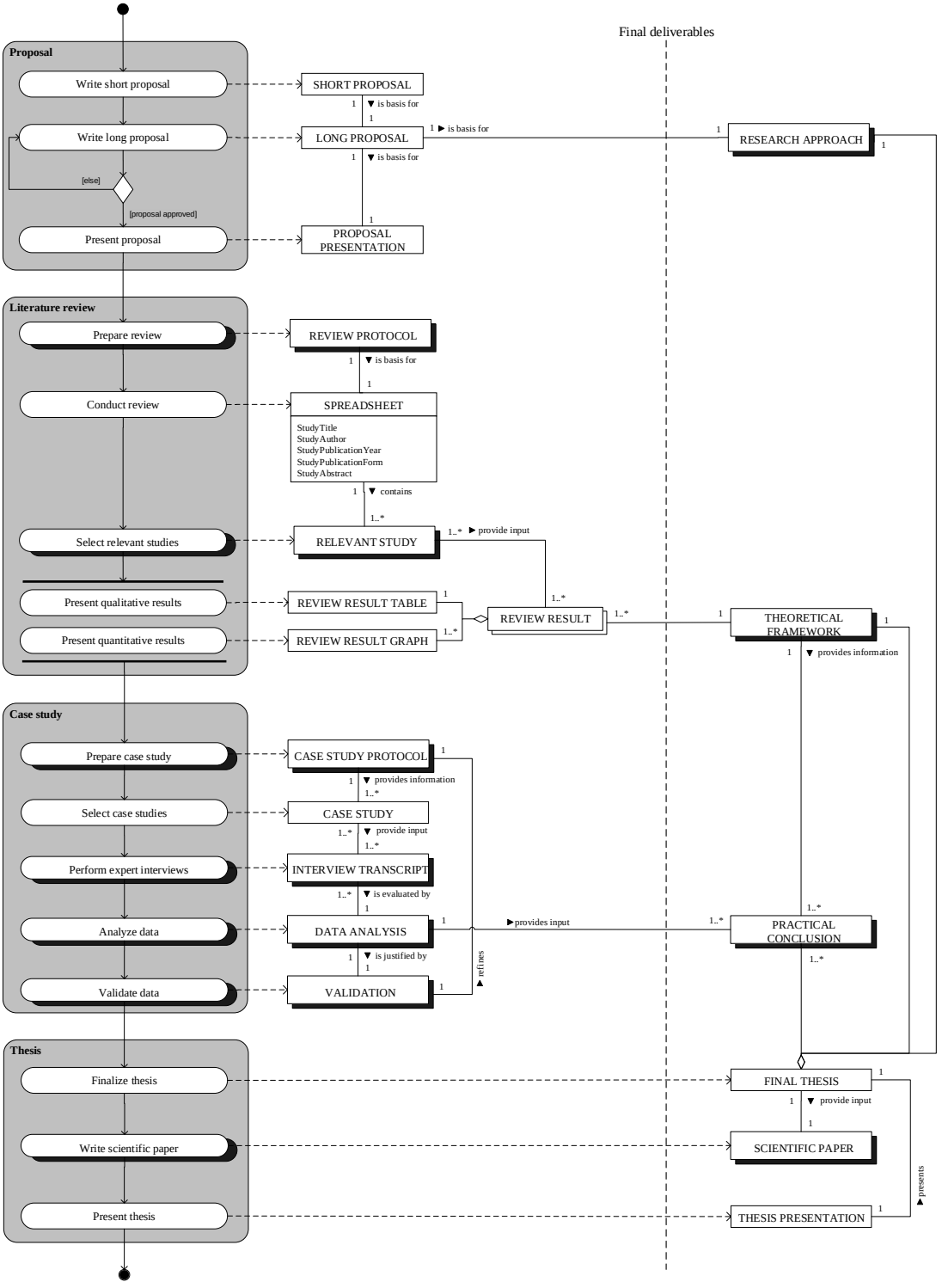


FIGURE 2.2: The process deliverable diagram describing this research.

2.3 Literature review

In order to identify, evaluate and interpret all the research that concerns the process of a software product overhaul as well as the research questions stated in section 2.1, a literature review was performed. Besides applying a systematic approach proposed by Kitchenham (2004), the snowballing technique was used (Jalali & Wohlin, 2012). A systematic literature review is a means to identify any gaps in current research and to provide a solid framework for further research activities - being multiple case studies in this research.

In order to determine primary literature to be reviewed, a specific search strategy was established. Taking the approach of Kitchenham (2004) into account, the search strategy consisted of three major steps:

1. Define electronic libraries to be searched.
2. Define search terms.
3. Define and apply inclusion and exclusion criteria.

As by definition, the first step of applying the literature review strategy was to determine those liberaries, which appeared to be appropriate for this thesis. Google Scholar [1] as well as the ACM Digital Library [2] were selected as the two electronic libraries. This selection was chosen as the two libraries are commonly used by researchers. Besides, they cover a wide range of academic literature. The ACM Digital Library furthermore mainly contains articles that are related to the field of information science. Both libraries were accessed via a Proxy-server from Utrecht University in order to increase the chance of retrieving more articles.

The second step of the literature review was to establish specific search terms, which were applied. As the main goal of the literature review was to provide an in-depth overview of existing literature, search terms were based on two major areas of interest. On the one hand, it was important to cover as much literature as possible containing information about the software business in general, particularly product software as well as the concepts surrounding the term product software. On the other hand, the literature review aimed at delivering findings about the overhaul process of software. Therefore, several terms were chosen, which are related to a software overhaul such as software migration, software modernization or software transition. The final search terms are stated as follows:

Software business

Software business OR product software OR software product management OR software lifecycle management

[1] http://www.scholar.google.com

[2] http://dl.acm.org.proxy.library.uu.nl/

Software overhaul
Software overhaul OR software migration OR software transition OR software replacement OR software modernization OR software rebuild OR software redevelopment OR software revision OR major software upgrade OR major software update

As a third step before performing the actual literature review, inclusion and exclusion criteria were defined in order to guarantee the inclusion of relevant studies only. These criteria were defined as follows:

Inclusion criteria

- Studies that are either written in English or German
- Studies that are peer-reviewed

Exclusion criteria

- Studies that are inaccessible
- Studies that are published before 1999
- Studies that are not closely related to the research topic

This means that studies were included that were either written in English or German or that were peer-reviewed articles. The first decision of inclusion criteria was made as the primary researcher is able to understand the two mentioned languages. The second decision was made in order to guarantee high quality outcomes of the literature review. Exlcuded were studies that were not accessible from either Google Scholar or the ACM Digital Library. Furthermore, studies that were written before 1999 were exluded from the literature review, as the main goal was to provide an overview of state-of-the-art literature. Eventually, studies that are not closely related to the research topic were not included. This was for example a study, which investigates the migration of legacy information systems, but not the overhaul of a software product.

After finishing with the aforementioned preparation of the strategy, the literature review was conducted by applying the following steps:

1. Identify relevant studies
2. Select studies based on title and abstract
3. Select studies based on inclusion and exclusion criteria
4. Define relevant studies

To identify all relevant studies, the previously mentioned search terms were applied to the digital libraries. In order to further limit the amount of studies, primary studies were selected by browsing the title and abstract of the relevant studies, which tremendously reduced the amount of studies. Furthermore, the inclusion and exclusion criteria were applied to the derived primary studies. This further reduced the amount of relevant studies by almost three fourths. The last step of the literature review was to remove duplicates from the search results, after which the final number of relevant studies could be determined. An overview of the exact numbers of extracted studies is presented in figure 2.3.

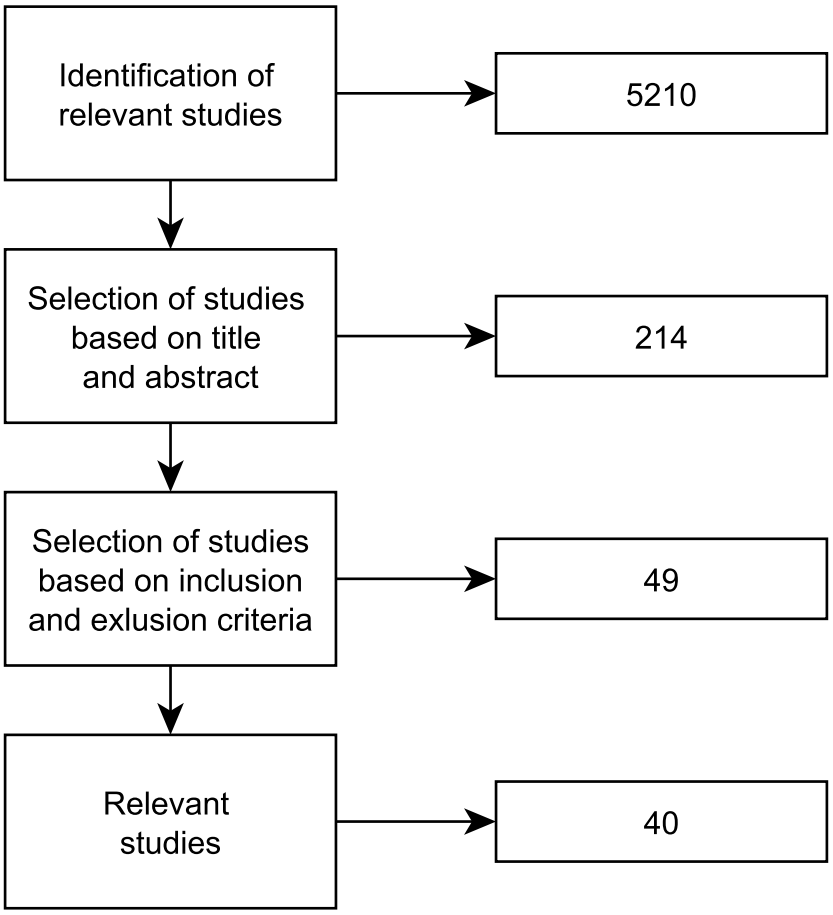


FIGURE 2.3: Literature review process with resulting amount of relevant studies.

As the number of relevant studies did not appear to be sufficient, all relevant studies were browsed for further references, which is refered to as snowballing (Jalali & Wohlin, 2012). The combination of systematic literature review as well as snowballing thus resulted in a selection of 74 papers and books. The review revealed that literature concerning the overhaul process of product software is scarce. The term *product overhaul process* did not lead to any search results. A lot of studies are focused on investigating the process of legacy system migration, with the emphasis on migrating large scale information systems

in organizations. This also explains the high reduction of relevant studies after applying exclusion criteria. The results of the literature review are presented in the theoretical framework of this thesis, represented by chapter 3 and chapter 4.

2.4 Case studies

This thesis follows a qualitative research approach, which aims at investigating the "novelty" of a research field from a rather practical point of view (Flick, 2009). As a research method, a case study approach was chosen to be appropriate to gain insights into a practical real-life context in order to solve the aforementioned research questions (cf. section 2.1).

Case studies are the most widely used qualitative research method in information systems research and well suited to understand the interactions between IT-related innovations and organizational contexts (Darke, Shanks, & Broadbent, 1998). Benbasat, Goldstein, and Mead (1987) describe a case study as means to examine a phenomenon in its natural setting, employing multiple methods of data collection to gather information from one or a few entities. They further give reasons for when a case study is an appropriate method in information systems research. First it is useful to learn about the state of the art of a phenomenon and generate theories from practice rather than testing hypotheses. Moreover, case studies help to answer any kind of "how" and "why" questions, which aim at understanding complex processes. Additionally, case studies are an appropriate approach to investigate a topic that has not been researched in detail yet. Considering these reasons and the argumentation in section 1.1, case study research was chosen as an appropriate method for investigating the phenomenon software product overhaul.

The structure of the case studies is mainly oriented on the research of Yin (2009), Jansen and Brinkkemper (2008) and Runeson and Höst (2008). Latter both provide guidelines for conducting and reporting case study research in software engineering. In the following subsections it is depicted, how the case studies were performed in terms of design, data collection and data analysis. Furthermore, basic validity aspects are discussed.

2.4.1 Case study design

The objective of the case studies was to gain insights on how product software companies experienced a product overhaul process. For this purpose, exploratory research appeared to be appropriate. This type of research is about *"[...] finding out what is happening, seeking new insights and generating ideas and hypotheses for new research"* (Runeson & Höst, 2008).

The case studies are designed as holistic multiple case studies (cf. Yin (2009)). The holistic multiple case study design means that one phenomenon/unit of analysis is being investigated with several cases (cf. figure 2.4). The unit of analysis in this research is represented by the phenomenon 'software product overhaul', which is explored at several specific case companies that all belong to the context of product software companies.

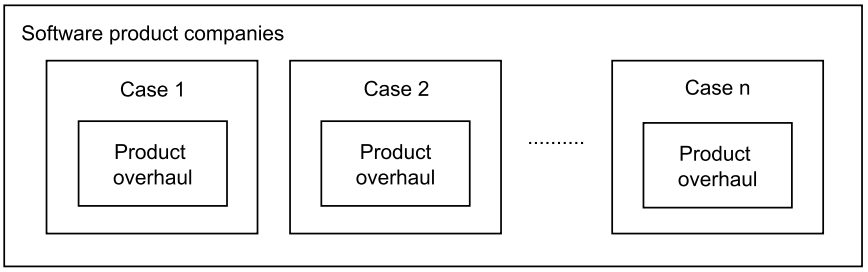


FIGURE 2.4: Holistic multiple case study design (adapted from Yin (2009)).

Why multiple case studies?

There are various reasons, why multiple case studies are appropriate for this thesis. Jansen and Brinkkemper (2008) mention some of them as follows: Multiple case studies are a suitable method to explore the current state of practice with regard to a specific phenomenon. Furthermore, multi-case research is considered to deliver more robust results than a single case study does. Robust results are inevitable, since a solid basis of knowledge needs to be attained for further research. Eventually, it is the goal of this research to obtain measurable constructs and a conceptualized overview of the phenomenon product overhaul, which is hardly achieved by only a single case study.

Another main advantage of multiple case studies is that similarities and differences between different organizations regarding one specific unit of analysis becomes evident. This made it easier to determine a set of best practices of how software organizations nowadays deal with a product overhaul.

As multiple case studies were chosen to be included in this research, the replication logic approach of Yin (2009) was applied (cf. figure 2.5). The replication logic means that the same procedure for investigating a case company is replicated for all participants.

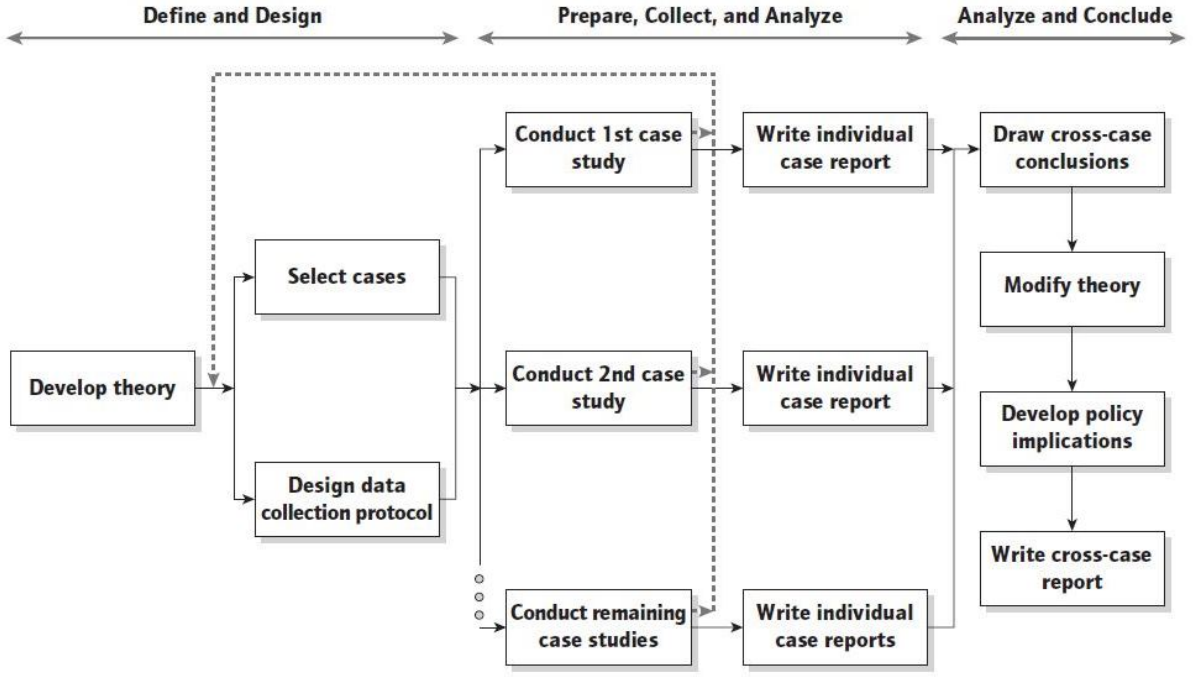


FIGURE 2.5: Multiple case study replication logic approach (cf. Yin (2009)).

A crucial step of this approach is the development of a rich theoretical framework. This framework indicates, under which conditions a particulars phenomenon is likely to be found, thus providing information for the selection of the cases.

Case selection

Besides designing a data collection protocol - in this research referred to as interview guideline (cf. section 2.4.2) - case selection was an inevitable step before starting to collect evidence.

The selection of the cases was done intentionally, however there was not a specific rule applied on what number of case studies is sufficient. Eisenhardt (1989) mentions that a number of four up to ten cases is desirable for theory building using case study research. In the multi-case research done by Jansen and Brinkkemper (2008), the authors chose six cases, which lead to sufficient results. Thus, the following rule should apply to the participating companies of this research: Each of the cases should be a software producing and distributing organization, headquartered in the geographical area of the Netherlands. Even more important was that each of these companies has experienced a software product overhaul.

Having fulfilled the aforementioned requirements, eight case companies from all over the geographical area of the Netherlands were participating in this research (cf. figure 2.6). As it is visible in table 2.2, companies from a wide range of sectors were selected. This was done intentionally as this better enables the exploration of similarities and differences of companies from different sectors. It would have been possible to include companies from one sector only, however this could lead to biased results. Furthermore, this research

aimed at establishing first empirical findings rather than generating representative results. With investigating one sector only, the results might be more represetative, however they would also be limited. Besides, all the companies are of different ages. Intentionally, some younger companies as well as well experienced companies were included in order to work out similarities and differences regarding the fact when a company has been established. The same applies for the number of employees and the annual turnover, which represent the company in terms of size. Apparently, some of the companies did not release their annual turnover for publication as they are privately owned. Eventually, a company is characterized by the number of customers - if available - as well as their market orientation. On the one hand, some companies are only serving the Dutch market, which means that they stay nationally (N). On the other hand, some companies distribute their software products internationally (IN) - which in this case means European countries only - or even world wide (WW), which means that their software products are sold all over the world.

FIGURE 2.6: Geographical distribution of case companies.

Company	Sector	Est.	Employees	Turnover	Customers	Market
A	Agriculture	1985	100	11 Mio	13,000	WW
B	Labor market	2001	50	-	400	IN
C	Content management software	1996	100	15 Mio	-	WW
D	Pension funds administration	1986	80	15 Mio	-	WW
E	Healthcare	2004	95	8 Mio	700	N
F	Insurance industry	2006	40	-	100	IN
G	ERP software	1996	320	71 Mio	10,000	WW
H	Building industry	1990	140	-	3,500	IN

Table 2.2: Overview of case companies.

Case study analysis

All cases were analyzed individually as well as in a cross-case analysis. This means that each individual case study contains a 'whole' study, from which conclusions are drawn. Such conclusions or discoveries can influence the case study design in such a way that it needs to be redesigned before proceeding with another case. Such redesign might for example entail the restructuring of the interview guideline. This part of the replication logic is represented by the dashed-line feedback loop in figure 2.5.

Case study preparation

Before any important step in the case study research was executed, a case study protocol was developed (cf. Appendix A). The case study protocol is oriented on the structure proposed by Maimbo and Pervan (2005). However, the position of the sections of the protocol was reorganized to make it more readable to the persons, who receive it.

The case study protocol specifically defines the aim of the research and explains the methodological approach. Besides, it provides a short but precise overview of the background of the research, thus emphasizing the importance of the investigated phenomenon. The protocol is a document, which was shared with each participanting case company before gathering any kind of data. This was done in order to provide the case companies with an in-depth outline of the research project in order to avoid confusion and conflict

in the future (e.g. when it comes to collecting data). The protocol was kept short in order not to overwhelm participating case companies with a vast amount of information. Furthemore, the protocol was kept the same for every case company thus providing the same kind of information to everyone. This presents a means to establish a chain of evidence and thus to increase construct validity of this research (cf. section 2.4.4).

2.4.2 Data collection

According to Yin (2009), the three principles for data collection are: use multiple sources of evidence, develop and maintain a case study database and establish a chain of evidence.

Data collection in this research consisted of two procedures: expert interviews as well as document study. By having these two procedures, multiple sources of evidence were applied, which leads to some kind of data triangulation, known as a tool to increase construct validity of case studies (Yin, 2009).

Semi-structured expert interviews

Semi-structured expert interviews served as the primary data collection method in this research. Interviews are part of first degree data collection techniques, where the researcher is in direct contact with subjects and collects the data in real time (Lethbridge, Sim, & Singer, 2005). Although being time consuming and cost inefficient, expert interviews are highly interactive and thus mostly result in high quality results. Furthermore, the researcher has full control of data collection as he is actively participating in it.

At each participating company, an interview with an expert was conducted. The interviews were semi-structured, which means that the interviews contained a set of predefined and open-ended questions, leaving enough space for a discussion in between. Thus, not only foreseen information could be elicited, but also unexpected types of information could be derived from the interviews (cf. Seaman (1999)). In table 2.3, all participating experts of each case company are listed. This list reveals that most of the participants have lots of experience in the field of product software. Furthermore, most of the experts either were the founder of the case company itself or in a leading position such as the chief executing officer (CEO), chief financial officer (CFO) or the chief operating officer (COO). Thus the interviewed participants are not only involved in some of the company's processes, but are aware of the company's overall activities. This is important, as this research aims at investigating the overhaul process not only from one specific view (e.g. technological view), but also from different perspectives such as the organizational change as well as temporal and financial impacts. Besides, it is visible that in one case company, two experts were attending the interview. This leads to a total number of eight conducted interviews with nine participants at eight case companies.

Comp.	**Interviewee**	**Position**	**In comp. since**	**Background & Experience**	**Interview duration**
A	1	CEO	Several years	27 years in software business	1h 30min
B	2	Founder & CEO	2001	Language Technology	1h 10min
C	3	Product consultant	2008	Business Informatics	1h 24min
D	4	CFO	2007	Economics	1h 30min
	5	Operational manager	2004	Computer science	
E	6	COO	2013	Software engineer 23 years in product software industry	1h 2min
F	7	Founder & Product strategy	2006	IT & Business Management	1h 15min
G	8	Director product development	1999	Computer science	1h 45min
H	9	Managing director & Lead of development	1990	Computer science	53min

Table 2.3: Overview of interview participants.

Before conducting the interviews, an interview guideline was established. The guideline consisted of three parts. First, the interview started with an introduction of the research by the researcher in order to provide the interviewed expert with a short recap. Afterwards, the participant introduced himself and the company he is working for. The third part of the interview consisted of a set of open-ended questions, which were divided into several parts. In order to simplify the analysis of collected data and to avoid the loss of any kind of essential information, each interview was digitally recorded.[3]

[3]For this purpose, the native iOS app Voice Recorder was used: Link: `http://smartlof6.wix.com/`

After each interview, the interview guideline was refined in order to improve the outcome of the following interviews. This refinement took place during the whole data collection phase with a duration of two months (May - June 2014). Mostly, the refinement consisted of adding questions, updating questions or restructuring the position of some parts. However, the key elements of the interview guideline were consistently retained in order to guarantee validity of the research. The final version of the interview guideline can be found in Appendix B.

Document study

Besides conducting semi-structured expert interviews, it was planned to use digital or paper-based documentation and presentations as another data source. However, many of the participating case companies did not have specific documents that were closely related to their overhaul process. Nevertheless, a few companies did provide some information in terms of power point presentations or paper documents. Furthermore, information was taken from the websites and brochures of the case companies, which mainly contained general information about the case company such as a description of their product and their mission. This was helpful to avoid misspellings of company-specific terms and to complement information that was not discussed in the interview. Moreover, this kind of information increased the researcher's knowledge about the company itself and its processes before the interview, so it served as a means to facilitate the conduction of the interview (an overview of the documents is provided in Appendix C).

Case study database

Acoording to the second principle of Yin (2009), another important role during data collection is represented by the case study database (cf. figure 2.7). The use of such a database is also suggested by Jansen and Brinkkemper (2008) in order to avoid the loss of any kind of obtained data. The database consists of material that has been provided by the case organizations (e.g. documents, presentations etc.) on the one hand, material that was actually obtained by the researcher (e.g. recorded interviews, interview notes, etc.) on the other hand. Thus, all the data that was collected by means of interviews or that was provided by the case company (e.g. documents) was stored in the case study database. Each case company had its own folder, which contained the specific interview guideline, the audio file of the interview, the interview transcript (cf. section 2.4.3), documents of the company as well as e-Mail conversations (cf. figure 2.8).

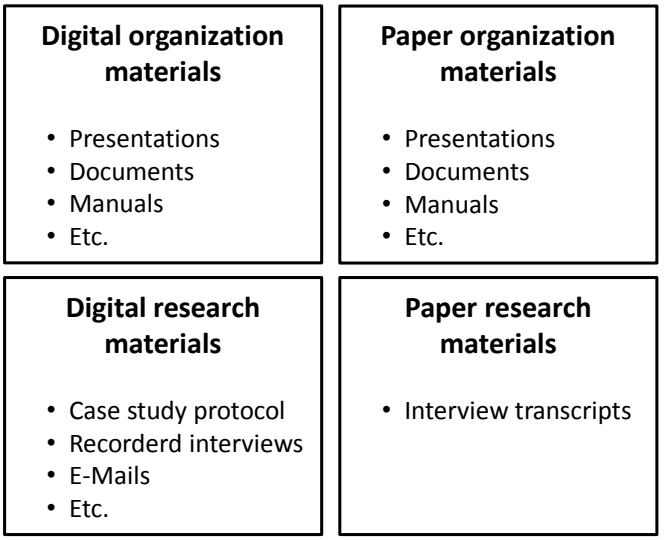


FIGURE 2.7: Case study database (adopted from Jansen and Brinkkemper (2008)).

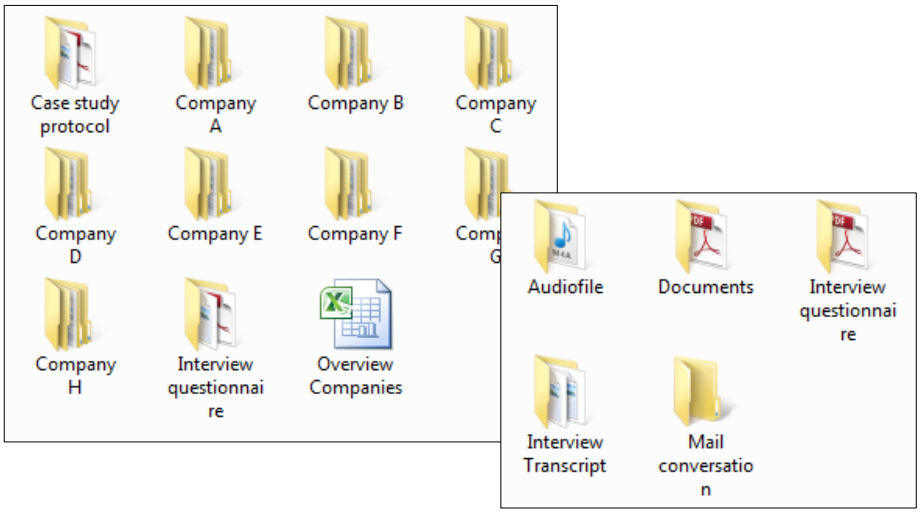


FIGURE 2.8: Structure of the case study database.

Chain of evidence

In order to fulfill the third data collection principle of Yin (2009), it is crucial to establish a chain of evidence. Thus, an interview guideline (in Yin (2009) referred to as data collection protocol) appeared to be inevitable in this project, since data was collected in multiple locations and over an extended period. It was very important to constantly update the guideline while collecting and analyzing the data, so that already obtained findings could be used for further data collection processes. Along with the case study protocol (cf. section 2.4.1), a reader can understand, where the evidence of data was derived from, thus providing an understandable chain of evidence.

2.4.3 Data analysis

In case study research, no standard approach for analyzing huge amounts of data is specified (Darke et al., 1998). Thus, data analysis was done using the Grounded theory coding principle (cf. Figure 2.9) elaborated in Corbin and Strauss (1990).

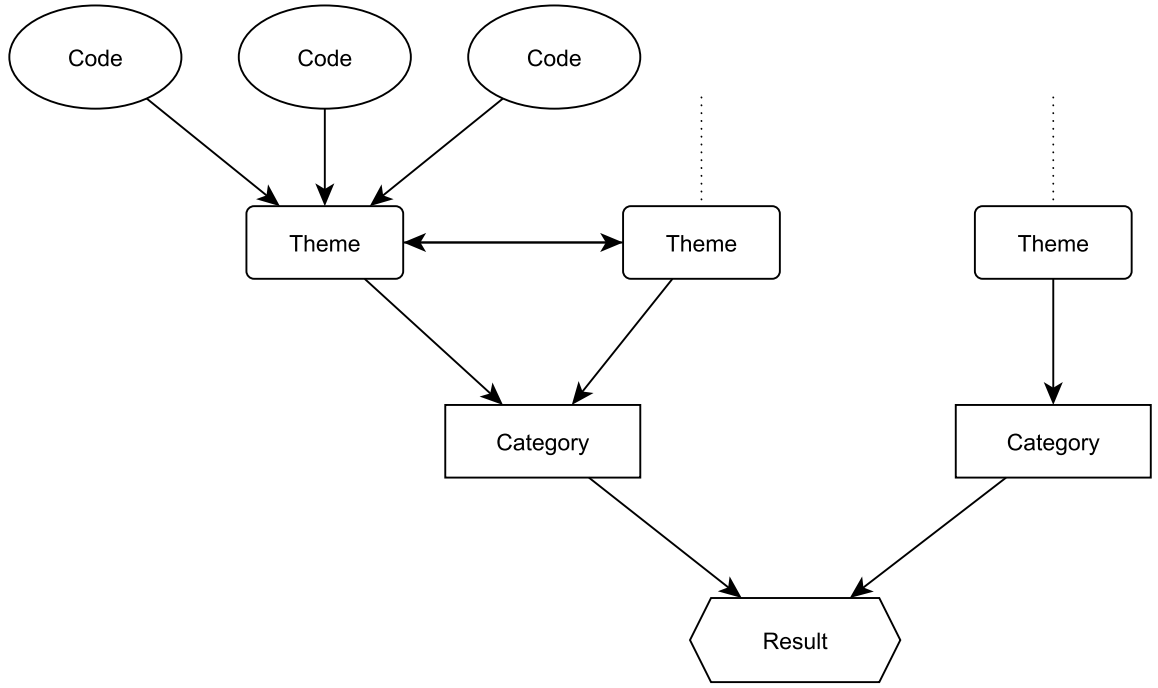


FIGURE 2.9: Data gathering by coding approach (Based on Corbin and Strauss (1990)).

This approach is specifically recommended by researchers in order to cope with the huge amount of data that is collected in case studies (O'Connor, 2012). O'Connor (2012) emphasizes that in contrast to Grounded Theory - where the focus lies on the participating individuals - the aim of case studies is *"[. . .] to generate a detailed insight of the case, its complex relationships and processes"*. Thus it is important to remember at this point that only the coding apporach from Grounded Theory was applied, not the whole methodology itself.

The approach is structured as follows: after having transcribed the recorded interviews, specific codes were built, which are based on words, sentences or paragraphs of the transcribed interview notes. Every code represents a specific theme that is closely related to this research. Different themes again were summarized in a specific category. Thus, conclusions are finally drawn from the categories. To support all these steps the research software NVivo 10 for Windows was used.[4] The audio files of the interviews were thereby transcribed word-by-word, at least the passages that contained information concerning

[4]Link: `http://www.qsrinternational.com/products_nvivo.aspx?utm_source=NVivo+10+for+Mac`

this research project. Any kind of smalltalk about unrelated things was left out of the interview transcripts. The final transcripts of each participating case company can be found in Appendix D.

After all, analyzing the data by coding the interview transcripts was done by means of NVivo. The software was of great help in this process as coded paragraphs were automatically added to the related theme. Figure 2.10 illustrates this process with the help of an example. While analyzing an interview, important sentences were highlighted in a paragraph of an interview transcript. These highlighted sentences represent a code, which belongs to a specific group of themes. In the example, the highlighted sentence is referred to the theme 'changing technology'. In the same paragraph, another code is referred to the theme 'changing technology'. Thus, already two codes are related to the theme 'changing technology'. By proceeding with this process, one receives further codes, themes and thus finally a category, to which these themes belong. In the example, the collected themes belong to the category 'overhaul reason', which contains all kind of information regarding the reasons and drivers of the overhaul process in software companies. As it is visible in the example, each information concerning a theme is clearly summarized in terms of the paragraph of the interview, the exact time information, the highlighted code as well as the Interviewee, who is responsible for the statement. This furthermore facilitates the process of reporting the case study results, as any references can be clearly marked. If any references are used within the description of the outcomes in chapter 5, they are refered to in the following structure: (Appendix, case company, paragraph). A reference of paragraph 2 of the interview with Company A for instance is thus marked as (Appendix D, Company A, 2).

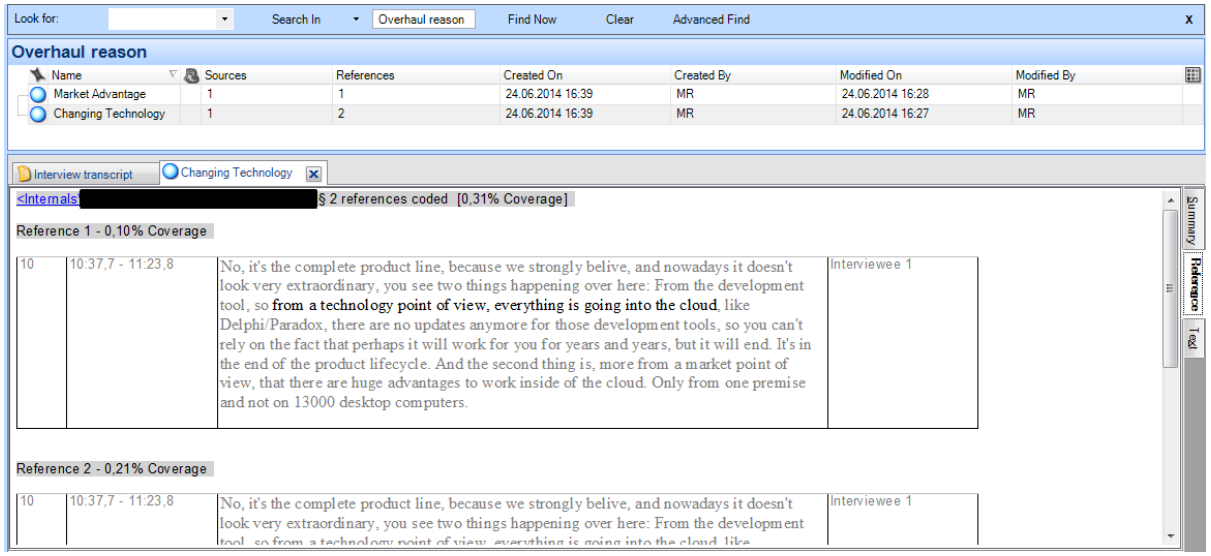


FIGURE 2.10: Specific example of the coding of an interview transcript.

By proceeding with the aforementioned process through all existing interview transcripts, the final main category system with sixteen different categories was established as follows (the complete category system with its subsequent themes can be found in Appendix E):

- Overhaul type
- Overhaul trigger
- Overhaul strategy
- Overhaul approach
- Technological aspects
- Temporal aspects
- Financial aspects
- Business model
- Project management
- Organizational aspects
- Customer interaction
- Market aspects
- Overhaul challenges
- Risks
- Countermeasures
- Lessons learned & further advice

Another important aspect of data analyis was memoing. Memoing required to comment the coding and any other process as thoroughly as possible. This was essential, since it was a complex task to keep track of every information that was gathered during theses processes. As already stated, data collection and data analysis are interrelated processes. This means that data was analyzed right after having started collecting it. This was done as the results of the analysis were directly reused in further data collecting processes. Any kind of new information thus led to a refinement of the data collection process, including the improvement of the interview guideline as well.

2.4.4 Plan validity

To establish the quality of a research, four design tests are commonly used: Construct validity, Internal validity, External validity as well as reliability (Yin, 2009). This research aimed to fulfill all of these criteria. An overview of the tests, applied case study tactics as well as a cross-reference to the phase when a specific tactic was used, is provided in table 2.4.

Tests	Case Study Tactic	Phase of research
Construct validity	use of multiple sources of evidence	data collection
	establish a chain of evidence	data collection
	review of case study outcomes by participating case companies	composition
Internal validity	use of coding approach for data analysis	data analysis
External validity	selection of typical software product companies	research design
	use of replication logic	research design
Empirical reliability	develop case study protocol & interview guideline	data collection
	develop case study database	data collection

Table 2.4: Case study tactics for dour design tests (based on Yin (2009)).

Construct Validity
Construct validity means to identify *"[. . .] correct operational measures for the concepts being studied"* (Yin, 2009). Multiple sources of evidence were used in terms of conducting interviews as well as scanning documents. Besides using multiple sources of evidence, a chain of evidence was established by applying the same case study protocol to each case study. Finally the results of the case studies were peer reviewed by the participating case companies. This process is also refered to as 'member checking' (Seale, 1999). All of the companies received the outcomes of the individual analysis so as to cross-check them in order to ensure the correctness and trustworthiness of the collected data. This is furthermore a means to minimize the personal bias of the researcher.

Internal Validity
Internal validity is seeking to establish causal relationships and distinguish spurious relationships. It is a means to make sure that collected data was correctly extracted from the case study database, which requires a structured data analysis approach. To guarantee internal validity in this research, the coding approach was applied to properly analyze data.

External Validity
To ensure external validity, the domain is defined to which the findings of a research can be generalized beyond the immediate case study to other case studies. For this purpose, the case studies were selected in such a way, so that they represent an example of a typical product software company. Moreover, the replication logic, which was presented in the case study design (cf. section 2.4.1), contributed to this part of validity as it ensured that every case study was done applying the same procedure.

Empirical Reliability
Empirical reliability implies that the operations of the study can be repeated with the same results. To ensure this aspect of validity, proper documentation of the case study procedures needed to be established. This is done by establishing and carefully maintaining the case study protocol and the interview guideline. This enables other researcher to understand and even reapeat the research. Furthermore, a case study database was established containing any kind of information that has been obtained during data collection.

Chapter 3

The Business of Software

This chapter, along with chapter 4, contributes to the theoretical framework of this thesis. It provides a broad overview of the software business in general, which is necessary to understand where the overhaul process of software products can be positioned.

As Michael Cusumano emphasized in his book *"The Business of Software: What Every Manager, Programmer, and Entrepreneur Must Know to Thrive and Survive in Good Times and Bad"*, the software business is unique and not comparable to any other kind of business (Cusumano, 2004). Software is something, which is often referred to as an *"[...] intangible economic good, with no physical form, its utility or value not even perceptible in another form"* (Kittlaus & Clough, 2009). Thus, in contrast to traditional consumer goods such as cars, software is characterized by the technology, they are made with, and the functionality they offer. With its functions, software can support and solve various kinds of everyday humane problems, which might range from simple calculation tasks to the navigation of cars or the construction of houses. Generally, software is a digital good, which consists of human knowledge that is transfered into bits and bytes (Kittlaus & Clough, 2009). As any other digital good, software can be copied and reproduced with almost zero marginal costs, enabling software companies to offer millions of copies of their software to the market (Buxmann et al., 2011; Cusumano, 2004; Kittlaus & Clough, 2009; Xu & Brinkkemper, 2005). Hence for software companies the costs for producing one copy or millions of copies are about the same. The software business is considered as a high return business, having up to 99 percent gross profit margins for product sales. This appears to be a huge advantage and opportunity for software companies on the one hand. On the other hand, the software business also has its dark side, facing several challenges.

According to Cusumano (2003, 2004); Xu and Brinkkemper (2007), the developers of software can be rather considered as artists, since the development of software is highly unpredictable and risky. Almost 80 percent of software development projects run over time and over budget. No other industry could be characterized by a planning accuracy of about 20 percent, which is seen as best practice in the software business (Fricker, 2012). Furthermore, the markets of the software industry are shaped by a high degree of internationalization (Buxmann et al., 2011). This can be considered as an advantage or opportunity, as software companies can easily enter world wide market segements and deliver their solutions via the internet with low costs and effort. However, more a

challenge than an advantage is the fact that there is not only companies offering their solutions nationally, but there is world wide competition in the market. Thus, there are a multitude of challenges for software companies that are of an entrepreneurial nature. Software companies need to stay competitive in a highly unpredictable and internationalized market, thus they have to keep up with the pace of innovation and be aware of how they deal with product strategy, business models, their people and the management of software development (Cusumano, 2004).

The software business is unique, coming along with a variety of advantages, opportunities and challenges. In order to describe the software business in more detail, the following sections are focusing on several aspects. First, different types of software are distinguished. As introduced earlier (cf. chapter 1), this thesis is particularly interested in the business of product software, which is further explained in another section. Eventually, to highlight the managerial challenge of the product software business, product software management is introduced in a subsequent section.

3.1 Types of software

Before presenting different types of software products, the term software product is shortly introduced. A simple, but highly comprehensible definition of a software product is provided by Kittlaus and Clough (2009). They first define a product as follows:

> *"Product = Combination of (material and/or intangible) goods and services, which one party (called vendor) combines in support of their commercial interests, to transfer defined rights to a second party (called customer)."*

According to them, a software product is thus a *"[. . .] product whose primary component is software."* One thing to mention upfront is that with party, not necessarily a corporate entity is meant, but also individual persons, so multiple consumers can be of interest. After providing a basic definition of a software product, it is important to elaborate on the general classification of software products.

There are several opinions on how to classify software products. Buxmann et al. (2011) provide a classification, according to which software products can be differentiated in three ways.

The first one is also conform with the classification of the *Organisation for Economic Cooperation and Development* (OECD). According to the them, there are three categories of software: system infrastructure software, software development tools and application software (OECD, 2002). Xu and Brinkkemper (2005) define system infrastructure software as for example operating systems, middleware or system management software. Besides, software development tools consist of tools such as database management systems (DBMS) or development environments. Eventually, application software represents the category of software, which is investigated in this research. According to Xu and Brinkkemper (2005), application software are all programs designed to execute operations for a specific application such as enterprise resource planning systems (ERP),

cross-industry business software, CAD/CAM/CAE or other vertical industry business software.

Besides classifying software according to a specific category, there is a differentiation between enterprise and consumer software. Enterprise software is aimed at being offered to companies, which make use of the software to facilitate different tasks within the company itself. Cusumano (2004) is arguing that software companies, which offer enterprise software products, must deliver *rock-solid* products in terms of reliability as well as include *"[. . .] easy-to-use features for late adopters, good documentation, thorough technical support, and a fully array of complementary products and services."* This means that the customers of such software companies are also usually willing to pay for those kind of extra services such as expert technical support or trainings on the product. Generally those software companies follow the specific strategy of selling the software product itself through a license fee on the one hand as well as selling such services along with a multiyear maintenance contract on the other hand. The second category of software companies are those, who target millions of individual consumers, whereas the software product they sell is highly standardized for a specific purpose. A good example of such a company is Mircosoft, which are well-known for their operating system MS Windows or their office package MS Office, both software products, which are sold world wide.

Eventually, there is a third criteria for the classification of software products: the degree of standardization. On the one hand, there is customized software, also referred to as customer-tailored or tailor-made software (Buxmann et al., 2011; Hietala, Kontio, Jokinen, & Pyysiäinen, 2004; Xu & Brinkkemper, 2007). Customized software is manufactured in such a way that it pefectly fits the requirements of one specific customer, thus it is only sold once (Kittlaus & Clough, 2009; Xu & Brinkkemper, 2007). On the other hand, there is standardized software, which is developed for a specific market aimed at being sold many times. This type of software is also often referred to as the term product software (Xu & Brinkkemper, 2007).

Another classification scheme of software is presented by Hietala et al. (2004). These researchers roughly divide software into three categories: software products, customer-tailored software and embedded software (cf. figure 3.1).

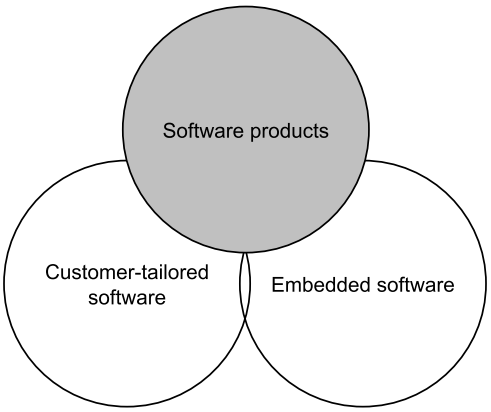


FIGURE 3.1: Classification of software according to Hietala et al. (2004).

Software products in their opinion are products that are traded on their own, thus not being dependent on any other kind of software or system. Contrary to that definition of software, embedded software is *"[...] built into other products, such as cellular phones, refrigerators, paper machines, ore television sets, and is not sold seperately"* (Hietala et al., 2004).

Another classification of software is provided by Xu and Brinkkemper (2007). The two authors differ software according to a matrix, considering both the number of copies of a software product as well as the specification of what is sold (cf. figure 3.2).

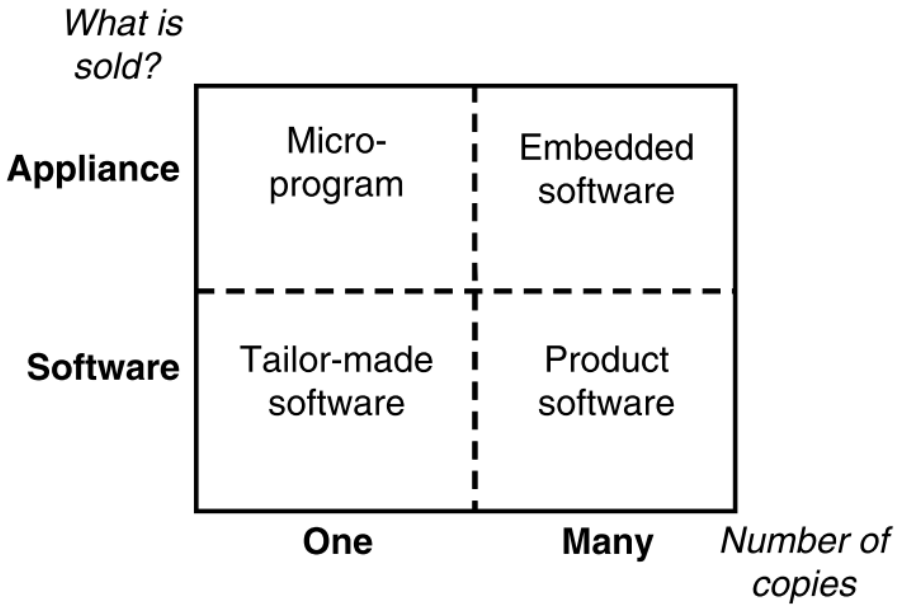


FIGURE 3.2: Classification of software according to Xu and Brinkkemper (2007).

A micro-program represents software that is embedded in just one single appliance such as a satellite. Embedded software on the contrary, as already explained previously, is embedded in multiple appliances such as TV's or mobile phones. Both lower quartiles are already known, yet given another term. Tailor-made software is also referred to as customized software, whereas product software means standardized software. Characteristics of both tailor-made software and product software are summarized in table 3.1.

Software	Typical characteristics
Tailor-made software	
Contractual tailor-made software	Software made for one particular buyer Busget and schedule fixed Penalties for late delivery
In-house tailor-made software	Used to improve efficiency/effectiveness of internal organization Limited number of end-users Possible conflicting interests between IT-department/end-user
Product software	
Business-to-business product software	Software sold to other business Many different buyers Critical to the buyer's business
Business-to-consumer product software	Software sold to individual buyers High volume buyers Market windows and buying seasons Failures can have fatal consequences

TABLE 3.1: Characteristics of tailor-made software and product software (Xu & Brinkkemper, 2007).

Summarizing this section, it was shown that software can be classified in manifold ways. Several researchers provide classification schemes, which enable the understanding of the different types of software. In the following section, the focus lies on product software, which was defined as a highly standardized product sold frequently to a specific market.

3.2 Product software

As illustrated in the previous section, amongst others there is the differentiation between tailor-made and product software. Although product software has not gained much attention by academia so far (Xu & Brinkkemper, 2007), it is yet a constantly growing business (Buxmann et al., 2011).

Inspired by the success of large software vendors such as Microsoft or SAP, more and more companies aim at switching from developing tailor-made software to developing product software (Artz, Weerd, Brinkkemper, & Fieggen, 2010). This process is also referred to as productization. An essential difference in development between the two types of software is that tailor-made software usually is delivered with one release, whereas product software development requires recurrent releases. Thus product software companies are confronted with the proper management of product software, which is known as software product management (cf. section 3.3). For software companies, which are willing to transform to

developing product software need to adapt their internal processes to the new business as well (Artz et al., 2010). Artz et al. (2010) provide a detailed illustration of the main differences in development of tailor-made software and product software (cf. figure 3.3).

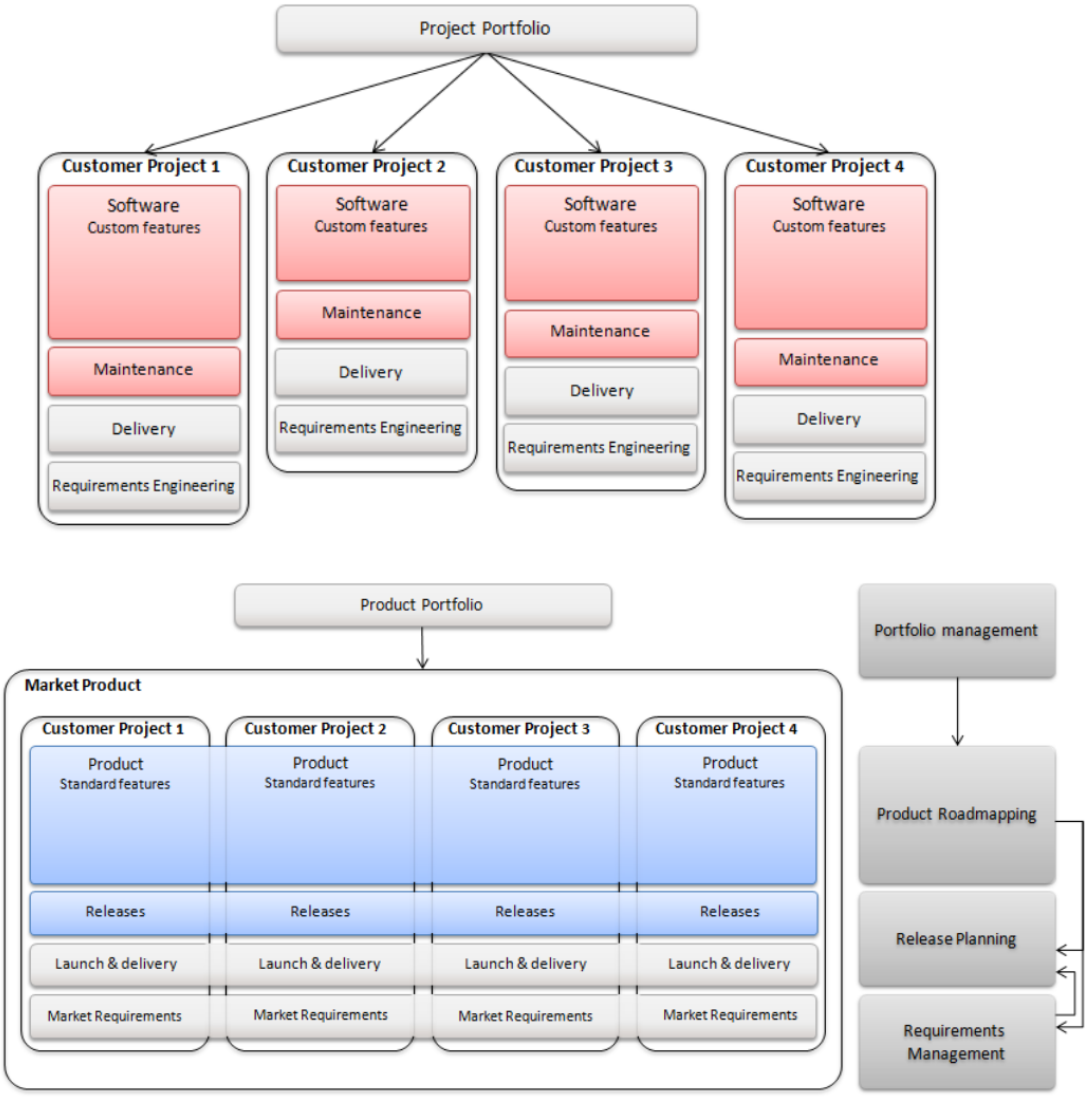


FIGURE 3.3: Comparison of the development of tailor-made as well as product software (based on Artz et al. (2010)).

While developing tailor-made software is much more focused on individual projects, which are differently performed with each customer, developing product software is much more generic, but involves several management aspects besides. The product is thus rather seen as a market product, which contains the same standard features for every customer or end-user. Besides standard features, also the requirements engineering part as well as launch and delivery activities are standardized for the whole customer base.

A definition of product software

The principle of "make one, sell many" is common to all kinds of product software. However, the differences between shrink-wrapped software, commercial off-the-shelf software (COTS), packaged and commercial software are still somewhat vague (Sawyer, 2000, 2001). Thus Xu and Brinkkemper (2007) established a framework containing an overview of the most relevant product software terms (cf. figure 3.4).

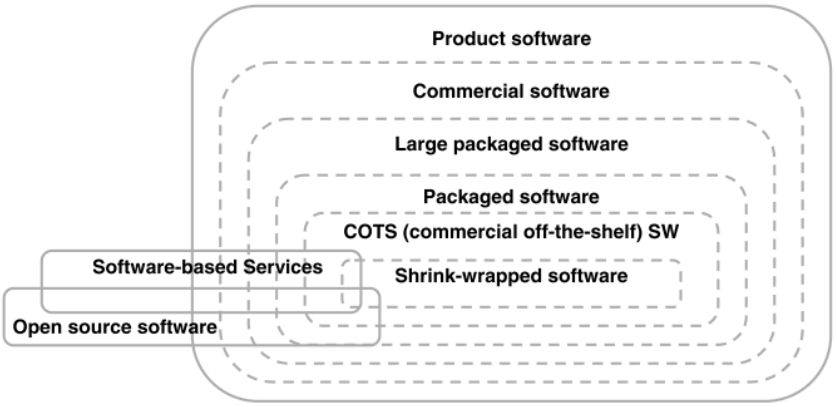


FIGURE 3.4: Relevant product software terms (Xu & Brinkkemper, 2007)).

Considering all the relevant terms that are related to product software, a common definition is provided by Xu and Brinkkemper (2005, 2007):

> *"A software product is defined as a packaged configuration of software components or a software-based service, with auxiliary materials, which is released for and traded in a specific market."*

This definition consists of four major components: 'packaged components', 'software-based services', 'auxiliary materials', as well as 'release and trading'. Packaged components refer to as the software that is delivered to the end-users, which means any kind of code, executables and web pages (Fricker, 2012; Xu & Brinkkemper, 2005, 2007). Software-based services covers any kind of software services that are commercialy sold to the customer as well. Both the packaged components and the software-based services form the actual software product. Along with packaged components and services, product software comes along with auxiliary materials in terms of documentation, training material, brochures and anything that is facilitating the use of the software product. Eventually, release and trading represent an essential part of successful product software. A software product thus is not only released to the market, but also requires the deployment of the customer system, training users, and related commercial activities.

Product software practices

Product software companies are generally characterized as companies that sell fully standardized product-packaged software to a specific market (Cusumano, 2004). Product software however is not only manufactured and sold along with a variety of services included, but there is also lots of further activities involved.

Xu and Brinkkemper (2007) illustrate the activities around product software from a development perspective. This so-called *product flow pipeline* represents the primary business processes that are necessary while producing product software (cf. figure 3.5).

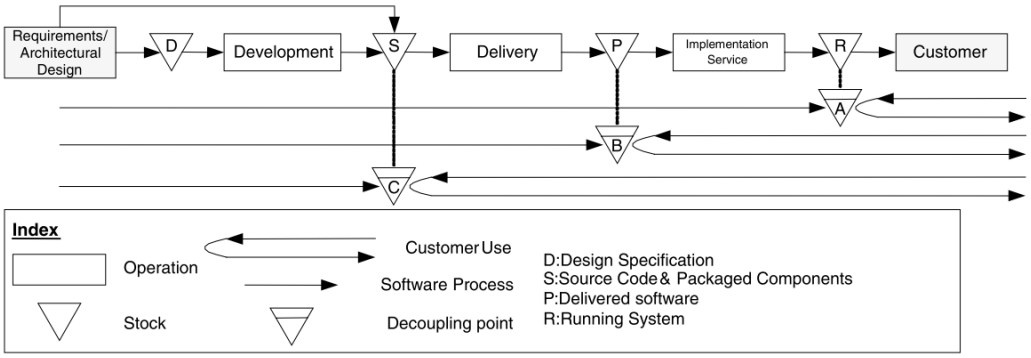


FIGURE 3.5: Product flow pipeline for product software and its decoupling points (Xu & Brinkkemper, 2007)).

As the previous figure reveals, there are several major processes involved until the customer can run the delivered software product. After the actual development process of a product, there are several decouplding points, which imply certain kind of business transaction between the software vendor and the customer. In some cases it is enough to simply sell packaged components to a customer, which is represented by decoupling point C. According to Xu and Brinkkemper (2007), open source and closed source packaged components belong to this category. Mostly those packaged components also need to be constantly developed, packed as a product and offered to that market, whose functional requirements they meet. Typically shrink-wrapped software belong to that category. A well known example of such product software is Microsoft Office, which usually does not require any kind of implementation or service for the customer as it is easy enough for them to install and update the product by themselves. Finally, product software like enterprise resource planning systems (ERP) or customer relationship management software (CRM) also requires proper implementation at the customer. Customers of such product software generally also have maintenance contracts, which guarantee that they receive expert support and several kinds of extra services by the software vendor.

For most product software companies selling their products is not only limited to sales activities, but also involves the deplyoment of the product at their customers. As this process appears to be 'inherently complex', Jansen and Brinkkemper (2006a) provide a model that focuses on such customer interaction processes. Their so-called CCU model (cf. figure 3.6) represents the process of customer configuration updating, which is defined

"[. . .] as the combination of the vendor side release process, the product or update delivery process, the customer side deployment process, and the activation process" (Jansen & Brinkkemper, 2006a).

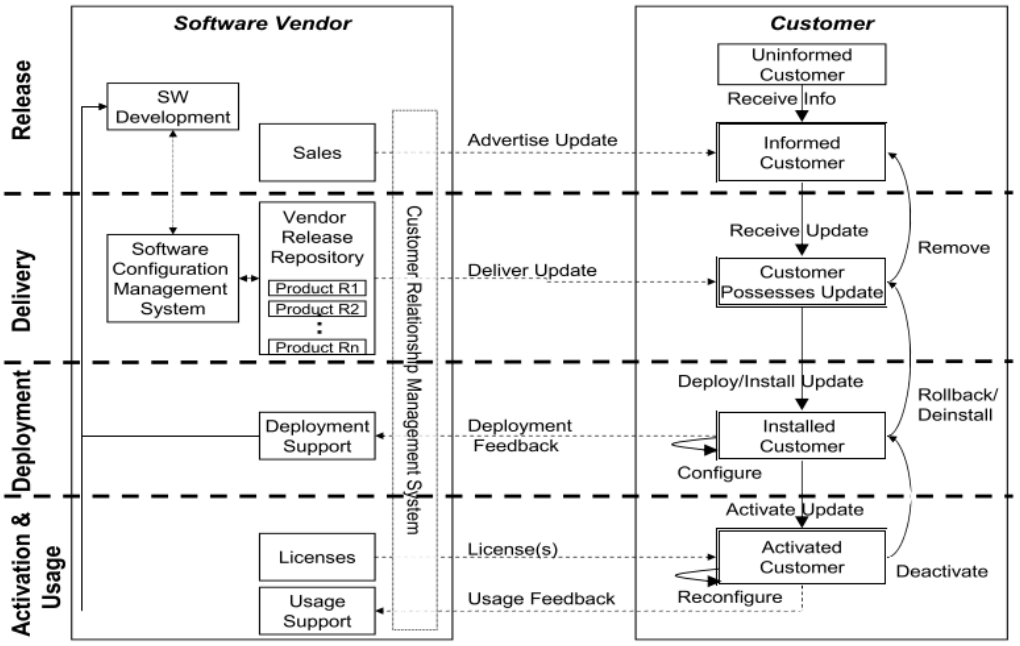


FIGURE 3.6: CCU Model (Jansen & Brinkkemper, 2006a).

Nowadays many product software companies encounter several problems when trying to improve customer configuration updating (Jansen & Brinkkemper, 2006a). The authors describe this process as highly complex because product software companies have to deal with variable features, different deployment environments and architectures, multiple revisions and much more. Furthermore, there is no tool support available that could assist such processes (Jansen, Ballintijn, & Brinkkemper, 2005).

Business models

Choosing the right business model is a critical decision for product software companies. However, the central question in this respect is, how a business model can be defined and what types of business models exist.

A generic definition of the business model concept is given by Osterwalder, Pigneur, and Tucci (2005):

> *"A business model is a conceptual tool containing a set of objects, concepts and their relationships with the objective to express the business logic of a specific firm. Therefore we must consider which concepts and relationships allow a simplified description and representation of what value is provided to customers, how this is done and with which financial consequences."*

The provided definition is rather broad, however the central message is clear and further developed by other researchers. As intended by Osterwalder et al. (2005), there are several concepts and factors that affect the business model of a firm. Rajala, Rossi, and Tuunainen (2003) established a framework that analyzes the business model in software firms and explains by what factors a business model is generally affected in the software market (cf. figure 3.7).

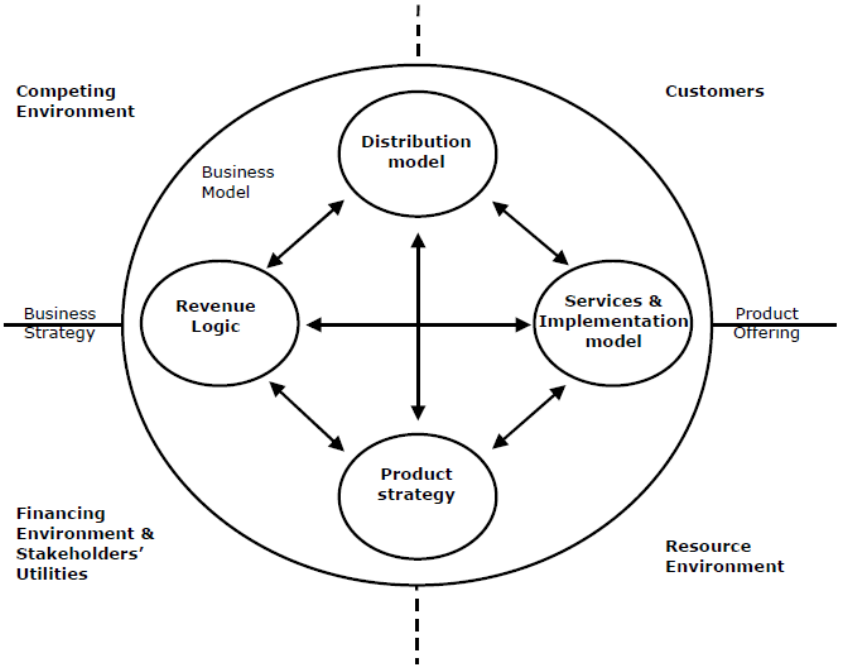


FIGURE 3.7: Elements of a business model (Rajala et al., 2003).

As the authors reveal, there are four major elements that affect the business model in software firms: Product strategy, revenue logic, distribution model, and service and implementation model. As software products are generally characterized by their intangible nature as well as their short product lifecycles, the business model of software firms also differs from those of other types of firms (Ojala & Tyrväinen, 2007). Product strategy refers to as the way how software companies organize product development. Thus, product software companies are confronted with the decision whether they produce customized or rather standardized software products. Besides, product software companies - as any kind of company - need to be aware of where their revenues are coming from. It is crucial for a product software company to know, which kinds of revenue sources exist. Furthermore, software products can be distributed in manifold ways. This element of a business model describes any kind of processes that are related to the organization of marketing, sales and service offering. Last, but not least important, there is the service and implementation model, which determines the way how a product is dispatched to customers and deployed as a *working solution* (Ojala & Tyrväinen, 2007). According to the authors, a working solution generally consists of activities such as the physical distribution, implementation and maintenance of the software product.

During the last decade researchers identified one business model, which is widely accepted and applied: the on-premises business model. When regarding the on-premises business model, there is the differentiation between product tailors as well as product licensors (Kontio, Jokinen, Mäkelä, & Leino, 2005). While product tailors are rather companies whose revenue is mainly based on licenses, charged for a product with a low degree of productization, product licensors produce highly standardized solutions that are offered along with several product management activities. The product licensor business model appears to be the most successful and dominant business model in the software industry (Popp, 2011). In its simplest form, the business model involves *"developing, selling, implementing, and managing the software"* (D'souza, Kabbeldijk, Seo, Jansen, & Brinkkemper, 2012).

To demonstrate the on-premises software business model and its components, D'souza et al. (2012) developed a high level representation of the model (cf. figure 3.8). As this representation depicts, there are several parties involved in a product software company's business processes. First there are external parties such as partners or software development tool suppliers, which support one core business activity of a product software company: development. As both software and knowledge are created by the Research & Development department of a software company, two further business units for sales and marketing as well as customer services appear to be necessary. Both business units are responsible for delivering solutions to the customer. The main sources of revenues for a product software company are thus the selling of licenses and service offerings. Selling licenses is a central challenge needs to be overcome by product software companies. Given that for software products almost the total product cost are fixed cost, reaching a certain minimum revenue or selling a minimum number of licenses is mandatory in order to reach the break-even point (Kittlaus & Clough, 2009).

The deployment of products based on the on-premises business model follow a simple rule: The software is shipped to the customer after it has been released. Afterwards it is installed at the customer, who has to maintain their own deployment, *"[. . .] including the provision of hardware, hosting, deployment, and configuration updating"* (D'souza et al., 2012). This also means that the customer physically owns the product to some extent and also controls the data.

Tremendously emerging and changing technology also opened new opportunities for software companies to offer their products differently. This has the consequence that product software companies now are confronted with a new kind of a business model: The Software-as-a-Service model (SaaS). SaaS nowadays is a trend, which is listed on the agenda of many CEOs in the IT industry. A term that is often used in this respect is cloud computing. Several researchers are interested in investigating the phenomenon of deploying software products in the cloud environment (Andrikopoulos et al., 2013; Cusumano, 2010; Dubey & Wagle, 2007; Jamshidi, Ahmad, & Pahl, 2013).

Contrary to selling licences, software companies with the SaaS business model offer software components and services, which are deployed at a service provider. The customers of such SaaS solutions are thus rather paying for renting a product, which are often offered via the internet as web services (Buxmann et al., 2011). In most cases, customers,

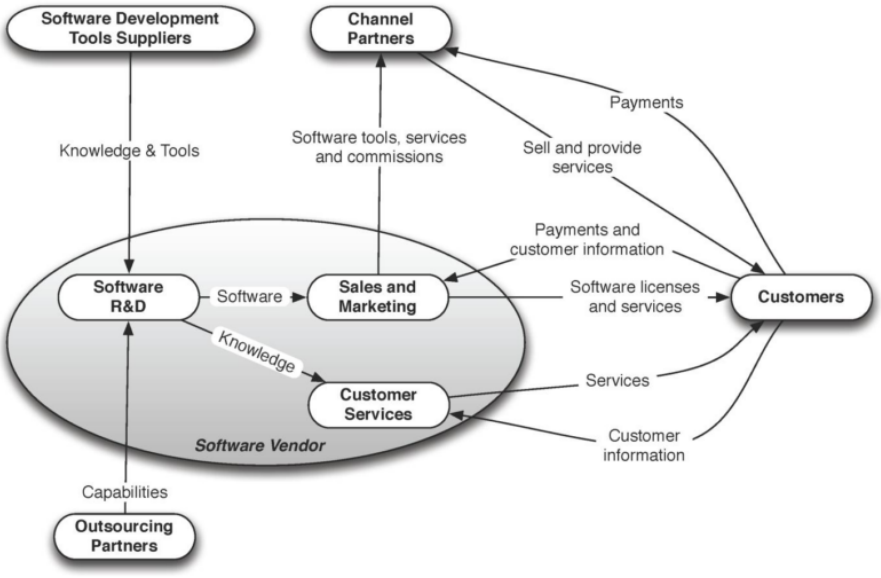


FIGURE 3.8: High level representation of on-premises software business model (D'souza et al., 2012).

who choose such solutions, are paying fees for a monthly subscription. Obviously, a new business model involves changes in the different elements of the business model.

D'souza et al. (2012) elaborate these kind of changes from different perspectives. First, there are practically changes in two domains: the business perspective and the technological perspective. Latter is specifically changing in terms of the deployment method. A simple visualization of the difference between the on-premises deployment method and the SaaS deployment method is shown in figure 3.9.

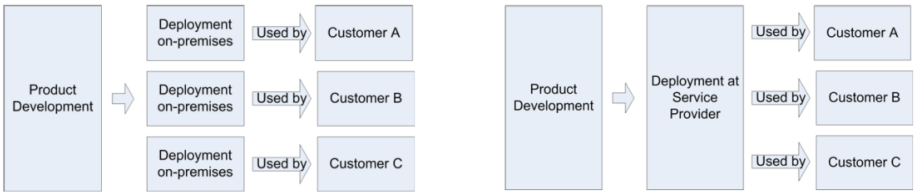


FIGURE 3.9: On-premises versus Software-as-a-Service Deployment Method (D'souza et al., 2012).

Software companies offer their product as a service through the internet instead of deploying it at the customer's physical location. Instead, the product is deployed at a service provider, which is either located in-house or at an external partner specialized in product hosting. For the classical product software company, there is the advantage of significant cost savings as the deployment and installation costs are absorbed by the SaaS provider. With this solution, product software companies have yet another advantage of serving multiple customers through one shared instance of a software product. This phenomenon is referred to as *multi-tenancy*, where both the database and several packaged software

components can be used simultaneously by multiple tenants (Bezemer & Zaidman, 2010; Guo, Sun, Huang, Wang, & Gao, 2007). Implying several excellent advantages such as lower overall costs as well as easier and cheaper application maintenance, security is a large issue when it comes to deploying a software product in the 'cloud'. In a multi-tenant environment, *"[...] a security breach can result in the exposure of data to other, possibly competitive, tenants"* (Bezemer & Zaidman, 2010). Thus, data protection is an essential issue for software companies that consider the introduction of SaaS-solutions (Guo et al., 2007). Another change, related to the technological perspective, is that the technical architecture of a product needs to be changed in such a way that it can be be used by multiple tenants at the same time. This requires changes in the database level as well as at the process level, where different tenants use different functionality (Bezemer & Zaidman, 2010).

The new emerging SaaS business model involves a lot of changes in the business perspective as well as the technological perspective. D'souza et al. (2012) provide a transition model, which summarizes the elements that are changing when moving to the 'cloud' considering the two aforementioned perspectives (cf. figure 3.10).

		On-premises	**SaaS**
Business/ Product Structure	**B**	Main costs related to: • Research and development • Sales and marketing • Customer support	In addition to costs mentioned under the on-premises model, there are hosting costs.
	T	After releasing, product is shipped to the customer	• Active tenant management • Tenant-aware database • Extensive logging system
Revenue Logic	**B**	Sales of licences	Sales of services
	T	Pay per user	• Pay per use • Pay per feature
Customer Relationship	**B**	Governed by contracts that mainly deal with warrants, customer supports, and upgrades	Shifts towards outsourcing type of contract where additional elements such as service level agreements and security are covered
	T	Customer hosts and owns all data	Software vendor is responsible for data security and safety
Partnerships	**B**	Main focus is on channel partners	Ecosystem approach
	T	Third party connection on premises	• Third party service connection is taken care of by software vendor • Easy connection for third parties through application program interfaces

FIGURE 3.10: On-Premises to SaaS Transition Model (B: business perspective; and T: technological perspective) (D'souza et al., 2012).

3.3 Software product management

As portrayed so far, product software companies are not only concerned with building and selling software. Delivering product software entails a broad set of other challenges, ranging from eliciting market requirements to delivering, maintaining and updating subsequent releases of a product. In order to strategically align a software product in accordance with a company's vision, it is necessary to sustain that process *"[...] with an interrelated set of competences, practices, and processes for planning, building, marketing, distributing, evoloving, and maintaining a software product"* (Fricker, 2012). This kind of management of a software product is referred to as software product management (SPM) (Ebert, 2007, 2014; Fricker, 2012; Gorchels, 2006; Haines, 2009; Kittlaus & Clough, 2009). Fricker (2012) defines SPM as follows:

> *"Software product management is the discipline, which governs a software product from its inception to its close-down to generate as large value as possible for the business."*

In short, SPM's primary objective is to *"[...] achieve sustainable success over the life cycle of the software product"* (Kittlaus & Clough, 2009). Governing a software product and managing it successfully requires a product software company to have someone, who is responsible for SPM. This person is referred to as the software product manager. The software product manager is thus responsible for sustaining the success of a software product, which is why he is often called the *mini-CEO* of a product software company (Ebert, 2007, 2014). His central role besides sustaining the success of a software product is to plan the scope and evolution of the software product and to align it with the user, market, and the company needs (Gorchels, 2006). Although it might sound simple to produce a software product and deliver it to the market, it is a highly complex undertaking, in which many internal and external stakeholders are involved (van de Weerd, Brinkkemper, Nieuwenhuis, Versendaal, & Bijlsma, 2006).

Due to the aforementioned facts, methods and frameworks to facilitate SPM are beneficial. Xu and Brinkkemper (2005) therefore developed the software product development framework, which views SPM from three perspectives, being the societal perspective, the company perspective and the development perspective (cf. figure 3.11).

The framework clearly illustrates which different factors influence SPM. The societal perspective in this case represents any kind of external factor that affects the SPM practices of a product software company, being factors such as laws and regulations, resource provisioning as well as the economy. From a company perspective, all kinds of activities that are not related to the actual development of the product are essential. This might be activities such as marketing, sales or the management of the product lifecycle. Eventually the development perspective concerns everything that is related to the development of the product. As already mentioned in earlier sections, the development not only consists of the development of the product, but also of activities before and after development. First, a product needs to be designed and requirements need to be detected, which fulfill market needs. This is referred to as the requirements architecture. After having developed and tested the software product, it needs to be first delivered (*delivery*) and then deployed at the customer (*deployment*).

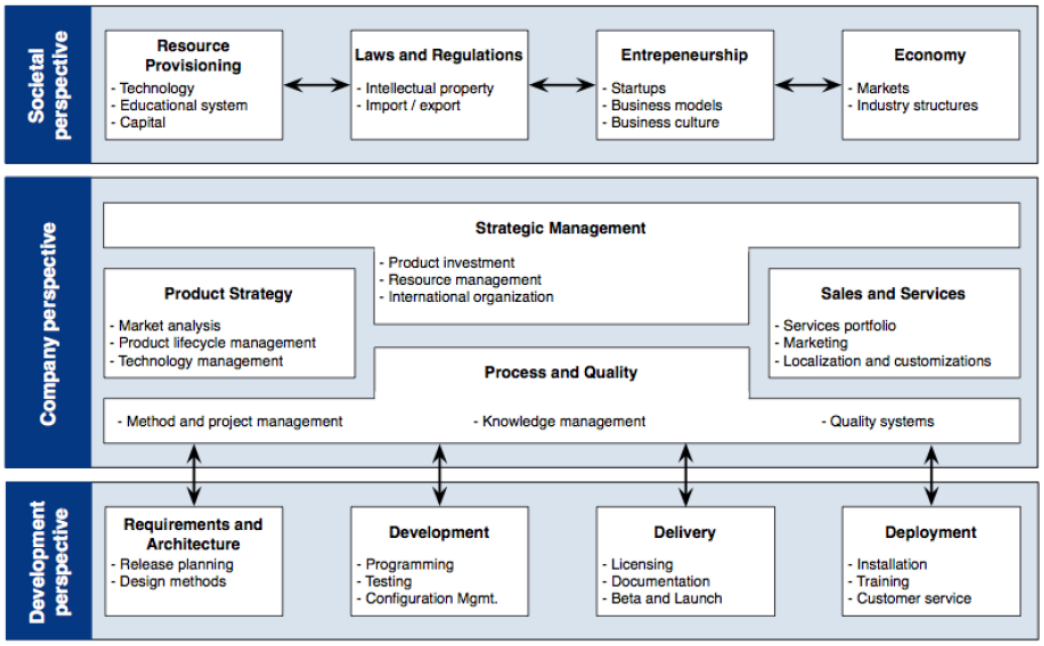


FIGURE 3.11: Software product development framework (Xu & Brinkkemper, 2007)).

As this thesis has the main objective to explore the overhaul process of product software from a rather organizational perspective, the focus is mainly set on the company perspective of the product software research framework. In this context, there are two aspects that are relevant for the specific research objective of this thesis: first , the product lifecycle management related to the product strategy of a company; second, the strategic decisions that product software companies make during a product overhaul. However, also the societal perspective and the development perspective are considered important aspects that need to be investigated.

Yet another framework that elaborates SPM practices in more depth is the reference framework for SPM, which has been established by van de Weerd et al. (2006) and further developed by Bekkers, Weerd, Spruit, and Brinkkemper (2010). The framework is structured in such a way that it represents the objects and artifacts of SPM as well as the stakeholders that are involved in the process (cf. figure 3.12).

The framework is divided in to four focus areas, being *Requirements management*, *Release planning*, *Product roadmapping*, and *Portfolio management*. Besides, the framework shows the external and internal stakeholders of a product software company. As external stakeholders, the following three are identified: market, customers, and partners. The internal stakeholders are represented by the company board, sales, marketing, research & development (R&D), development, support, and services. It is becoming clear that SPM is surrounded by many stakeholders, who influence all the processes.

Requirements management

Requirements management represents one of the key areas in product software companies (Carlshamre & Regnell, 2000). It consists of the following activities: requirements

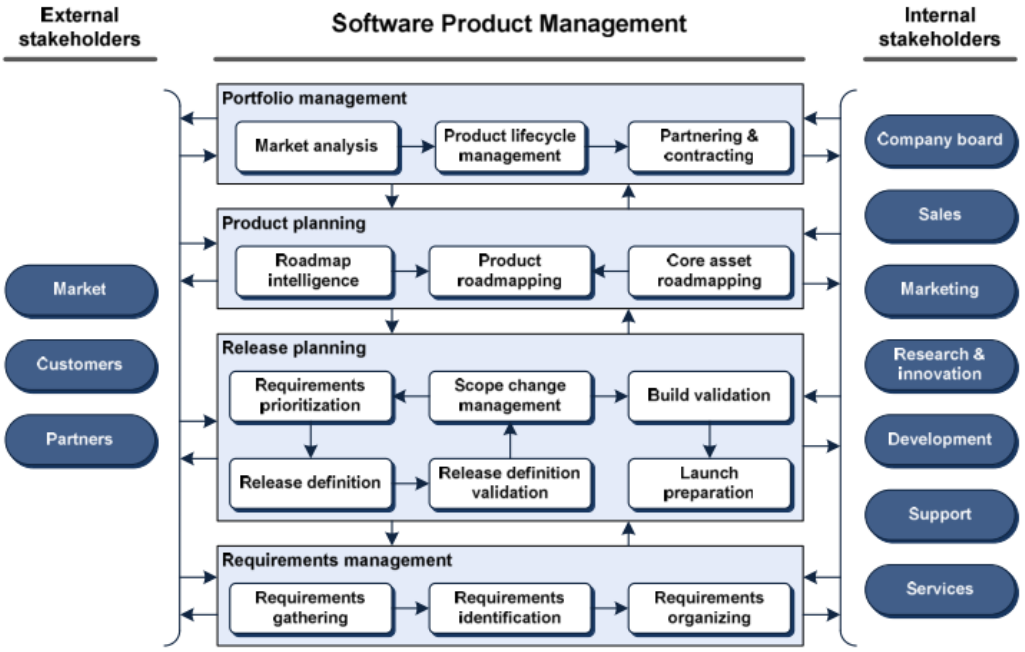


FIGURE 3.12: The Software Product Management Competence Model (Bekkers et al., 2010).

gathering, requirements identification, and requirements organization. These activities are challenging for product software companies as internal and external stakeholders are constantly delivering new requirements to the company, which need to be managed in an appropriate way. This process can cost a lot of time, which is the reason why several methods such as linguistic engineering have been proposed to facilitate these processes (Natt och Dag, Regnell, Gervasi, & Brinkkemper, 2005).

Release planning

Release planning is the second key process area in SPM. In this process, the set of requirements for the next release is determined, which is then delivered to the customer. Essential elements of this process are requirements prioritization and selection, release definition and validation, launch preparation as well as scope change management. A highly challenging activity in release planning is the prioritization of the requirements, which will then be selected to be included in the next release of the product (Karlsson, 1996; Karlsson & Ryan, 1997; Wiegers, 1999). According to Berander and Andrews (2005), requirements prioritization is a fundamental activity for the overall project success of a product software company. The main benefit of accurate requirements prioritization is summarized by Ruhe, Eberlein, and Pfahl (2002):

> *"The challenge is to select the right requirements out of a given superset of candidate requirements so that all the different key interests, technical constraints and preferences of the critical stakeholders are fulfilled and the overall business value of the product is maximized."*

To successfully prioritize the organized requirements, methods such as Wiegers method (Wiegers, 1999), bubblesort (Karlsson, Wohlin, & Regnell, 1998), the analytic hierarchy process (AHP) (Berander & Andrews, 2005; Karlsson & Ryan, 1997) have been proposed by researchers. Besides prioritizing and selecting requirements, it is crucial to write a release definition, which is validated by different stakeholders of the company. Once accepted, a launch preparation package is constructed and the release is launched to the customers.

A typical requirements state model is presented by Regnell and Brinkkemper (2005). The state model differs between two modes, which is the continuous mode and the release mode (cf. figure 3.13). In the continuous mode, the requirements are received and registered by the software product manager. It is decided, whether to accept or discard a requirement for the next release. Once approved and specified, the requirements are transfered to the release mode.

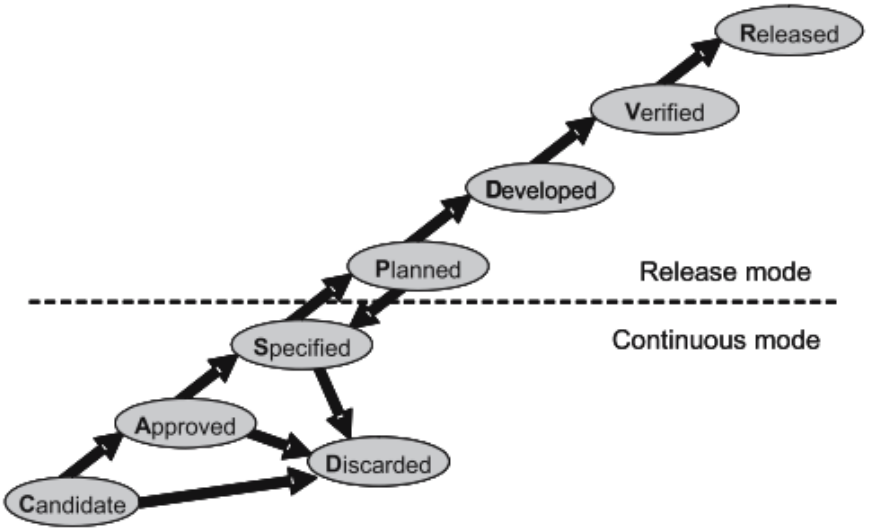


FIGURE 3.13: Requirements state model, or requirements salmon ladder (Regnell & Brinkkemper, 2005).

Product planning/Product roadmapping

Core activities of product planning are as follows: roadmap intelligence, product roadmapping, and core asset roadmapping (Bekkers et al., 2010). Roadmapping is called *"[. . .] a metaphor for planning and protraying the use of scientific and technological resources, elements and their structural relationships over a period of time"* (Vähäniitty, Lassenius, & Rautiainen, 2002). This means that information is obtained for the creation of a roadmap for a product or product line and its core assets (Bekkers et al., 2010).

Portfolio management

Eventually, portfolio management is the fourth key process area of the reference framework, which has the closest link to the theme of this thesis. As van de Weerd et al.

(2006) emphasize, portfolio management *"[. . .] entails the decision making about the set of existing products, introducing new products by looking at market trends and the product development strategy, making decisions about the product lifecycle, and establishing partnerships and contracts."*

In general, a portfolio is a collection of assets and investments of an organization (Krebs, 2008). When regarding their product portfolio, a product software company is concerned with a number of questions. It is essential for them to clarify, whether their product is still strategically beneficial, in which product they should invest and which product should be stopped from being delivered to the market. Thus, product portfolio management also plays an important role when it comes to a product overhaul process. By means of having proper portfolio management practices, it can be determined whether the overhaul of a product is reasonable or not in terms of the financial situation of the company and its overall strategic alignment. In Pohl, Böckle, and Van der Linden (2005), portfolio management is defined as the strategic decision process, whereby a business portfolio is constantly updated and revised in order to meet business objectives. Haines (2009) provide a further definition when referring to product portfolio management:

> *"Good portfolio management requires close oversight, constant review of historical and current performance, and the courage to rebalance and rationalize the portfolio when necessary while aligning your actions with the overall strategy of the organization."*

There are several methods and models that foster portfolio management in product software companies. One type of models is portfolio assessment models, which help you to make finanical decisions concerning your existing products. One of these models is the *Boston Consulting Group Growth Share Model* (cf. figure 3.14).

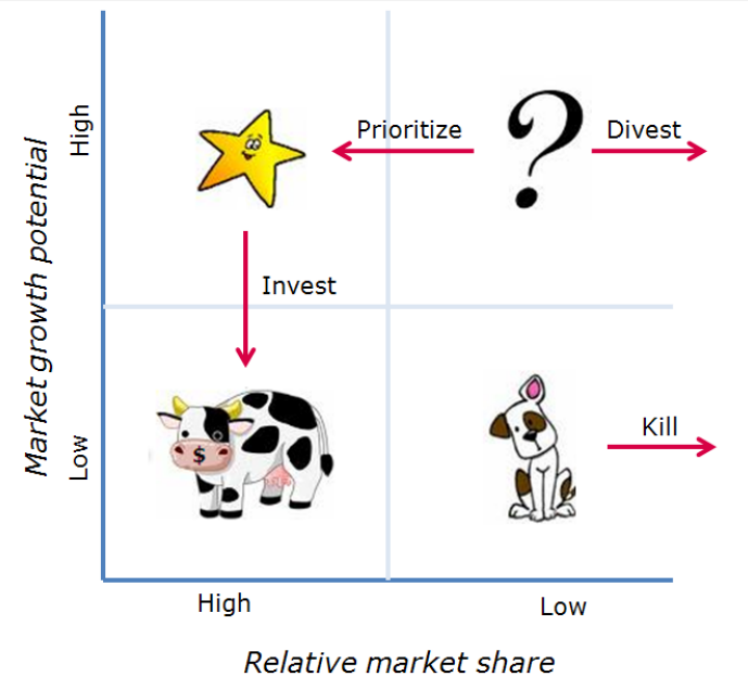


FIGURE 3.14: Growth Share Model (Henderson, 1973).

The model, established by the Boston Consulting Group in the early 70's, is divided in four quadrants, which represent the following four categories:

- Cash cows: products, which have a high market share in a slow-growing industry. Furthermore, they shall generate funding for other products or new product development.
- Dogs: products, which have a low market share in a mature, slow-growing industry. Thus, dogs are likely to be phased-out at some point.
- Stars: products, which have a high market share in a fast-growing market. Companies need to invest in stars to position them in the market, possibly leading to a change to a cash cow.
- Question marks: products, which grow rapidly and consume large amounts of investments. Question marks are not very profitable yet, however they could change to stars as well. Further analysis is required to make decisions about investements or divestments in the product.

3.4 Software product lifecycle management

Another central activity of product software companies in SPM is to deal with the lifecycle management of the product. Product lifecycle management is focused on the government of a software product's life, from initial conception to its termination. Haines (2009) presents the lifecycle of a software product from a product manager's point of view, also considering the economical lifecycle perspective of a product (cf. figure 3.15).

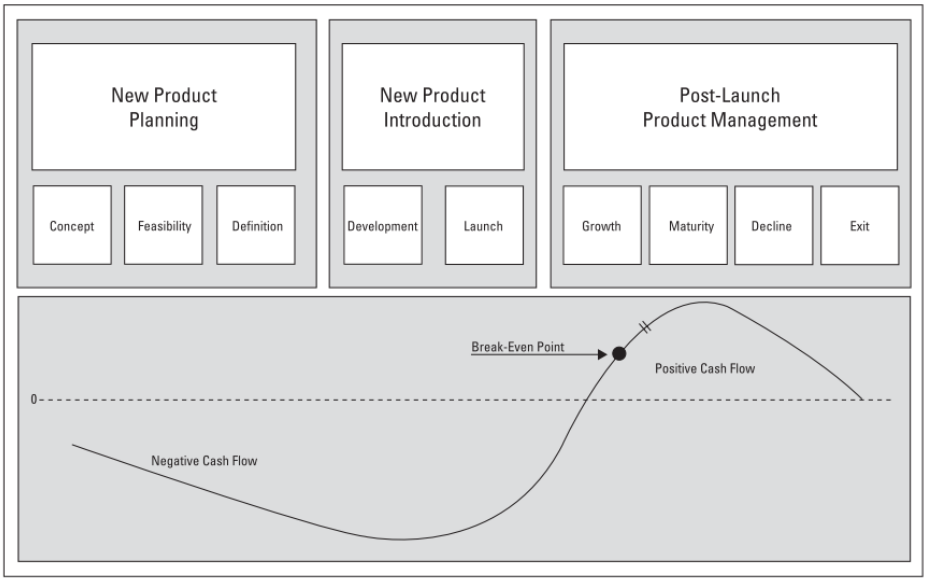


FIGURE 3.15: Portfolio management life cycle model (Haines, 2009).

The model consists of three major parts: *New product planning*, *New product introduction*, and *Post-launch management*. In the first phase, ideas are designed and defined. It is about the concept, feasibility and definition of a product. As figure 3.15 illustrates, companies do not create any revenue in the product planning phase. Investments are mainly made for new *product projects*, with which the company hopes to achieve market success.

In the second stage, a new product is developed and launched to a specific market. Generally, a product software company still invests a lot of financial and human resources to make the product ready for the market. This means that products in the new product introduction phase still do not contribute any money to the company.

Eventually, software products reach the phase, in which they have the potential to grow rapidly, thus contributing to the company's revenue and business success. This phase is referred to as Post-Launch Product Management. Product software companies' main objective is to break even as quickly as possible to finance further operations and pay salaraies. This is not only inevitable for startup companies, but also mature product software companies that are successfully operating in the market already.

A plethora of software products reach a level of maturity, where they generate a positive cash-flow thus highly contributing to a company's revenue. However, every software product also reaches its end-of-life at a certain point in time. This can have manifold reasons such as decreasing market demands, too expensive maintenance costs or a competitive product by another company. Ending the life of a software product can an immense challenge for product software companies (Jansen et al., 2011). Phasing out the product concerns every stakeholder that is involved in software product management. In most cases, product software companies need to introduce a follow-up product, which implies further challenges to those companies.

Chapter 4

The Overhaul Process of Product Software

This chapter represents the second part of the theoretical framework of this thesis. The previous chapter depicted how the software business in general can be characterized and what kind of software products exist. Furthermore, the management process of software has been thoroughly examined. This introduction was necessary to understand the process that is explained in this chapter - the overhaul process of product software.

Product software requires appropriate management as it has a constantly evolving nature, which is illustrated in the first section of this chapter. This is the basis for the subsequent sections, which are introducing and discussing the product overhaul process in software companies. First, a definition as well as an explanation of the associated concepts is provided. Moreover, the reasons and triggers of the overhaul process are demonstrated. Ultimately, some strategies and methods that are related to the overhaul process are discussed.

4.1 Software maintenance and evolution

According to the IEEE Standard 1219 (IEEE, 1998), software maintenance is defined as follows:

> *"The modification of a software product after delivery to correct faults, to improve performance or other attributes, or to adapt the product to a modified environment."*

This means that software enters a maintenance stage as soon as it has been delivered to the market. Generally, maintenance activities are categorized into four classes (K. H. Bennett & Rajlich, 2000):

- Adaptive: modification of a software product after delivery, often entailing changes in the software environment.

- Perfective: modification of a software product after delivery to improve performance and maintainability.
- Corrective: modification of a software product after delivery to correct faults in the software.
- Preventive: modification of a software product to prevent problems in the future.

The core problem of software maintenance is that new requirements are continuously evolving, which need to be incorporated in the product. As shown in chapter 3, a key process area of software product management is release planning, which is concerned with planning and introducing new releases of a software product. This area is also an important part of successful software maintenance. Software maintenance can be seen as a part of the primary lifecycle processes of software. April, Hayes, Abran, and Dumke (2005) illustrate, where the maintenance process is integrated in the lifecycle processes of a software company and which activities it consists of (cf. figure 4.1).

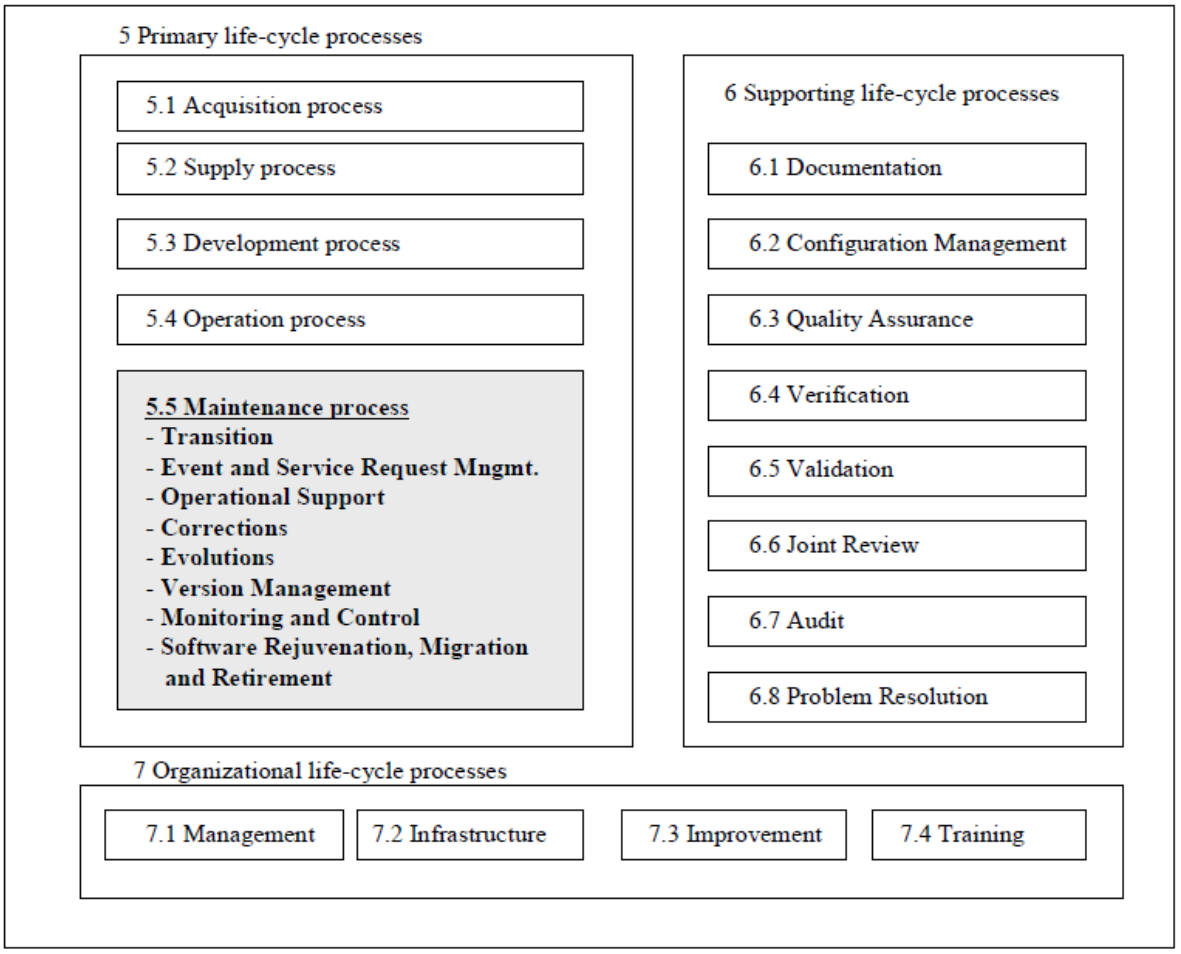


FIGURE 4.1: ISO/IEC 12207 software maintenance process model (in April et al. (2005)).

One activity that is mentioned in the model is evolution. Lehman and Ramil (2003) describes evolution as follows:

> *"[. . .] evolution describes a class of phenomena observable in many different domains [. . .], it can involve entities [. . .] such as natural species, [. . .] concepts, theories, ideas. If any of these undergo continual progressive change [. . .] they are said to evolve over time."*

The terms evolution and maintenance are frequently used for the same activity. However, evolution is the evolution of the software in terms of the overall functionality, whereas software maintenance rather has reduced objectives such as performing corrective or adaptive changes in the software (Rajlich, 2014). A model that helps with planning software evolution in a company is provided by Rajlich (2014) and K. H. Bennett and Rajlich (2000). Their so-called *versioned staged model* illustrates the software lifecycle as a sequence of stages, ranging from initial development to close-down, whereas the maintenance phase is split into three phases: evolution, servicing and phase-out (cf. figure 4.2).

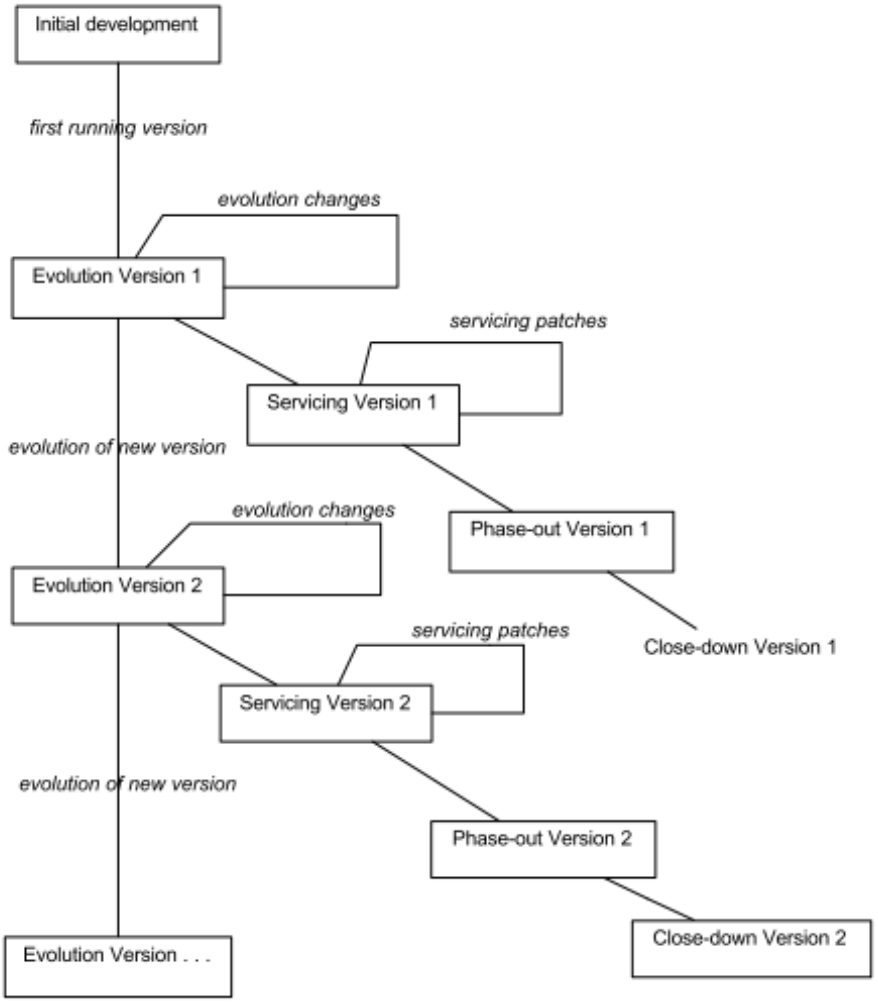


FIGURE 4.2: The version staged model (K. Bennett et al., 2002; K. H. Bennett & Rajlich, 2000).

When initial software development is successful, evolution takes place, which has the main objective of adapting the software product to the ever-changing requirements and operating environment. This is done as long as the software is successfully offered to the market. Otherwise, evolution turns into servicing, which means that only small tactical changes such as code changes are possible (K. H. Bennett & Rajlich, 2000). Final stages are phase-out and close-down. During phase-out, no more servicing is done, however the product might be still manufactured. During close-down, the software is finally disconnected and the users are directed towards a follow-up product. During the evolution of software, a plethora of changes take place. Kittlaus and Clough (2009) present the three-level hierarchy of IBM as widely accepted categorization of software change levels:

- Version: new product, which comprises crucial expansions and improvements as well as requires a new fee and product number.
- Release: new level of software with bigger functional or other improvements, which does not require change in the product number and which is free of charge.
- Modification level: new status of software with limited expansions (mainly bug fixing).

Jansen and Brinkkemper (2006b) furthermore differentiate between several release packages:

- Update package: package that promotes a customers configuration to a newer configuration.
- Bug fix update package: package that only contains bug fixes.
- Feature update package: package that only contains new features.
- Minor update package: package that contains both bug fixes and new features.
- Major update package: package that contains not only both bug fixes and new features, but also changes structural parts of a product such as the architecture or the data model.

During software evolution, a software product undergoes various changes. The following section elucidated how the overhaul process is incorporated in this context.

4.2 The software product overhaul process

In academic literature, the term overhaul has not yet been associated with the business of software. In the Oxford Dictionaries [1], the noun *overhaul* is defined as follows:

> *"A thorough examination of machinery or a system, with repairs or changes made if necessary."*

As an example, the *major overhaul of environmental policies* is named. Considering this definition, an overhaul is something that usually leads to changes of an entity. This entity can be a system , too, such as a software system or software product. So how does it come that the term overhaul has not been adapted by the software industry so far?

As stated in the previous section, software undergoes steady evolution that leads to manifold changes. However, evolution does not only mean the constant improvement of a software product in terms of changes. In fact, at some point in time, software products move into a sunset phase, in which they are replaced by a new product (Kittlaus & Clough, 2009). Sunsetting a software product is often used as a synonym for the phase-out or end-of-life of a software product (Jansen et al., 2011). When software products reach the sunsetting phase, which means that they need to be replaced by a follow-up product, the term migration is often applied. A basic definition of migration is provided as follows:

> *"In information technology, migration is the process of moving from the use of one operating environment to another operating environment that is, in most cases, thought to be a better one. [...] moving from Windows NT Server to Windows 2000 Server would usually considered a migration [...]. Migration could also mean moving from WIndows NT to a UNIX-based operating system, [...] can involve moving to new hardware, new software, or both. [...] One can migrate data from one kind of database to another kind of database" (Rouse, 2005).*

Another definition of migration is provided by the IEEE Standard 1219 on Software Maintenance (IEEE, 1998), which classifies migration as a kind of adaptive maintenance that is a *"modification of a software product performed after delivery to keep a computer program usable on a changed or changing environment."* As the previous definitions of migration clearly express, a migration to new software generally implies many changes and alterations on different levels. These changes can range from changing hardware platforms, changing operating systems and changing the architecture to simply changing the client interface of a software product (Gimnich & Winter, 2005; Torchiano et al., 2008). As Kittlaus and Clough (2009) describe, a software product is depended on several external entities, which can be influenced by a migration of the software product as well:

- Operating system (e.g. Windows, Linux, etc.).
- Database management system (MySQL, Oracle, etc.).

[1] Link: `http://www.oxforddictionaries.com/definition/english/overhaul`

- Middleware (messaging, queuing system, etc.).
- Other application systems that the product has interfaces with.

According to Torchiano et al. (2008), who conducted a survey about software migration projects in the Italian industry, 70 percent of the surveyed software companies have already undergone a software migration. Besides the 70 percent, further 15 percent are currently planning a migration project, which makes only 15 perecent in total that have never experienced any migration yet. This emphasizes the importance of the phenomenon for the software industry. Furthermore, thestudy revealed that there is a trend of replacing the database management system (DBMS), the programming language as well as the architecture, which is often a change from outdated architecture to a client/server architecture or web applications. In most cases, so-called legacy systems or legacy languages (e.g. COBOL) are replaced by state-of-the-art solutions.

Besides all these changes from a technological point of view, the migrated software product should also contain new functions that bring real business value to the customers (Kittlaus & Clough, 2009). This is essential for product software companies in particular as they depend on their customers. As illustrated in the previous chapter (cf. chapter 3), the product software business is about selling product licenses or in the new era, selling subscriptions to software offered as a service via the internet (SaaS). Thus, the more licenses or subscriptions product software companies sell, and the more customers they can attract, the more revenue they will make. If the software changes and is migrated, the customers also need to benefit from the new functionalities in order to be satisfied.

Another issue that comes along with the migration to a new software product are the changes that a product software company itself undergoes in terms of the product strategy. Staudenmayer et al. (1998) demonstrate in a field study how legacy software organizations cope with volatility and change. Some of their major findings are that software companies often change or extend the target market and thus the customer base of their product. In case a company is doing so - which might be the switch from a national orientation to selling software internationally or even worldwide - the organizational structure and composition is usually changing as well. Such a change is for instance represented by the reorganization of the development team. As technology and market orientation changes, also software developers need to adapt to a new technological paradigm. Quite often, software companies thus have to hire people, who are aware of this new technological paradigm.

As previously shown, migration defines the switch of a software product to a new technological environment, leading to the close-down of a product and its replacement by a new product. So why should the term overhaul not simply be replaced by the term migration as it is widely applied and accepted by researchers and practitioners in the field of software?

Software migration indeed is the transition of a software product to a new technological environment. However, there are types of migrations and strategies to approach them (cf. section 4.4). As by definition software is not only transfered to a new technological environment by just migrating its packaged components. There are also two further ways

of how to develop a new version of a product. According to Bisbal, Lawless, and Grimson (1999); Brodie and Stonebraker (1995); Gimnich and Winter (2005), there is not only migration. Generally, there are the following possibilities to apply changes to a software product:

- Migration: direct transition of the functioning product to the new technological environment, while retaining the original data and functionality of a product.
- Wrapping: provision of a new interface to a software component to make it more easily accessible by other software components.
- Redevelopment: redvelopment of the software product in the new environment from scratch.

Despite the fact that *migration* is not only the migration of a software product, it is still commonly used by researchers and practitioners when talking about the phenomenon. Besides, when mentioning migration, the migration of legacy systems is often described (K. H. Bennett, 1995; Bisbal et al., 1999; Brodie & Stonebraker, 1995). Legacy systems in this respect refer to as large computer systems that are operated by any kind of company. In contrast, the overhaul of product software specifically aims at explaining the process of 'migrating' product software, which is - as previously explained (cf. section 3.2) - highly standardized software, offered to a specific market. In order to differentiate these two phenomena, the term overhaul is introduced.

4.3 The triggers of the overhaul process

In the previous section, the overhaul process of product software was defined. This section aims at demonstrating the reasons and triggers of such an overhaul.

To mention first, products that are undertaken an overhaul are often reaching the sunsetting stage before they are replaced by a new version of the product. There are different opinions on the triggers for sunsetting a software product. Haines (2009) refers to the sunsetting of a software product as its discontinuation. According to him, there are a number of reasons why product software companies make the discontinuation decision:

- The product is no longer strategically viable or valuable to the firm.
- Sales volumes and revenue are declining rapidly.
- Market share of the product is falling precipitously.
- Customers have been encouraged to switch to another competitive product.
- Production or maintenance costs are escalating and the product is losing money.
- Team members are disinterested and unmotivated by a sinking product.

- Salespeople are not willing to sell the product anymore.

Haines (2009) rather sees the reasons for discontinuation as being business-related. Jansen et al. (2011) present a more thorough view on the triggers of discontinuation. The reasons for discontinuation, which they refer to as sunsetting, in their opinion is found in three different categories:

- Product strategy
- Platform changes
- Portfolio decisions

This categorization of the reasons is also mentioned in Jansen, Brinkkemper, and Finkelstein (2009) and in van de Weerd, Bekkers, and Brinkkemper (2010). Product strategy consists of reasons such as deprecation and lack of demand for the product. Further reasons in product strategy are more technological reasons such as the maintenance of the product becomes too expensive or developers for a specific technological platform become scarce, i.e. there are no more people, who are aware of the specific programming language the product is written in. Jansen et al. (2011) deliver the central message in the product strategy as follows:

> *"If a product is successful presently, but will be hard to monetize in a couple of years because the product is no longer needed or based on technology that will become outdated soon, it can become a candidate for discontinuation."*

The second category is refered to as platform changes. As already described, a software product is depended on its platform and technological environment. This means that, if the underlying technology of the product changes or becomes obsolete, the software product manager has to decide whether the product can be still maintained properly in the changed environment. If a component such as the database management system is phased out, the software product needs to be based on a new system as well.

The third category of reasons is associated with portfolio decisions. In some cases, a product may be outdated, which leads the company to replace it by a follow-up product in order not to lose market shares. Furthermore, the profitability of the product plays a role in this respect. In case a product is not profitable to the company anymore, they need to consider its replacement as well. As a last reasons, the authors mention legal constraints. It might happen that the company is forced to replace the product as intellectual property laws are broken by its offering.

Similar to Haines (2009) and Jansen et al. (2011), Gimnich and Winter (2005) list two further reasons for migrating a software product to a new technological environment. On the one hand there is the customer, the company derives new external requirements from. These new requirements often are a trigger to fundamentally overhaul the product. On the other hand the old product becomes too hard to maintain, which leads to an increase in costs.

Finally, Malton (2001) reveals three reasons that influence the migration decision of a product software company. First, there is the reason of changing technology. Either the old platform of the product is obsolete or poorly supported, or a new platform and environment just provides a better future for sustaining a software product. Moreover, maintenance costs are also seen as a main trigger. Lastly, an overhaul of a product can significantly increase the business value of a company.

4.4 Approaching the overhaul process

As illustrated previously, there are different types of product overhaul strategies, such as redevelopment or migration. In this section of the thesis, different ways of how to approach these strategies are introduced.

Brodie and Stonebraker (1995) and Gimnich and Winter (2005) distinguish between two ways in which software companies can approach either a migration or a complete redevelopment:

- Big-Bang approach
- Smooth migration

In a big-bang approach, the currently running software product is replaced by the new product all at a sudden at a certain point in time. This means that the customer of a product is concerned with the replacement of the new product at one point in time as well. The second approach that is mentioned is referred to as the smooth, hybrid or incremental migration. In this approach, the old product is replaced by the new product incrementally. For software development this means that each component of the software product - might it be functionality or architectural components - is developed step-by-step. Again, step-by-step, this components are integrated in the new product, while the old product is still in used and maintained.

These two approaches are mentioned in academic literature. Although they are not specifically associated with the overhaul of product software but with software and legacy system migration, they appear to be suitable strategies for a product software overhaul as well. Besides these approaches, not much is known about any methods that are concerned with the replacement procedure of product software. Much literature is focusing on legacy system migration from a technological point of view. Methods have been proposed such as the *cold-turkey*, the *forward migration method*, the *reverse migration method*, the *Chicken Little Methodology*, the *Butterfly method* or the *FASMM: Fast and Accessible Software Migration Method* (Brodie & Stonebraker, 1995; Forite & Hug, 2014). However, all these methods are rather focused on large organizational legacy systems as well as technology. This agrees with what Khadka et al. (2014) also experienced as they illustrate that software migration is too often limited to the technological activities only. However, they found that software modernization or software migration also need to be viewed regarding organizational issues.

A method, which is related to the sunsetting process of product software has been proposed by Jansen et al. (2011). The authors present a method that should help product software companies in this process, which is called the *Product Software Discontinuation Method* (cf. figure 4.3). The method lists all activities that product software companies have to undertake in order to sunset a software product. The first main phase is discontinuation assessment, where the product portfolio is reviewed and a customer assessment is performed. In the second phase *Phase-out planning*, the discontinuation is planned in terms of having a list of alternative options (such as the replacement), creating an organizational change plan as well as performing a legal assessment. Furthermore, there is a *Pre-phase out*, where a communication policy is established that can be sent to internal and external stakeholders. Eventually, the product is phased out, which means that delivering the product to the customers stops, who are provided with a transition plan for a follow-up product. It is visible that the method is limited to the sunsetting process of product software. The authors mention some actions such as the information of the customers, it lacks a specification on what happens after the discontinuation of the product.

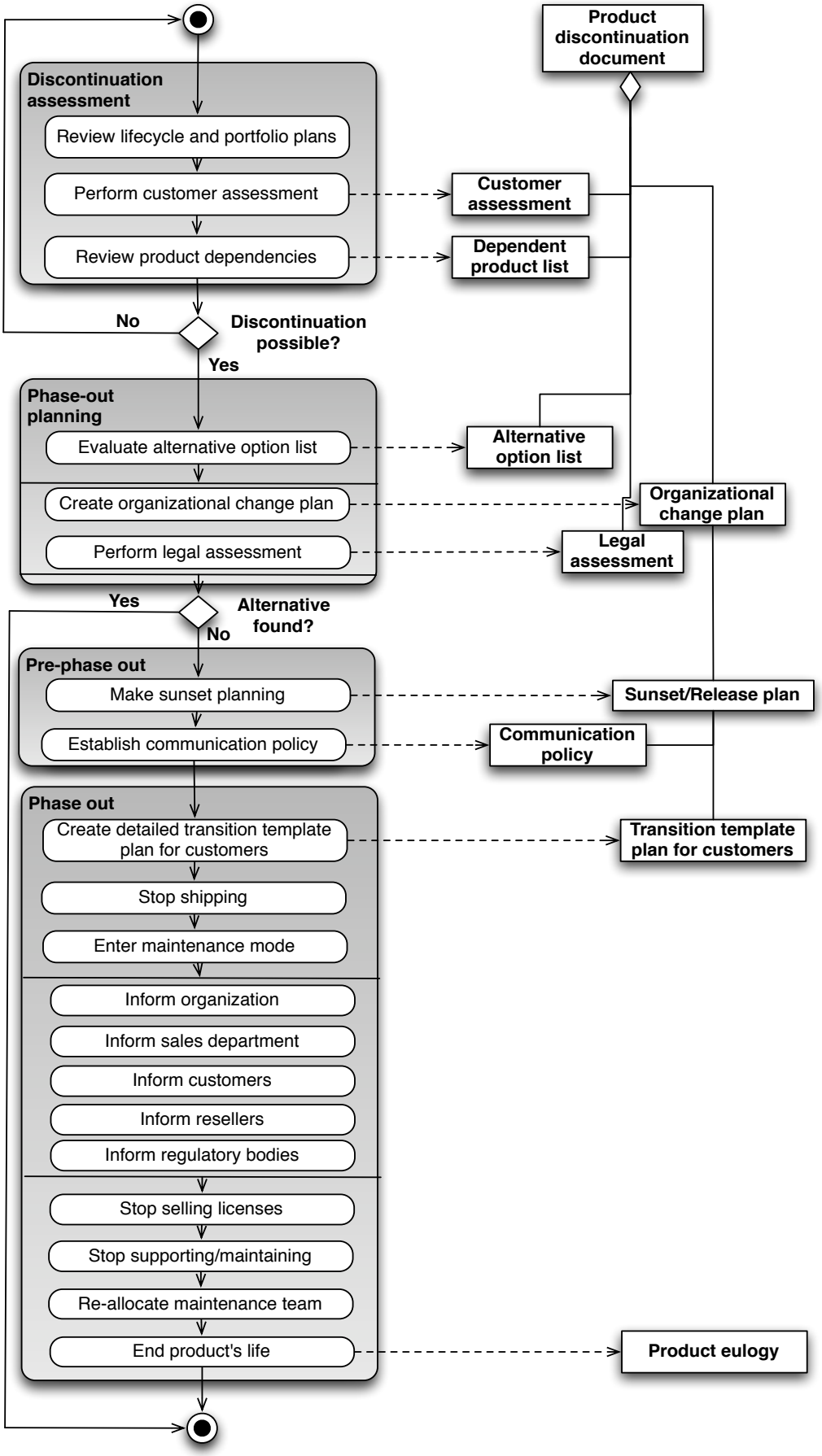


FIGURE 4.3: Product Software Discontinuation Method (Jansen et al., 2011).

Chapter 5

Case Study Results

In the previous two chapters, the theoretical framework of this thesis has been established. The software business in general as well as the major overhaul process of product software have been introduced. The practical investigation of this process constitutes the main part of this chapter, as the results of the multiple case study are revealed. As described in section 2.4 of this thesis, results are presented for each of the eight case companies. In the following sections, the main findings of each case study are elaborated, after a short introduction of the company's main business is given.

5.1 Case company A

Company A is a pure product software company, supplying specialized software for the international agricultural sector. The company, established in 1985, nowadays serves around 13000 customers around the globe, targeting two specific groups mainly: farmers and the agribusiness. The product portfolio of the company consists of products that are aimed at supporting the whole agricultural chain and all sections from production to processing to marketing.

Name	Company A
Sector	Agriculture
Established	1985
Employees	100
Annual turnover	11 Mio
Customers	13,000
Market	Worldwide

The company's primary mission is to be the market leader of software in the agricultural sector in multiple European countries. Besides their mission, the company's vision is amongst others that by 2020, software will be solely used online as well as that more and more digital data will be available in real time. These two vision statements led the company to proclaim an important goal on their strategic agenda, which is the migration of their desktop software to the cloud. The migration of the products, which were formerly built as desktop software, towards web enabled online solutions thus represents Company A's overhaul project, which is the subject of the following description.

The overhaul project - A migration to the web

Company A's product portfolio consists of software, which is made for the agribusiness on the one hand, as well as software, which targets farmers on the other hand. While the software for the agribusiness was already available online from the beginning, thus enabling the customers to operate the software through the web, the farmer software still remained as a desktop solution.[1] The overhaul project of Company A concerned the migration of their farmer software *"[. . .] from the desktop to the cloud"* as it was stated by Interviewee 1 (Appendix D, Company A, 6). As the farmer software of Company A consists of multiple several products the product overhaul is considered as an overhaul of the entire product line (Appendix D, Company A, 7-8).

Overhaul trigger - 3 major factors

As Company A states, there are three factors that triggered the overhaul of their product line:

- **Technology:** Primarily, necessity to migrate the product line to the cloud came from technology. As the desktop versions of Company A's farmer products are based on obsolete technological platforms such as Delphi/Paradox, which are not supported with updates anymore, a change to more modern platforms appeared to be necessary (Appendix D, Company A, 10). Such legacy technology is generally in the end of the product lifecycle and - in the forseeable future - it will end also, which could infuence a company's products tremendously.

- **Market:** Besides legacy technology, a central factor that triggered the overhaul project of Company A is the market, or rather the market conditions. As mentioned earlier, Company A strongly believes that by 2020 everyone is using online software. This made the company starting to invest in online software rather than in desktop solutions (Appendix D, Company A, 4). According to them, moving their solutions to the cloud entails immense advantages. One essential advantage is the installation of their software. Instead of configuring the software on around 13.000 desktop computers of their clients, the product can be hosted in the cloud, which makes it accessible through the web (Appendix D, Company A, 10). Furthermore, all data and thus information can be stored in one large database, which facilitates the process of uploading customer data (Appendix D, Company A, 12. Eventually, Company A considers their migration to the cloud as a 'game changer' as the company can enter new markets in other European countries more easily (Appendix D, Company A, 95).

- **Efficiency:** From a development point of view, there is a third factor that triggered the overhaul project: efficiency. Offering the software online instead of installing it on desktop computers does not only improve the chance to serve the market in a better way, the product can also be maintained in a more efficient way (Appendix D, Company A, 12). According Interviewee 1, the *"[. . .] technology itself is much further and faster in presenting [the functionality of the product], much more visible"* (Appendix D, Company A, 28). So there are not only advantages for

[1] At that time, web-based application could technically not be developed yet.

the developers of the company, who can maintain the software more efficiently and produce new modules more easily, but there is also the advantage for the customers as the presentation of the product was improved.

Overhaul strategy & approach - From a big-bang to a smooth migration

Company A's overhaul project was a migration from an on-premise desktop solution towards the cloud. However, it was not only a migration to a new technological platform, but every single module of their software products was redeveloped from scratch (Appendix D, Company A, 54). This decision was made as the source code of the software could not be transfered in a 1-to-1 migration. As a software product evolves over the years, it also gets more complex regarding the source code and the documentation. To develop well functioning follow-up versions of their legacy products, a redevelopment of the modules from scratch appeared to be the only way.

In any case, the modules of their products were developed from scratch. They way the company approached the redevlopment was different however. First they started to follow a big-bang approach (Appendix D, Company A, 16). This approach implies the following order:

- Rebuild of software
- Delivery of product
- Phase-out of old product

After some time, Company A realized that the big-bang approach was challenging and risky. So they decided to switch to the hybrid approach. In the hybrid approach, the company first made all data from the legacy product available in an online database. Afterwards, the development and delivery of the product was done in a rather incremental way. Small modules of the product were developed and directly made available for the customers (Appendix D, Company A, 16).

According to Company A, the hybrid approach had several advantages. First of all, the hybrid approach only required one big investment, which was uploading the data to the new database. Although this process was time-consuming for Company A, they could proceed with having smaller projects in terms of developing and dlivering modules (Appendix D, Company A, 16). The more icremental method also had more advantages for the service employees of Company A as they could become acquainted with the new product in smaller steps (Appendix D, Company A, 87).

The smaller steps of the hybrid strategy entailed constant conversions. According to Interviewee 1 of Company A, a one time conversion that is generally done in the big-bang aprroach can be highly dangerous (Appendix D, Company A, 111). This is why the company believed in the hybrid approach and stopped applying the big-bang approach.

From a customer communication point of view, the hybrid approach was also beneficial. The customer constantly received parts of the new product. Thus he was not confronted

with a complete new product at a certain point in time, but he could accustom himself to the new product step-by-step. With the hybrid approach it was hence easier for Company A to convince their customers of the new product (Appendix D, Company A, 117).

Technological aspects - Few functionality, lots of new technology

Company A's overhaul consisted of the smooth migration from on-premise software to a web-based application. This transition implied a lot of changes regarding the underlying technology. Functionality-wise however, Company A did not have to add many new features that were requested by their customers. As the company is active in the market for more than 25 years, their software can be classified as mature. The goal of the overhaul was clearly defined as the move of their products to state-of-the-art technological platforms, which facilitate maintenance of the products as well as make them more attractive for their customers.

Although functionality-wise the products did not change, there were many things to consider regarding the technological platforms that surrounded the product. In the course of the overhaul project, Company A had to consider change regarding the following platform aspects:

- Architecture
- Development environment
- Database management system (DBMS)
- Operating system
- Programming language

The architecture of Company A's follow-up products was a major aspect of the overhaul project. As the overhaul involved a switch from on-premise software to cloud-based solutions, there was a big change in the software architecture of the products. One aspect of this new architecture was the hosting of the software. A switch from desktop software to web applications means that the software needs to be deployed at a service provider instead of deploying it on-premises. Company A started hosting the software products themself, however they chose to give it over to a professional hosting company in the course of the project (Appendix D, Company A, 36).

The development environment was furthermore one major part of the overhaul of the products' technological environment. With their old product suite, Company A used several development environments such as Delphi/Paradox, Powerbuilder/Sybase or Visual Basic/Access (Appendix D, Company A, 52). In the course of the transition of their products to the cloud, the company changed the development environment to Microsoft.NET (Appendix D, Company A, 38). The reason for this choice is based on the slogan of the company: "Choose proven technology."

Besides the architecture in general and the development environment, Company A had to make changes in their runtime environment, more specifically the DBMS. When moving the desktop-based software to the cloud, a database was set up, which was stable

and performing well in a multi-tenant environment. As Company A made their choice for Microsoft solutions, the obsolete DBMSs were replaced by a Microsoft SQL server (Appendix D, Company A, 45-46). As the installation of the online database was different compared to the one Company A had before, an expert was hired, who did the setup of the database as well as several tests (Appendix D, Company A, 75). In case of online software, a large database is setup online serving a multitude of customers at the same time (cf. multi-tenancy). Thus, there need to be considered security issues as well. Confidential data of customers need to be protected from possible theft and piracy. This was, currently is and will be a challenge for Company A inthe future of their cloud-based products.

With the change of almost every part of the technological environment, also the operating system and programming language have undergone change. The company's preference for Micrososft products also influenced the decision of using the operating system of Microsoft (Appendix D, Company A, 38). Finally, Company A had to consider changes in the programming language as well. As most of their previously used programming languages did not appear to be suitable for the online solutions anymore, the company decided to use the Visual Studio solution from Microsoft, which is refered to as C# (Appendix D, Company A, 50).

As it is with every project that implies new, often unkown technology, there are external people involved. Also in Company A's overhaul project external people with expertise in a specific area were consolidated. In Company A's experience, external employees can be very helpful when you are concerned with a topic, your own team is not experienced enough in (Appendix D, Company A, 75). The company thus consolidated an expert for mobile applications as well as for setting up and administrating the architecture of the new software product. According to Interviewee 1, a company needs to be aware of the fact that during a product overhaul a lot of things change and lots of new technologies are involved, you never have concerned with before (Appendix D, Company A, 174).

Temporal aspects - A remarkable delay

The overhaul project of Company A has been a project of a hundred man-years. As the company had 20 employees (mostly developers) working on the project, this can be translated to a duration of around five years (Appendix D, Company A, 134-136).

As the company struggled with following one specific overhaul approach, the overall overhaul project was delayed. The switch from a big-bang approach to the more incremental hybrid approach made the company to extend the project by around 25 man-years extra (Appendix D, Company A, 123). Besides changing the overhaul approach, there were other factors that influenced the duration of the project itself. One thing that took the company a lot of effort as well as time was the set-up of the new technological architecture. As explained before, this was one of the most time-consuming activities during the overhaul project, although it was performed by an external expert. Eventually, Company A was not only concerned with the overhaul project itself, but there were other constraints or projects running at the same time. This was another factor that produced some delay in the overhaul project (Appendix D, Company A, 38).

The major phases the overhaul project of Company A consisted of were as follows:

- Planning the project
- Setting up the technological architecture
- Update database
- Incremental development
- Data entrance and reporting (analysis)
- Phase out legacy product

The overhaul project of Company A started with the planning phase. In this phase, a project team was introduced, which was responsible for planning anything related to the project such as resource allocation, time planning etc. A big step in Company A's overhaul project furthermore was setting up the technological architecture (Appendix D, Company A, 144). This step emerged as one of the most time-consuming as well as challenging phases during the overall project. In the company, there was one head architect, who did not do anything than focusing on the establishment of the technological framework. Another important step was the update of the new online database (Appendix D, Company A, 163-164). With the new technological architecture ready and the database updated, actual development of the new product was initiated. As Company A followed the hybrid approach, small certain pieces of the current software product were chosen to be rebuilt. Thus, the company had the vision of rather developing small pieces of software, which are then released to the customer so as to follow the devise: *"Take a smaller piece and make it work in 14 days"* Appendix D, Company A, 144). To test the developed *pieces* of software, the company applied the structure of entrance and reporting. In a rather agile way, data was thus into the system so as to test the piece of redeveloped software by performing an analysis in terms of reporting (Appendix D, Company A, 144). Finally, having finished the development of every single piece of software, the legacy product is completely replaced by an follow-up product and thus phased out.

Financial aspects - A reasonable investment

As the overhaul project of Company A turned out to have a duration of a hundred man-years, which is five years of work for 20 full-time employees (FTE's), there was a huge investment to make for the company (Appendix D, Company A, 119). Company A was only capable of starting the project as they are a highly profitable company. The project was thus not backed by any external shareholder but just by the company itself (Appendix D, Company A, 95). The high profitability of the company was also the reason that the project could have been going on for so long as there was no point in time that the company ran out of money. Before the project actually started the company made a business case, in which the company was performing all kinds of financial analyses in order to guarantee the feasibility of the project.

Project management & organizational aspects - Scrum as essential factor

Company A applied the agile development method Scrum as it suits well for development projects with that kind of hybrid/incremental approach (Appendix D, Company A, 28). The Scrum development method influenced the company's project management in such

a way that it required a proper structure. This structure invloved a lot of planning and preparation meetings, where the project team discussed further activities with the product owner (Appendix D, Company A, 87). The Scrum methodology functioned well in Company A as it is also aimed at finishing small increments rather than having everything done in one big project. An important tool during the management of the project appeared to be the operational roadmap, which contained any kind of information regarding the status quo of the project (Appendix D, Company A, 119).

> *"[. . .] to maintain everything over here, we had about 20 developers. To make this [overhaul project] possible, we had to hire 20 developers extra" (Appendix D, Company A, 63).*

The team that was working in the project was planned with 20 FTE's in the beginning - being mostly developers. As the overall project was delayed due to the issues with the right overhaul approach, Company A had to hire 20 developers extra to support the project and accelerate it (Appendix D, Company A, 63).

The company has its own development department. This is especially helpful to rapidly respond to current events and developments in the market (Appendix C, Company A). Nevertheless, the company was trying to outsource smaller development parts of the project to speed up the process. Thus, developers from India were supporting the project for a while. However, as the company rather prefers to work with progammers in-house and the cooperation did not work out to funtion well, it was stopped soon (Appendix D, Company A, 158).

Market and customer aspects - Customer communication as basic principle

One of the main triggers of Company A's overhaul project was considered the market. Primarily the company wanted to develop the new version of their product in order to meet today's technological requirements and provide their customers with online software. However, the company did not make a new software product that aims to target a new market segment (Appendix D, Company A, 93). Of course, the company prepared itself for the future as they can now enter other European markets more properly, however this was not the primary objective.

The hybrid approach the company applied to perform the overhaul project also implied benefits for the customer interaction process. The customers of Company A played a central role in the project as they were the ones, who needed to be satisfied with the new product most. An important means during the whole project was thus also customer communication. When Company A migrated their new product from the desktop to the cloud, their customers were informed in the early stages. The company provided them with newsletters and also held large presentation to show the benefits of the new software product (Appendix D, Company A, 105). Besides that, the company generall offers an online selfservice, which is available at 24 hours a day (Appendix C, Company A). All these customer communication activities served as a means to keep the transition process to the customers as smooth as possible. Once the new software product was delivered to the customer, the configuration process was mostly driven by Company A. To instruct their customers about the new software product, Company A provided trainings.

Challenges - A highly complex project

The overhaul project appeared to be time-consuming and challenging for Company A regarding the following aspects:

- **Complexity:** According to Interviewee projects of more than ten up to fifteen man-years are hardly manageable by typical product software companies (Appendix D, Company A, 16). As the overhaul project turned out to be a project of 100 may-years, the company was facing a highly complex project, which entailed several challenging activities.
- **Overhaul strategy selection:** One challenging activity in the very beginning of the overhaul project was the selection of a proper overhaul strategy. The company started the project applying a big-bang approach. Their actual plan was to first finish with development, after which they would have release the new product at a certain point in time to their customers. As this approach took too much time, the company later decided to switch to the hybrid approach. Thus, they could release smaller packages in a shorter period of time.
- **Architecture development:** The setup of the new fundament, the architecure was a challenging task as the required multi-tenant architecture was completetly new to them.
- **Maintenance of legacy product:** Company A experienced that most of their developers did not want to work with the old technology anymore (Appendix D, Company A, 77). To overcome this challenge, the company hired two experienced developers, who were eager to maintain the old product with the old technology until its actual phase out.
- **Innovative pressure:** If the company would have followed the big-bang approach until the end of the project, their customers would not have seen any change before. It was a challenge to overhaul their products and to constantly deliver innovations to the market at the same time. Innovations that were required by the customers as they wanted to see the benefits of the new product (Appendix D, Company A, 113).

Lessons learned & advice - Follow the KISS principle

During the overhaul project, Company A truly derived some lessons learned and adivces that they can forward to other companies, which are concerned with such a project. One important aspect is that companies need to make a choice about the overhaul strategy and approach (Appendix D, Company A, 170). This is a basic lesson, Company A learned in the very beginning of the project as they decided to change their overhaul approach. For the company, the switch to the hybrid approach was beneficial. However, they would only suggest to apply the hybrid approach, if a company's product is in the later stages of the product lifecycle. Finally, Company A suggests to follow the KISS-principle: *"Keep it simple stupid."*

5.2 Case company B

Name	Company B
Sector	Labor market
Established	2001
Employees	50
Annual turnover	
Customers	400
Market	International

Company B, founded in 2001, is focused on delivering software products to the labor market, particularly to companies with HR and recruitment processes - which is where people are looking for jobs, and companies are looking for employees. In this sector, Company B is specialized in information extraction, document understanding, web mining and semantic searching & matching (Appendix C, Company B). The company is trying to make the recruitment process more intelligent and meaningful by enabling other companies to efficiently extract information from CVs and job profiles on the web. With their software solutions, the company supports around 400 customers in The Netherlands, France and Germany.

The mission of Company B is thus to *"[...] leverage Artificial Intelligence and natural language processing technology to make the labor market more transparent and more efficient"* (Appendix D, Company B, 4). Around 50 employees are constantly concerned with producing and delivering software products, which enable computers to convert unstructured textual information into usable data.

The overhaul project - Evolutionary process and web migration

Company B's software products are designed facilitate the extraction of data from unstructured documents as well as to provide matching and search solutions to make any kind of information available to the users. During the last decade, the company has experienced a lot of change in IT and technological change in the area of natural language processing and the labor market. As technology is evolving, it happens that Company B needs to overhaul their products in order to meet the requirements of expanding feature sets. A specific example is represented by the migration of one of their products from desktop/ client-server technology to the web. With that migration, the company started redeveloping their Perl-based modules (mostly modules of the UI) by Java-based ones. This process started some time ago, however the company did not see this migration as a project with a predefined duration, but rather as a continuous change, which is still in progress nowadays (Appendix D, Company B, 12). In essence, Company B is considering their product overhaul as kind of an evolutionary process. Main reason for the company to follow this process is that the company is rather a smaller company, which cannot have a second development team that is completely focused on such a big overhaul project.

Overhaul trigger - Technology, personnel, customer

The overhaul of Company B's products was triggered by three major factors: technology, personnel and the customer. In particular, the reasons can be summarized as follows:

- **Legacy technology:** One main trigger for the product overhaul was the desktop/ client-server technology, the product was based on, has become obsolete (Appendix

D, Company B, 12). Besides this outdated technology, the popularity of the programming language Perl, the product was written in, declined as well. In order to stay competitive in a fast growing market, the company had to adapt to new technologies that came along (Appendix D, Company B, 10).

- **Personnel requirements:** As technology became obsolete, developers, who were aware of that technology became scarce. New developers, the company wanted to hire, were not aware of the Perl anymore, which made the company thinking of switching to some more modern languages (Appendix D, Company B, 32). By switching over to state-of-the-art technology, more people could meet the company's requirements.
- **Extension of features:** Finally, one of the most important triggers for Company B was the feature completeness of the product (Appendix D, Company B, 30). This was especially crucial for the company in order to satisfy their customer, which required an extension of the existing functionality (Appendix D, Company B, 31-32).

Overhaul strategy & approach - Rebuild in an evolutionary manner

> *"[...] it's kind of an evolutionary approach, where you try to slice down major changes into manageable and prototypable events [...]" (Appendix D, Company B, 131).*

The company did not make any big plans before starting their product overhaul. Neither did they consider it as a project, which requires proper finanical forecasts and a predefined scope as well as timeline. Nevertheless, the company had a specific strategy while facing the product overhaul. Whenever the company was forced to overhaul one of their products, it happened in such a way that they rebuilt any module of the product from scratch, which was then released to their customers with an incremental approach. The company considers this approach as the evolutionary approach. Along with that evolutionary approach, the company introduced the concept of innovation weeks (Appendix D, Company B, 20). In such innovation weeks, everybody in their company can propose a kind of innovation project. Every now and then, the best innovation project is voted. The idea with the most votes is responsible for developing a functioning prototype with one week. According to the company, some of these prototypes finally turned out to be the next generation of the product.

Technological aspects - Architectural change and new programming language

> *"Well, in terms of cloud, what is cloud, I don't know, when you deploy it with Amazon or something like that, right? Well, we process a lot of privacy-sensitive data [...], so we host everything ourselves. So we haven't gone into the public cloud" (Appendix D, Company B, 62).*

With the migration of their product to the web, the company had to adapt some new technologies as well. This started in the architecture of the product, which has been changed from a client-server architecture to a web-based multi-tenant architecture. Furthermore,

the company started replacing their Perl-based application by Java-based modules. However it was not a complete switch from Perl to Java as the core extraction engine of the product was still kept the same. Only the layer on top of the engine was replaced by a Java-based web application layer (Appendix D, Company B, 12). Besides Java, some parts were also developed in Python from then on (Appendix D, Company B, 71-72). Although the company did migrate their product towards the web, they did not move it to the cloud as they hosted the application themselves (Appendix D, Company B, 62)

Temporal aspects - No project, no specific timeline

As the company is having some kind of evolutionary processes, the product overhaul is not considered as a big project with a deadline. A change in their products can be done in a week, but it can also have a duration of several years. The migration from client-server technology to the web for instance is constantly running since several years. It is furthermore difficult to say, whether there was any delay in their product overhaul. According to Interviewee 2, there is always delay in the evolutionary approach, which is caused by the fact that there are several other activities going on in parallel.

In general, the major phases of Company B's product overhaul are composed as follows:

- **Identification of change:** First there is a stage, where the required change of the product was identified. After communicating with people within the company as well as customers, a decision is made whether a specific change is needed.
- **Preparation:** The second stage of a product overhaul is to have preparation meetings, in which the overhaul is roughly planned with internal people such as developers, sales as well as marketing.
- **Identification phase:** In the identification phase, draft design and plans are made, which serve as a framework for the actual development.
- **Prototype development:** Another major phase of the overhaul is to actually develop a prototype of the product. Therefore, the modules of the product are split up and transfered into prototypes.
- **Finalization:** The last phase of Company B's product overhaul is to further develop the prototype, if the represent a potential for the company as well as for their customers. If so, the new version of the product is further evolved by the company's evolutionary approach, which also means that more resources are put on development.

Financial aspects - Guided by feelings of demand

> *"But overall we are more guided by our feeling of demand in the market rather than actual scientific studies of it" (Appendix D, Company B, 48).*

Applying their kind of evolutionary approach, Company B is neither doing any big calculations for their product overhaul, nor they have an indication of what a product overhaul generally costs. In any case, the company is funding all kinds of changes purely from their

own revenue (Appendix D, Company B, 54). One additional source of funding for the company - more like a tax incentive - is the WBSO, a Dutch subsidiary, where companies get a tax deduction for product innovation.

Challenges - Technology, Organization, Customer

Also with their evolutionary approach, Company B faced some challenges.

- **Visible benefit & effort:** A big challenge for the company when producing a new version of a product was to keep a reasonable balance between visible benefit and effort (Appendix D, Company B, 45). Thus the company tried to avoid restructuring big modules without increasing the feature set of the product. It was very important to deliver value to the customers as well. Only modifying the code base of the product to make it more easy to maintain would have implied no improvement to their customers. Thus, it was always a challenge to have a balanced ratio of the effort and the visisble benefit, the customer receives.
- **Balacing short-term urgency with strategic importance:** Another challenge for Company B was to allocate resources so that a strategically important long-term project was realized, while there was lots of day-to-day pressure (Appendix D, Company B, 131). In order to overcome this challenge, the company tried to slice down small chunks of the software for the actual development.

Lessons learned & advice - Be aware of the company's mission and vision

Finally, a tip that Company B can give to other companies, which are concerned with the overhaul of their products, is as follows:

> *"It's the story why you have that software and what's the value, what's the vision of features and how the customers use them, that has to guide [the product overhaul]" (Appendix D, Company B, 133).*

5.3 Case company C

Company C is a global provider of Web Content Management and Online Marketing software, established in 1996. Around 100 employees serve customers around the globe, whereas the Benelux and the United States represent the their major target markets.

Name	Company C
Sector	CMS software
Established	1996
Employees	100
Annual turnover	15 Mio
Customers	
Market	Worldwide

Company C's mission is to provide their customers with *"[...] the best technology and solutions to maximize the effectiveness of their online communications to fulfill their business goals"* (Appendix C, Company C).

The overhaul project - A redevelopment of the user interface

Company C produces two kind of software: a web content management system (WCMS) as well as an online marketing software product. Some time ago the company was concerned with an overhaul of their WCMS. The overhaul of the product consisted of the major change of the product's UI. Main goal of their overhaul project was to better support the processes that their customers are doing in the WCMS. The main processes of the overhaul project were the redesign of the UI as well as the development of new functionality for their customers. Along with those two processes, Company C put effort in modifying the database layer of the product's architecture.

Overhaul trigger - Outdated technology and the market

The product overhaul of Company C was mainly triggered by two factors: technology and market.

- **Outdated technology:** The UI of the company's old product was quite old and in the meantime based on outdated technology. Some of the the users of the product even considered the product as *"[...] ugly and not fun to work with"* (Appendix D, Company C, 8). This made the company thinking of introducing a new UI and thus a new version of their product in order to satisfy their customers.
- **Improved market conditions:** Secondly some market analysts adviced the company to redevelop the UI of the product in order to better meet the requirements of the market (Appendix D, Company C, 14). According to the analysts' review, Company C's previous UI was not good enough to distinguish themselves from competitors.

Overhaul strategy & approach - Redevelopment with a big-bang

Company C applied the strategy of redeveloping the components of their old product from scratch. The company took some older parts as a basis, however they finally rewrote most of the code. The product overhaul project of Company C was done in one big project, after which the company released the new product to their customers in a big-bang

scenario. First, each component of the product was redeveloped, then the product was finalized and delivered to the market.

Technological aspects - New architecture and programming language

Technology-wise, Company C kept most of the functionality of the product as it was before. In addition, several features that were required by the customers and internal stakeholders were added to the new product.

Along with the changes on the feature set of the product, Company C was concerned with changes in the architecture as well as the programming language. Concerning the architecture, Company C incorporated a new UI framework called Dojo (Appendix D, Company C, 30). This was done in order to facilitate the communication with the RESTful APIs, which are APIs that adhere to the architectural constraints. Concerning the programming language, the company mostly added components based on Java script.

Temporal aspects - A slight delay due to unpredictability

Considering the project kick-off as a starting point for Company C's product overhaul, the project had a duration of approximately one year. Originally, the project was planned with a duration of around eight til 9 months, which means that the project was extended by three to four months. The following statement provides an explanation for this delay:

> *"But we didn't really know how much work it was and really hard to estimate. And we've tried to estimate it, but it was really hard. Because we used new technologies, a lot of dependencies, you have to build, a new UI and RestAPI's against the old architecture, so you don't know what's going to happen" (Appendix D, Company C, 50).*

To provide a clear picture of what Company C specifically did during the software product overhaul, the major phases of the overhaul project are described in the following:

- **Planning phase:** In the planning phase, the company was involving their customers for several decisions first. Such decisions considered the the functionality of the product mostly. The company also created a business case, which was however not very detailed. According to Interviewee 3, the company rather chose the pragmatic way (Appendix D, Company C, 54). The planning phase moreover contained some architectural vision sections, in which the new target architecture was defined.

- **User experience research:** An essential phase before starting the development of the new product was the conduction of user experience (UX) research. The main purpose of this phase was to find out about the customers', users' and developers' needs.

- **Development:** Based on the results of the UX research, Company C started to develop a prototype of their new product with so-called *1-10-100* project approach. Applying this approach the company developed a functional prototype during the first day, while involving some of the stakeholders of the company. To further evolve this prototype, the same procedure was repeated in a timespan of ten days. The

company however ran over time and decided to start with the development of the final product together with the whole R&D team.

- **Release:** Parallel to the development of the new product, Company C already involved their customers in terms of marketing materials or trainings in order to prepare them for the switch to the new product.

Project management & organizational aspects - A small team with another expert

Company C planned the overall product overhaul project with a team of eight employees from the beginning. As the project turned out to be more complex than expected, the company added four developers to finish the project in an acceptable timeframe. These developers were not hired, but they came from another department of the company itself. After Company C finished with the software product overhaul, they reallocated the developers again.

In first phases of the project, mostly developers were involved. When it came to delivering the product to the customers, also sales people, marketing, support people and professional service people were integrated.

In addition to internal employees, Company C was consulted and supported by a specialized visual designer/interactive UX architect as the company did not have the required skills.

Market and customer aspects - Well functioning customer communication

The communication and information exchange between the company and the customers was managed well. To inform the customers about the new product, Company C did several things:

> *"We had informal account management, we had newsletters, we have had also a COMPANY C connect meeting, which is a customer day, and we have demonstrated our product. And some of the customer, we already involved in the beginning of the project, so they already knew something was going on" (Appendix D, Company C, 114).*

Challenges - "We couldn't get it running"

For Company C, the most challenging activity was to get the software product overhaul started. As the company rather applied a pragmatic approach, the necessity of the product overhaul was not clear from the beginning on. Thus, there were discussions about whether the product overhaul would be reasonable or not.

5.4 Case company D

Company D is a software product company that develops standard application for group life insurance and pension administration. The company is globally active with around 80 employees.

Name	Company D
Sector	Pension funds administration
Established	1986
Employees	80
Annual turnover	15 Mio
Customers	
Market	Worldwide

The overhaul project

The company started with the migration of their product from Windows to the web. In essence, the product of the company is 25 years old, so the change to a more up-to-date technological environment appeared to be necessary. Therefore, the company decided to migrate from an oudated Progress platform to a Microsoft.NET platform. The project was split up in two major parts: first, the company was migrating the user interface to the new technological environment. After that - and the company is currently still concerned with that activity - the company wants to migrate the business logic.

Overhaul trigger

The main trigger of the company's overhaul project were as follows:

- **Technological reason:** In the old environment, it was not possible to develop a fully web-based application. The Progress platform, the product was based on, is furthermore considered as old-fashioned, which is not state-of-the-art technology.
- **Market reason:** Another problem that existed for the company was that potential customers did not know the technology, their product was based on. As most of the companies did not want to get the product because of the oudated technology, Company D was forced to do the product overhaul in order to reach new customers.
- **Scarcity of qualified personnel:** The third factor that triggered the software product overhaul was the scarcity of qualified personnel. As the product was based on technology, many young people are not familiar with anymore, the company had to ove to state-of-the-art technology in order to attract qualified developers.
- **Decrease of maintenance costs:** With the new application, the company also believed that maintenance costs will be reduced (Appendix D, Company D, 40).

Overhaul strategy & approach

The company chose the strategy to perform the product overhaul in an as-is migration, trying not to redesign or refactor any parts of the software. They chose this kind of strategy in order to *"[. . .] keep control of time and costs"* (Appendix D, Company D 14). Furthermore, the product of the company is an extensive application, which would have made the redevelopment even harder and which would have involved bigger risks to move

to another platform. An as-is or 1-to-1 migration was possible for the company, as they defined a layered architecture a couple of years ago (Appendix D, Company D, 26).

The approach the company applied for this software product overhaul was a mixed approach. In order to migrate the UI, the company released the finished components in a big-bang scenario. The other part of the product overhaul project will be apporached in a more incremental way. Then every module of the business logic will be developed and delivered right away. The reason for applying the big-bang approach for the UI migration is that it is not possible to deliver *"[...] 10 screens web-based and 50 screens still Progress-based"* (Appendix D, Company D, 18).

Technological aspects

The most essential technological changes were explained by the switch to a web-based application. The company almost changed every single component of the underlying technology. The change from the Progress platform to a Microsoft.Net platform entailed changes in the DBMS, as well as the programming language, which was chosen as .NET. On top of that, a web server had to be installed for the new product.

Temporal aspects

The project duration of Company D's product overhaul was estimated with around five years. The migration of the UI that the company already finished, had a duration of around nine month, which the company did not overrun.

The major phases of the company during the product overhaul project furthermore were as follows:

- **Partner selection:** As the company did not have the knowledge of such big software transitions, they selected a partner that supported tha project regarding several aspects, development in particular.
- **Proof of concept:** Right after the partner selection was finished, the company elaborated a proof of concept, in which technological aspects were discussed, the architecture was defined and some cost estimations were done (Appendix D, Company D, 32).
- **Overhaul plan:** Before the company started with the actual development, they created three kinds of project plans: a technical migration plan, organizational change plan, and a customer implementation plan.

Financial aspects

The rough estimations of the company before the project revealed costs of 30 million Euros for the whole product overhaul project including hardware, software, personnel costs and costs for outsourcing (Appendix D, Company D, 20).

Project management & organizational aspects

The project of Company D was supported by an external company. This kind of outsourcing was beneficial for the company as they could meet deadlines of the project as well as develop the product more efficiently. The external company was integrated in the company during the project and provided with space within the company's building. Thus, the project team of Company D and the team of the partner could cooperate in a better way.

With such a big change, the developers of the company were challenged regarding their capabilities of the new technology. To test the knowledge of their employees, the company started a .NET assessment, which ended disappointing for Company D (Appendix D, Company D, 109). Only a few of their employees participated in the test. Out of the small group, the company selected a few developers, which finally became part of the project team. The company even had to hire extra personnel, as no more internal developers finalized the assessment with satisfying results. The fact that only few of their developers participated in the assessment had another consequence: what to do with them afterwards? So far, the company is still hiring those developers, as they are still busy with maintaining the old product. However, as the company plans to phase out their legacy product in the near future, they are facing a challenge related to the management of the developers, which are not able to work with the technology of the new product.

Challenges

One of the most challenging factors that Company D mentioned is related to the following saying: *"De winkel blijft tijdens de verbouwing open"* (Appendix D, Company D, 21). This Dutch saying means that there are several parallel activties going on besides the product overhaul project. Thus the company cannot just close down for a couple of years in order to finish with their overhaul project. Serving the market with the old product is essential to the company during the migration in order to keep the cash-flow going.

Furthermore, it was a challenge to convince the stakeholders of the company of the purpose of the project in order to get a suitable business case.

Another major challenge is related to the following statement, which concerns the migration of the busines logic of the product, the second part of the company's overhaul project:

> *"We want to migrate the whole application, but as more far you get, the business value for the customer gets less and the costs get higher. UI is pretty simple, it's added value for the customer, it's pretty defficult, because he sees nothing for it. For us it's much more complex than only the UI" (Appendix D, Company D, 37).*

Lessons learned & advice

According to the company, there are the following guidelines to consider:

> *"[. . .] you need to really get control of the migration and that's only possible with reducing any noise within that migration. So no new functionality, if not*

necessary, no redesign, just basically focus on the migration and do all those other things in parallel or after the migration, but dont interfer this with your migration itself" (Appendix D, Company D, 151).

5.5 Case company E

Company E, established in 2004, is a product software company, active in the healthcare sector. The company develops products to support companies in care such as hospitals or mental health care companies. In this respect, the company does three things with their product: first, they ease the management of medical records. Second, they do the legislation of treatments. Third, the company does the entire billing process to all insurance companies.

Name	Company E
Sector	Healthcare
Established	2004
Employees	95
Annual turnover	8 Mio
Customers	700
Market	National

The overhaul project

The overhaul project of Company E started about two and a half years ago. The decision was made back then to make a major overhaul of the product. This major overhaul concerned the user interface of the company's product, which needed to be redeveloped in order to meet the requirements of the market.

Overhaul trigger

The main triggers of the product overhaul project were as follows:

- **Performance issues:** As the company also entered new market segments and got bigger clients, the product lacked in its performance for customers with a lot of data (Appendix D, Company E, 15).
- **Structure of code:** Another reason for the company was the structure of the code of the product, which was considered as "spaghetti code" (Appendix D, Company E, 15).
- **Customer wishes:** Furthermore, the customers of Company E wished to have the product along with a redevloped user interface.

Overhaul strategy & approach

The company redeveloped their product from scratch and released it to the customer in an incremental way. Each of the modules of the product were thus redeveloped step-by-step, after which they were delivered to the customer. As the company already had introduced a SaaS-solution, the incremental roll-outs of the modules did nor represent a problem.

Technological aspects

The product of company E was relatively young, but still already considered as outdated in terms of its user interface. However, it was not surrounded by legacy technology, which was already decades old. Thus, the company did not need to change many parts of their product except for the new architecture of the user interface.

Temporal aspects

The project of the company was originally planned for one and a half years. However, as the company had problems with planning the project and stating *"a clear goal or a clear definition"* of what they want to achieve with the product overhaul, the overhaul project took the company twice as long.

Financial aspects

The total costs of the company's project were estimated as 45 million Euros, which is 35 people working on the project full-time for two and a half years (Appendix D, Company E, 63). This extremely high number was the reason, why the company almost ran out of money during the project. The consequence for the company was to cut costs in terms of firing employees. Within a timeframe of about two months, 15 people were sent home in order to avoid bankruptcy.

Challenges

There were several challenges that the company had to overcome during the overhaul project. The major challenge and problem was that the management board of the company was not clear about the purpose of the overhaul project. Thus, the company struggled with starting the project. It took them aorund one and a half years to define the precise goals of the overhaul project, which finally led to an immense delay of the project.

Lessons learned & advice

The main lesson that Company E derived from their software product overhaul project is as follows:

> *"I think the whole process was driven by optimism. But it was a bit naive, so not enough research, not enough planning, not enough work preparation. And trying to do too much at the same time. And then a huge optimism that everything will go ok. So there's two sides. On the one hand this optimism is, what made us grow, made us big, but in this case that same optimism has led to a huge increae of costs and duration of this project" (Appendix D, Company E, 125).*

5.6 Case company F

Company F is a realtively young company, founded in 2006, which is focused on delivering solutions for fraud and risk management for the insurance market. With their 40 employees, the company serves around 100 customers inn the European insurance sector. As the company is privately owned, no information about financial numbers is published.

Name	Company F
Sector	Insurance industry
Established	2006
Employees	40
Annual turnover	-
Customers	100
Market	International

The overhaul project - UI redevelopment

Company F provides their customers with a platform, where several software products are offered as a service. The project that the company was currently concerned with dealt with the restructuring of their SaaS-based software solutions. The changes the company applied to their products were manifold. First, the company renewed the user interface layer of their products. Secondly, the company was focused on setting up more services, thus performing a rebuild of the business layer as well.

Overhaul trigger - Outdated technology and customer wishes

The main reasons for Company F to start the software overhaul project were related to technology on the one hand, the market on the other hand. The three major triggers of the project can be summarized as follows:

- **Outdated technology:** The major reason for company F to start their overhaul project was related to user interface, which was considered as *"outdated"* (Appendix D, Company F, 8).
- **Customer wishes:** Secondly, customers of the company were coming up with new reuirements, which the company forced to renew the product (Appendix D, Company F, 8).
- **Efficiency:** The third main reason for the company was related to the fact that their current products did not have flexible user interfaces, which made the maintenance of the products more difficult (Appendix D, Company F, 23).

Overhaul strategy & approach - Redevelopment in a hybrid way

Company C was redeveloping every module of their new user interface from scratch. In order to finish the redevlopment of the modules and to deliver them to their customers, the company applied the hybrid approach. The company was able to do that as their software products can be rather considered as service, which are offered through the web. As one of the founders of the company stated, *"[. . .] most activities are based on services and not on user interface clicks"* (Appendix D, Company F, 14).

Technological aspects - Change of the architecture and programming language

In the course of the software product overhaul project of Company C, some of the technological components of the product have undergone changes. These changes were related to the architecture on the one hand, the programming language on the other hand. Although the company had some components already developed in JavaScript and HTML5, with the redevelopment of the user interface they changed completely from ASP.Net to the two aforementioned languages (Appendix D, Company F, 8). The architecture of the product was generally reconstructed in such a way that it fulfilled the requirements of a service-oriented architecture (SOA).

Temporal and financial dimension - Financial support by key customer

The company's plan to rebuild the largest part of the user interface of their products was estimated with approximately one year. According to the company, the Scrum methodology was helpful in reaching this goal. Not to overrun the estimated duration of the project was furthermore essential, as the company had a big client involved in the project, who financially suppported it. A delay in the product overhaul project would have dissatisfied this customer.

Conerning the costs of the software product overhaul project, the company did not provide any details. However, according to the founder and CTO of the company, it is difficult to estimate the costs of such a project as *"[...] it's not only software development, but also project management, configuration, training and implementation*" (Appendix D, Company F, 33).

Project management

In order to efficiently manage the software product overhaul project, product development was seperated from service management, service management again was seperated from product delivery (Appendix D, Company F, 37). The development of the product was done by means of the Scrum agile development methodology. Thus, one Scrum team was resonsible for the development of the product, whereas another team was responsible for "[...] new innovations and support and services" (Appendix D, Company F, 37).

There were also some changes in the organizational structure of the company, which were related to the structure of the teams. Primarily, the company planned their development activities with three teams, which were in the course of the overhaul project reduced to two teams.

Challenges & lessons learned

As the company was already concerned with a rebuild of their products before - which they could not finish - they were a step ahead compared to other software companies. Back then the company made the mistake that the project was not properly planned from the beginning on. The motto back then was: "let's try it and do it" (Appendix D, Company F, 61). In their second software product overhaul project, the company realized that such a process required more planning and a more structured way of working. Furthermore, the company experienced that people react differently to technological change:

> *"But we also learned that switching to new technology, some people are very good and really like new innovative technology. So this is my chance of making a new architecture with all the new stuff, all the cool stuff that's available, so I can really make a good version for that. So they really enjoyed that. So that was really nice to see. But, let's say, some are faster in adapting new technologies than others" (Appendix D, Company C, 61).*

The biggest challenge of the company during the overhaul project was to satisfy their stakeholders, particularly the key customer, who financially supported the project. Thus it was an essential goal for the company to finish their product overhaul in time.

5.7 Case company G

Company G is a globally acting software company that delivers enterprise ressource planning (ERP) solutions to around 10,000 customers. Their software product contains several features to support companies in activities such as customer relationship management (CRM), human resources (HR), finance, logistics or invoicing. With a number of 320 employees and a turnover of over 70 million Euros, the company can be considered as a big player in the area of product software.

Name	Company G
Sector	ERP software
Established	1996
Employees	320
Annual turnover	71 Mio
Customers	10,000
Market	Worldwide

The overhaul project - A cloud migration

Currently, the company is concerned with the overhaul of their ERP product. According to the company, the future of product software is to have them developed as web-based application. Therefore, the Company G decided to move their ERP product "from Windows to web" (Appendix D, Company G, 2). More specifically, the company wants to move their product towards the cloud, which means that it will be hosted in the data centers of an external hosting provider. Along with the transition to a fully web-based application, the architecture of the product will be renewed completetly. The company is currently working on this migration, however they did not specify any deadline from the beginning. As the company is successfully delivering their current product, they do not see any need to finish the product overhaul process in a specific timeframe.

Overhaul trigger - Legacy technology

The reasons for the company to start with their product overhaul project are manifold. The most essential triggers were named as follows:

- **Legacy technology:** The main reason, why the company chose to move their software product from Windows to the web, is legacy technology. As their Windows-based version becomes older, the technology that surrounds the product becomes outdated. An overhaul of these legacy components thus seemed to be a crucial task for the company (Appendix D, Company G, 6).

- **Customer wishes:** The transition of the product to the web is furtermore driven by the customers, who wish to have the product deployed online on the one hand, and who wish to have an extended feature set on the other hand.
- **Efficiency:** Besides legacy technology and the customers, who wished the web-based version of the product, the company believed that the product can be maintained more efficiently and with less costs in the new environment (Appendix D, Company G, 10).

Overhaul strategy & approach - Redevlopment in a big-bang

The company chose to do their overhaul project as a complete redevelopment from scratch, or as the company would rather say: *"from zero to hero"* (Appendix D, Company G, 4). This choice is also explained by the fact that the company is changing most of the underlying technology fundamentally. The company furthermore will release their new product only when each component of the product is completely developed. This meas that the company will release their product in a complete replacement all at a sudden - a big-bang scenario (Appendix D, Company G, 14).

Technological aspects - Open up the path to a new technological era

From a technological point of view, the company is entering a new sphere with their new product. Their main goal is to develop their product fully defined, which means that the product will not need developers anymore. The core engine of the product is currently redeveloped, which then enables to generate the user interface of the company's product. Thus, for the company it is not only redevloping their product from scratch, but they apply a new way of thinking. For the company, the software product overhaul represents a major step into a new technological era.

In the first step, this technological change only affects the architecture of the product and the functionality, which is extended. The user interface however is not being changed from the beginning, as the replacement with a big-bang scenario could dissatisfy customers.

So far, the company only decided about the change of the architecture as well as the programming language. In components such as the database management system or the underlying hardware platform, the company did not make any decisions yet. Company G did not decide about these technological changes, as it is difficult to forecast any trends in IT. This is why the company aso had problems with choosing a suitable programming language. The operational manager of the company emphasizes this statement: *"about 4 years ago everybody said, you have to do it in silverlight, that's the technology you have to use, so we did. And two years later, everybody said it's HTML5. And now we generate our new product in HTML5"* (Appendix D, Company G, 6). The company now decided to switch from ASP.Net to Java and HTML5.

The structure that Company G chose to develop their product was to develop and test the components of the product every day. Only when the product will be fully redeveloped, the company will move to the stage of releasing it.

Temporal dimension - No pressure, no deadline

As Company G did not set any deadline, it is not possible to talk about the the temporal dimension of their product overhaul project. The following statement emphasizes that the company's thinking:

> *"[. . .] when you look deep in our hearts, we want to have it tomorrow. But if we deliver, it doesn't need to be good only, it needs to be great. So we have to think about the future. If it's a year earlier, it's fine. If it's a year later, then it's also fine. If it's better then, then it's great. If we look at today, we are still doing good with the product" (Appendix D, Company G, 28).*

Financial aspects

Similar to the company's thinking regarding the temporal dimension of the product, the company did not make any extensive financial plans. According to the management of the company, the overhaul project is an investment for their future (Appendix D, Company G, 36). The following statement underlines the company's thinking about the financial aspects: *"This is our dream. So we are not building a business case. Financing the dream is important, but we are doing so great, so it's possible"* (Appendix D, Company G, 36).

Project management & organizational aspects

The company started their project with around 23 developers. As the project appears to be a major step for the company towards the future, they hired seven developers extra in order to develop new functionality more efficiently. As the company aims to move their product towards a new technological era, they will furthermore undergo changes in the organizational structure. One of these changes is related to the functions that will not exist anymore in the future.

Challenges & lessons learned

A problem that Company G encountered during their product overhaul was to find the right people that can support the project. Besides that, there was the risk that essential knowledge might get lost, if staff will leave the company.

5.8 Case company H

Company H delivers innovative design software for the installation branch and technical housing management. Since 1990, the company supports customers from the building industry to develop better technical installations. As of today 140 employees are busy with serving more than 3000 customers in the Netherlands, Belgium and France. With a daily user grroup of more than 8500 egineers, the company is the market leader in this segment.

Name	Company H
Sector	Building industry
Established	1990
Employees	140
Annual turnover	-
Customers	3500
Market	International

To keep their position in the market, Company H's mission is to enable engineers to *"[...] design, visualize, control and optimize technical installations"* (Appendix C, Company H). For the company, it is highly important that their customers benefit regarding three aspects: cost savings, quality and innovation.

The overhaul project - A switch from Windows to client-server

Company H has experienced several overhauls of their product during the last 24 years.[2] One of their biggest overhaul projects was when the company decided to switch from Windows installations to the client-server model (Appendix D, Company H, 14). From a product view, this project project was kind of a restart for the company. It was the same as if you would have started a new company with a complete new product (Appendix D, Company H, 60).

Overhaul trigger - Technology as a driving factor

The factors that triggered the overhaul project of Company H were manifold. However it was particularly technology that was crucial for starting such an overhaul project (Appendix D, Company H, 40). One of the reasons related to technology was that their Windows-based version of the product was just an updated DOS-version, which means that it could be considered as old-fashioned legacy technology (Appendix D, Company H, 20). Furthermore, maintaining the product efficiently in its technological environment became difficult for Company H (Appendix D, Company H, 40).

While legacy technology affected the product in terms of its performance as well as in terms of the ability to be efficiently maintained, there was another consequence of the product being built upon outdated technology: the scarcity of people that were aware of the technology. This problem was mainly related to the programming language, which was C back then. The employees of the company as well as people outside the company were tending to work with solutions that were backed with modern technology rather than the old-fashioned one (Appendix D, Company H, 40).

[2]Although the company has undertaken several overhaul projects so far, the results are presented for one specific case in particular. This is done in order to facilitate the understanding of their experiences as well as to provide an appropriate basis for the comparative analysis in Chapter 6. However, any statements concerning overhaul projects in general are considered in Chapter 6 as well.

Finally, the market also played a role in Company H's planning. With the switch from Windows to client-server, thus to a new technological paradigm, the company openend doors to new markets. As with the new product multi lingualism was possible, Company H could offer the product to markets of other countries (Appendix D, Company H, 65).

Overhaul strategy & approach - Redevelopment with a big-bang

When Company H switched over from Windows to client-server, the product was undertaken a rebuild (Appendix D, Company H, 14). First the company was planning only to do a restructuring of the product. However, as time went by and it was obvious that most of the technological components of the product would change, the company started redeveloping the product from scratch (Appendix D, Company H, 16). According to Interviewee 9, there was no other option available for the company. Their Windows version of the product already consisted of old-fashioned technology, which meant that just a migration to the new technological environment was impossible (Appendix D, Company H, 20).

The fact that Company H's product consisted of legacy technology back then had the consequence that there was only one approach that fitted the company's overhaul project: the big-bang approach (Appendix D, Company H, 22). Incrementally redeveloping the product was in fact not possible as the technology behind was obsolete and needed to be redeveloped completely before the company could convert the product to the new environment. Although the big-bang approach contained high risks, there was no other option available for the company.

Technological aspects - A fundamental change

The overhaul of company H's product implied fundamental change of its technological components. Besides the technological environment, there was also the functionlity of the product, which was affected. Lots of new functionality was added to the new product as the project evolved (Appendix D, Company H, 32). In the beginning considered as an opportunity, this turned out to be a realy challenge for the company in later stages of the project as there was the difficulty of setting limitations of new functionality.

The functionality however was not the biggest change the product was undertaken. To make the product fit for the superior technological environment, most of its components were renewed. The renewal concerned any technological components except for the operating system of the product, which was chosen as Microsoft Windows. Besides introducing a new layered architecture and the switch of the DBMS to a Microsoft SQL server, Company H replaced their old programming language C by C++, whereas everything was done in a .NET framework.

Temporal aspects - A tremendous delay

An overhaul with the complete redevelopment of the product's technological components is a big project. The overhaul project of Company H had a duration of five years in total (Appendix D, Company H, 44). This was however not what the company had been planning in the beginning. Initially, the overhaul project was planned to be finished within two and a half years. This means that the duration of the product doubled compared to

what was planned at first. This tremendous delay had several root causes, it was however mainly the unpredictability of such a project that lead to it. This means that there were no clear boundaries set on what the project consists of. It was for instance unclear in the beginning, which functionality would be needed in the new product as well and what functionality needed to be added in order to satisfy the customer (Appendix D, Company H, 24).

In general, the major phases of Company H's overhaul project were composed as follows:

- Project planning
- Business case
- Technology selection
- Customer feedback on technology decisions
- Redevelopment of product
- Customer interaction
- Phase out of legacy product

After an informal planning of the project, which also contained rough estimations of the financial investments in a business case, one major step before actually starting the project was the selection of new technological components. It was an important step also for the company, as for them it was very difficult to decide what technology would exist for the next ten years (Appendix D, Company H, 62). Besides thoroughly performing research, the company did involve the customer in that process. After clarifying, which technological components would be used, Company H started with the actual redevelopment of the product. In a final phase before the phase out of the current product, the company was responsible for delivering the product to the customer. This involved several steps of custoemr interaction such as the customer communication, customer trainings as well as the direct configuration of the new product at the customer.

Financial aspects - A huge investment without external funding

Company H started the project with performing some kind of business case. The financial forecasting that the company did was however rather roughly estimations (Appendix D, Company H, 63-64). As the company's product reached end of life, there was no other choice than starting the project, which is why the company decided not to perform thorough finanical planning.

The overhaul project had a duration of almost five years. In total, there were between 25 and 50 people involved. Considering these two facts, Company H had to make a huge investment to switch over to the new version of the product. Specific numbers concerning investments have not been reported by the company, as they are privately owned and thus no fincanical numbers are published. The project however was funded only by the company itself, so there was no external investor involved.

Project management & organizational aspects - Nearshoring and organizational change

Having a proper planning of the overhaul project was essential in the beginning for Company H. An important step of the planning procedure was to do research and investigation on technological components, which were thought to be used in the new product (Appendix D, Company H, 58). This entailed investigations about for instance the DBMS, layered architecture and the Microsoft.NET framework. All preselected technological components were tested as it was crucial to choose technology, which would be able to support the required performance of the product as well as which would exist for a while.

For this purpose there was a team structure set up, which was composed of having several team leaders, who supervised the developers of the product. Software developers constituted the main part of the project team, which consisted of in total 50 people after completion of the product overhaul. Not only the project itself was tremdeously delayed, but also many developers had to be added to the project team. In particular, the number of developers doubled from a number of 25 in the beginning of the project (Appendix D, Company H, 73). The reason for this increase is mainly that the project was running late as well as that there were many unpredictable circumstances that came up during the project. As the company lately realized that the big-bang approach that they applied appeared to be a huge challenge and very time-consuming, they started with some nearshoring activities (Appendix D, Company H, 77). This decision was less expensive and less difficult than hiring people in The Netherlands.

Besides adding people to the project and integrating people from the nearshoring teams, Company H has undergone some changes in the management structure. As the overhaul project turned out to be highly complex, some of the team leaders were not able to guide the project team until the end (Appendix D, Company H, 107).

Market and customer aspects - An important mission to satisfy existing customers

For Company H, the overhaul project was not only very large, but it was a real challenge to satisfy their existing customers. The problem that the company had back then was that they had to choose a big-bang approach. In the big-bang approach, the switch from one software product to the other is done at one certain point in time. Thus, it also represents a big hurdle for the customers as they have to switch over at one certain point in time as well. This was a time-consuming process, also for their customers.

Furthermore, Company H's project was delayed by more than two years in total. However, as the company had initially planned to deliver the new product earlier, most of the customers were already informed that such a big overhaul would happen in the near future. Due to the delay of the delivery of the new product, customer were about to lose their temper (Appendix D, Company H, 122). Luckily for Company H, they managed to keep most of their customers except for one (Appendix D, Company H, 125). Back then, it was truly a disaster for the company to even lose one of their customers. However, the only thing that company H could do was communicating to the customer in regular meetings as well as convincing them of the importance of the overhaul.

Challenges - Technology, Organization, Customer

Company H was facing manifold challenges during their switch from Windows to client server. The following aspects appeared to be the most challenging ones during the overhaul project of the company:

- **Maintenance of legacy technology:** While the new product was developed, Company H still had to maintain their current product as this one was still being served to their customers. The biggest challenge in this respect was to manage these two large activities at the same time. It was thus not beneficial that the company administered these two processes with the same team (Appendix D, Company H, 44).
- **Migration of functionality:** It was a challenging task to set limitations, which functionality had to be taken over to the new version of the product - and which not - in order to avoid an explosion of the product's functional size (Appendix D, Company H, 31-34). One of the countermeasures the company did was the interaction with their customers in order to define essential functionalities.
- **Technological decisions:** The company was moving to a complete new technological infrastructure, which meant the change of several technological components. To decide about these decisions was a further challenge for the company. Which technology would be valid for the following years, which technology would support a good performance best? All these type of questions were crucial from the beginning on and were faced by performing thorough research, testings and involving the customer in that decision process (Appendix D, Company H, 62).
- **Unpredictability:** Uncertainties of technology, project duration and capacity made the overhaul project unpredictable (Appendix D, Company H, 143). Thus there was a big challenge of managing those uncertainties properly in order to avoid further delay of the project.
- **Innovative pressure:** Another real challenge of the product overhaul appeared to be the pressure to deliver innovations to the customer. As the product had been advertised to the customer already, they also expected the new product to be fundamentally improved (Appendix D, Company H, 116-117).

Lessons learned & advice - A big drama

The product overhaul of Company H turned out to be *"pretty a drama"* as it was stated by the managing director of the company (Appendix D, Company H, 141). The company struggled with finishing the project in time - they ran over time by two and a half years. As the product of the company was built upon old-fashioned technology, it needed to be replaced in a big-bang. The big-bang however implied several problems for the company itself as well as for their customers. Only by means of strategic actions such as the close interaction with the customer, the loss of those could be prevented.

Company H experienced that a product overhaul can be a really negative project. Nevertheless, they could derive some central lessons:

- **Lesson 1:** It's good to go over to new technology (Appendix D, Company H, 133). Although the switch to new technology entailed a huge effort, this step was necessary to be successful in the future.
- **Lesson 2:** Managing the project properly is essential (Appendix D, Company H, 150). This includes the necessity of having a proper structure of the team as well. A team can only be managed well with an experienced team leader.
- **Lesson 3:** The lean way of approaching a product overhaul has many advantages compared to the big-bang approach. It has advantages for the development team as they are concerned with the production of smaller modules. But it is also beneficial for the delivery of the product to the market.

Chapter 6

Comparative Results Analysis

This chapter reflects on the findings of the individual case studies by means of a comparative analysis. The main goal of the analysis is to find out simililarities and differences that exist among the overhaul projects of the participating case companies. Therefore, the analysis follows the same structure as the description of the experiences of the single case studies. To provide a summary of the results that were derived by examining the case studies individually, a comparison table is introducing this chapter (cf. table 6.1). The table contains information about the overhaul type, the factors that triggered the project, the chosen overhaul strategy and approach as well as the challenges and risks that appeared during the product overhaul project.

Besides those factors, further findings such as the duration of the individual overhaul projects, the most important phases, finanical aspects, organizational aspects as well as market and customer aspects are explicated in the following sections of this chapter. Eventually, this chapter presents several models such as the *challenges-countermeasures model* and the *product software overhaul framework*.

Company	**Overhaul type**	**Triggers**	**Strategy**	**Approach**	**Challenges**	**Risks**
Company A	Cloud migration	- Legacy technology - Technological efficiency - Better market conditions	Redevelopment	Change from big-bang to hybrid	- Architecture development - Maintenance of legacy technology - Innovative pressure	- Security issues in the cloud - Change of overhaul approach - Extreme delay - Low ratio of visible benefit and effort
Company B	Web migration	- Legacy technology - Feature extension - Scarcity of qualified personnel	Redevelopment	Evolutionary approach	Backward compatibility	low ratio of visible benefit and effort
Company C	UI redevelopment	- Legacy technology - Better market conditions	Redevelopment	Big-bang	- Backward compatibility - Unpredictability	
Company D	Web migration	- Legacy technology - Technological efficiency - Better market conditions - Scarcity of qualified personnel - Organizational benefits	1-to-1 migration	Mixed approach Big-bang & hybrid	- Creation of feasible business case - Migration of huge application - Parallel activities - Unpredictability	- Disappointed customers - Phasing out current cash-cow product
Company E	UI redevelopment	- Legacy technology - Technological efficiency - Better market conditions	Redevelopment	Hybrid approach	- Unclarity of the project goal - Organizational change	- Shortage of budget - Extreme delay - Low ratio of visible benefit and effort
Company F	UI redevelopment	- Legacy technology - Better market conditions	Redevelopment	Hybrid	Architecture development	Disappointment of stakeholders due to delay
Company G	Cloud migration	- Legacy technology - Technological efficiency - Better market conditions	Redevelopment	Big-bang	- Technological decisions - Finding suitable personnel - Innovative pressure	- Delay due to missing deadline - Knwoledge drain due to loss of personnel
Company H	Migration from Windows to client-server	- Legacy technology - Technological efficiency - Personnel experience - Internationalization	Redevelopment	Big-bang	- Maintenance of legacy technology - Migration of functionality - Technological decisions - Innovative pressure	- Unpredictability - Motivation of personnel - Loss of personnel - Loss of customers - Extreme delay

Table 6.1: Comparison of the individual case companies.

6.1 The overhaul project - a variety of types

First of all, a product software overhaul is in most cases associated with a project. There is only one case that did not consider their product overhaul as a project with a specific duration and project planning involved, but rather as a constantly ongoing activity. In section 6.3, more information on that approach is revealed.

There are two major types of overhaul projects: the migration to a new technological platform and the redevelopment of the user interface (UI). More specifically, the types of a product overhaul can be distinguished as follows:

- **Client-server migration:** Client-server migration is the switch from a Windows-based product to a client-server architecture, where the product is installed on-premise at the customer (Appendix D, Company H, 14).
- **Web migration:** Web migration is the migration of a software product from the desktop/client-server paradigm to the web (Appendix D, Company B, 12). The web-based application is not installed on-premise anymore, but can be accessed by the customers online. All data is hosted by the company themselves.
- **Cloud migration:** Cloud migration has similar features as web migration, however data is not hosted by the company itself but by an external service provide (Appendix D, Company A, 32).
- **UI redevelopment:** The redevelopment of the UI is mainly concerned with changing the interface design, adapting architectural changes as well as adding functionality (Appendix D, Company C, 8). In this context, it is important to mention here that only a UI redevelopment, which leads to major changes in the architecture and to a new product version, can be considered as a software product overhaul.

In the individual case study analysis, the aforementioned types of product overhauls were mentioned. Figure 6.1 shows a percentage distribution of the different types among the case companies.

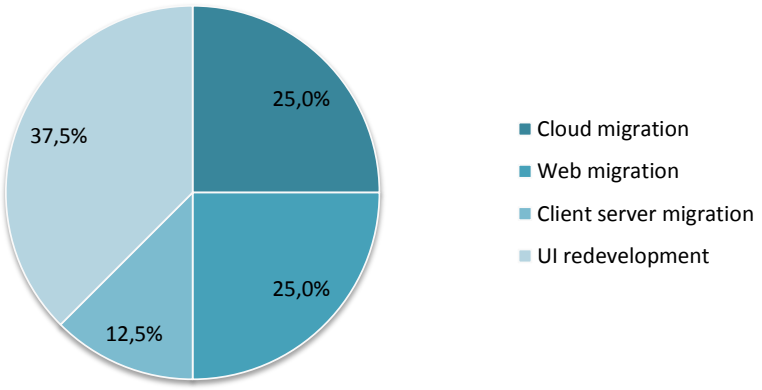


FIGURE 6.1: Overview of the types of overhaul projects.

Considering the overview, most of the companies were concerned with the migration of their product to a new technological level. A trend can be recognized in the development of online software, where the old on-premise software is replaced by a multi-tenant architecture. The software can thus be accessed by the customer online. Most of the companies chose to offer their new product as a web-based product, whereas they host the product themselves. The cloud was only mentioned once in the classical sense. Other companies argued that the term cloud is a bit vague to some extent as a product, which is offered online but hosted by the company themselves, can also be regarded as a cloud product - deployed in a private cloud (Appendix D, Company B, 62). The term cloud migration in this thesis is thus only used for the deployment in the public cloud, where the product is hosted by an external hosting provider. Besides, the redevelopment of the UI was often major part of the overhaul project. In most of the cases, the redevelopment of the UI was explained by the fact that a renewal was required by the customers or the previous UI was lacking in usability and design.

6.2 Overhaul triggers

The individual case studies revealed that there are three types of factors that trigger a product overhaul: Technological factors, organizational factors, and the market. Six major triggers can be finally illustrated as follows:

- Technological triggers
  - **Legacy technology:** The necessity to overhaul a product comes from legacy technology, which surrounds the product. This might be the architecture, the database or the programming language, which are obsolete, old-fashioned and thus not state-of-the-art technology (Appendix D, Company D, 6; Appendix D, Company B, 12). Product software companies have to adapt to new technologies in order to avoid risks such as low performance of the product (Appendix D, Company B, 10).
  - **Technological efficiency:** A further technological trigger is the technological efficiency. A product overhaul entails the advantage that the new product is delivered and deployed more efficiently (Appendix D, Company A, 12). Also it is more efficient regarding the maintenance costs of the product, which can be reduced by overhauling the product (Appendix D, Company D, 40).
- Organizational triggers
  - **Personnel requirements:** This trigger is concerned with the risk that people within or outside the company are not aware of the technology anymore, which is used by the product. First of all, developers tend to further progress in their development skills rather than sticking to obsolete technology. Moving to new technology is thus necessary for product software companies, as they can recruit experienced personnel more easily (Appendix D, Company A, 32; Appendix D, Company D, 7; Appendix D, Company H, 40).

  - **Organizational benefits:** A product overhaul involves organizational benefits as well. With the switch to a newer version of a product, the outsourcing of activities can be facilitated (Appendix D, Company D, 121). Moreover, the company itself becomes more attractive to *"young professionals"* (Appendix D, Company D, 7).

- Market and customer-related triggers

  - **Improved market conditions:** With a new product in a new technological environment, the chances to serve the current market more efficiently and to enter new markets segments are higher (Appendix D, Company H, 65). Customers and potential customers are more interested in having a product with up-to-date technology (Appendix D, Company D, 7). Furthermore, with the new generation of online software, companies can more easily deliver their product to foreign markets (Appendix D, Company A, 97).
  - **Extension of features:** Another main trigger is the feature completeness of the product (Appendix D, Company B, 30). A product overhaul is thus also triggered by the customers, who wish to have further functionality in the product.

The overview of the overhaul triggers reveals that technology is not the only relevant factor. There are several other triggers that are related to the organization itself or the market and customers. In any case, there is always the opportunity to improve the product by means of an overhaul. This might be a better performance, simplified maintainability, easier personnel recruitment or improved market conditions.

An overview of the frequency of factors that triggered the overhaul projects of the case companies is provided in figure 6.2. This overview again illustrates that technology is a main trigger, however there is also the improved market conditions, which companies see a great potential in.

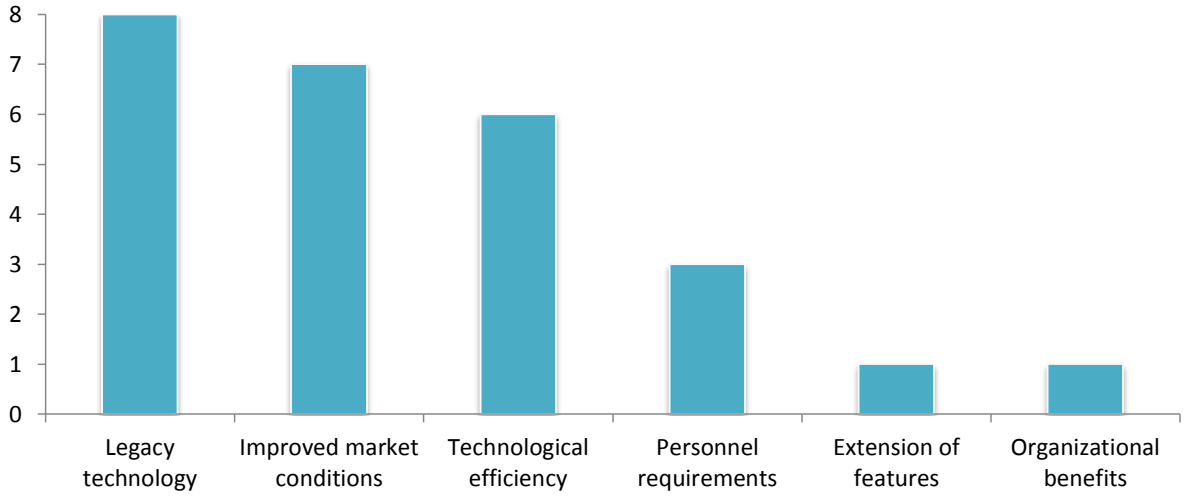


FIGURE 6.2: Overview of the triggers of overhaul projects.

6.3 Overhaul strategy & approach

The overhaul strategy was often chosen similarly by the case companies. Most of the companies were switching from an old architecture to a new one such as from client-server technology to the web. So they were migrating their product to a new technological environment, which was thought to be the better one. Their strategy however was not only to make a 1-to-1 migration of the product, but in almost every case the modules of their legacy software product were redeveloped from scratch and then migrated to the new technological environment. This means that the migration and redevelopment strategy have some kind of interwoven nature, where migration mostly entails redevelopment to some extent and the other way around. Most often a redevelopment of the software product was inevitable however, especially when the current product of the company was quite old and got complex structures (Appendix D, Company H, 57).

Although the strategy was chosen similarly, there are quite some differences in the way the overhaul projects were approached. An overview of the approaches that were chosen by the case companies is provided in figure 6.3.

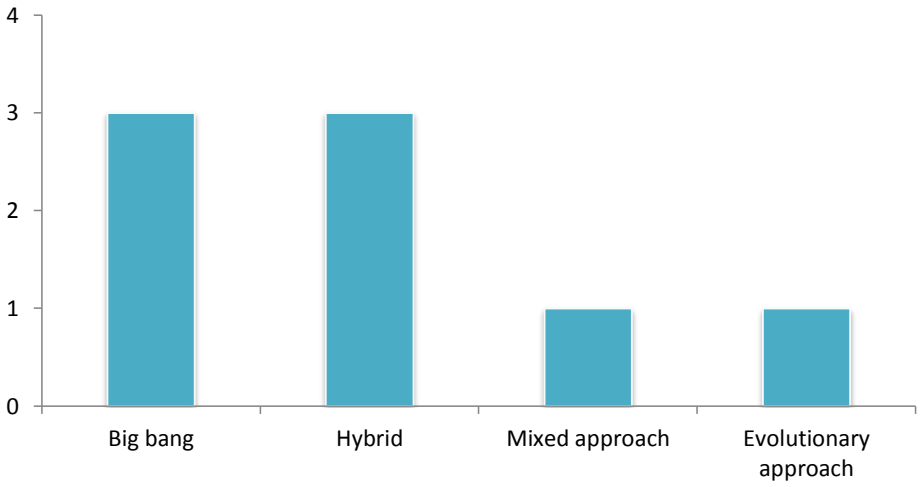


FIGURE 6.3: Overview of the approaches applied to the overhaul projects.

In general there were four different approaches the case companies applied:

- Big-bang approach
- Hybrid approach
- Mixed approach
- Evolutionary approach

The first two approaches were applied predominantly by the companies. Two cases (Company A, Company H) applied the big-bang approach for their migration to a new technological platform. Looking at the duration of their overhaul projects, there was a massive

delay in the overall project (cf. section 6.5). Company C and Company D for example applied the big-bang approach for the redevelopment of the UI of their product. In these cases, the big-bang approach was wisely chosen. The big-bang approach is rather useful for smaller projects that have a duration of approximately one year. In any other case, the hybrid approach is the one that companies recommended as it has several advantages. First of all, the development follows a more incremental method, which means that smaller modules of a product are developed and delivered step-by-step. Hence the developers, the service people as well as the customers of the company can benefit from that approach. Although these constant conversions seem to work better in practice, companies should also be aware that all these small conversions - depending on how many - can be cost intensive and hard to maintain (Appendix D, Company A, 113).

The following two statements emphasize the importance of the hybrid approach on the one hand, the risk of applying the big-bang approach on the other hand:

> *"[With the big bang approach] it was very difficult to get [the product] to our customers. So we took many more years than expected" (Appendix D, Company H, 24).*

> *"And therefore we learned that hopefully you will never ever come in the situation that you have to do a big-bang again. Because it's very dangerous for your company" (Appendix D, Company H, 48).*

The mixed approach is actually considered as a mixture of the two first mentioned approaches big-bang and hybrid. This approach was applied by case company Company D for instance. Although having one large project, they split it up in two parts, which they approached differently. The migration of the UI was done with a big-bang. This is explained by the fact that a UI cannot be replaced and released in an incremental way as it is not possible to have some parts in an outdated design, some not (Appendix D, Company D, 18). On the other hand, the company also moved their business logic to a new technological environment. This part of the project, the company approached in an incremental way, migrating module per module of the product. When looking at the project duration of Company D (cf. section 6.5), there was barely delay in their project.

Latter approach - the evolutionary approach - was only chosen by one company, however it exemplifies the multitude of different ways to approach a software product overhaul. This approach is more about the evolution of the product rather than overhauling it in a big project. In general it is similar to the hybrid strategy that is chosen by many companies. The difference is that the company is not aiming at overhauling a product in a classical sense, but they rather want to constantly improve and change the product in smaller steps.

6.4 Technological factors

During a product overhaul, companies do not only need to be aware of the fact that the functionality of the product can fundamentally change, but there are further technological parts of the product that are influenced by the software product overhaul:

- Architecture
- Development environment
- Database management system (DBMS)
- Operating system
- Programming language

Table 6.2 provides an overview of whether the individual case companies changed some of these technological components.

Company	**Architecture**	**Development environment**	**DBMS**	**Operating system**	**Programming language**
A	x	x	x		x
B	x				x
C	x				x
D	x	x	x	x	x
E	x				
F	x				x
G	x		x		x
H	x	x	x		x

Table 6.2: Overview of the technological components that were influenced by the product overhaul.

The change of a product was often triggered by the technological necessity, as already discussed in section 6.2. The components that were mostly undertaken changes are the

architecture of the product, its DBMS as well as the programming language it is written in. As some of the companies migrated to the web, a new multi-tenant architecture appeared to be crucial. Furthermore, companies tended to replace obsolete DBMS's such as Paradox or Sybase by modern SQL servers. A modern and solid DBMS is especially important when companies newly develop online software, as the security risks are higher in online environments. Besides, legacy programming languages such as Delphi, Perl, Powerbuilder or COBOL were replaced by mostly Java or C#.

6.5 Temporal dimension & major overhaul phases

In the previous argumentation it became clear that overhaul projects are characterized by a complex nature. Figure 6.4 provides an overview of the duration of each of the overhaul projects - both the duration that was planned and the real duration. Thus, any delay in the project is evident.

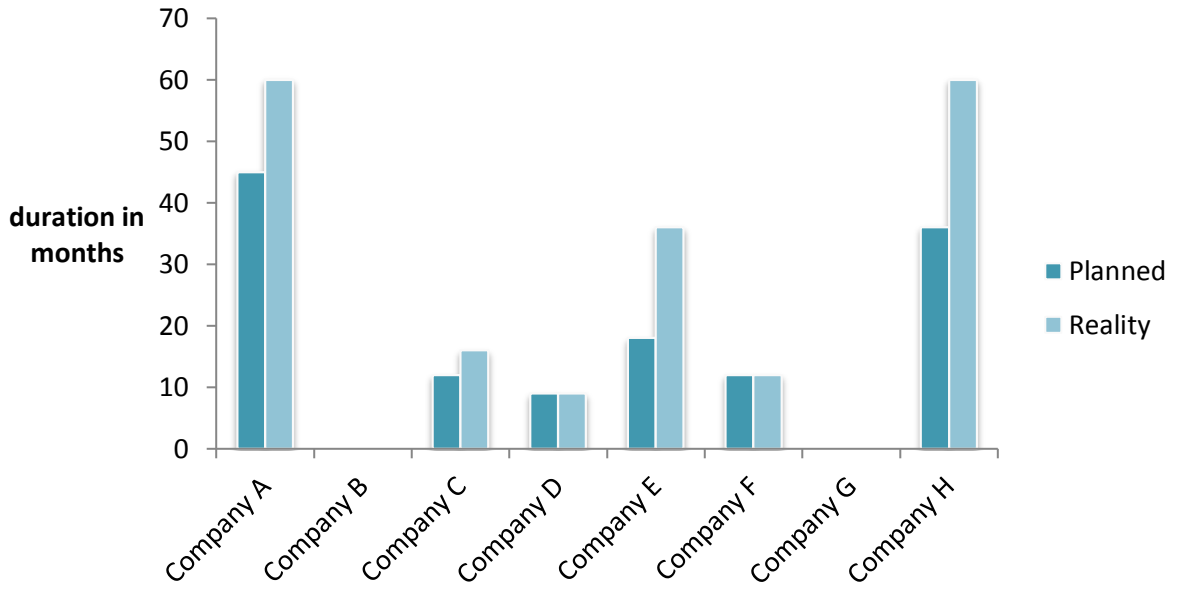


FIGURE 6.4: Planned and real duration of the overhaul projects.

> *"This project I see as pretty a drama, so it was not so positive. [...] the way we think we had to do it, no, I hope I will never meet again [...] with this big overhaul" (Appendix D, Company H, 141).*

The previous statement reveals that a software product overhaul is a big project in terms of the project duration. There was no project that took the company less than nine months. Company B as well as Company G could not specify the duration of their project exactly. On the one hand Company B does not perform overhaul projects in the classical way but rather in an evolutionary way. Thus a fundamental change that is applied to their products can also be done in a short period. On the other hand, Company

G is still currently migrating their product to the web. So far, there is no specific deadline set for finishing the project.

Interestingly, the projects of Company A and Company H were delayed the most. These are also the projects, which were originally planned with the longest duration. Furthermore, the two companies were the only companies that applied the big-bang approach for the migration to a new technological environment. In case of Company A, the high duration is explained by the fact that they switched their overhaul approach from time to time. Only by hiring experienced people, the company could limit the delay to a minimum. In case of Company E, the project duration was almost doubled after they started with it. The reason for this immense delay is that the company could not decide whether the overhaul is feasible or not. However, the company already started with planning the project and allocating resources, even if no specific decisions were made yet regarding their strategy and other relevant aspects.

The overall duration of a product overhaul process can be subdivided into different phases, which were summarized and illustrated in the following process deliverable diagram (cf. figure 6.5). The guideline on how to read the PDD can be found in section 2.2 of this thesis.

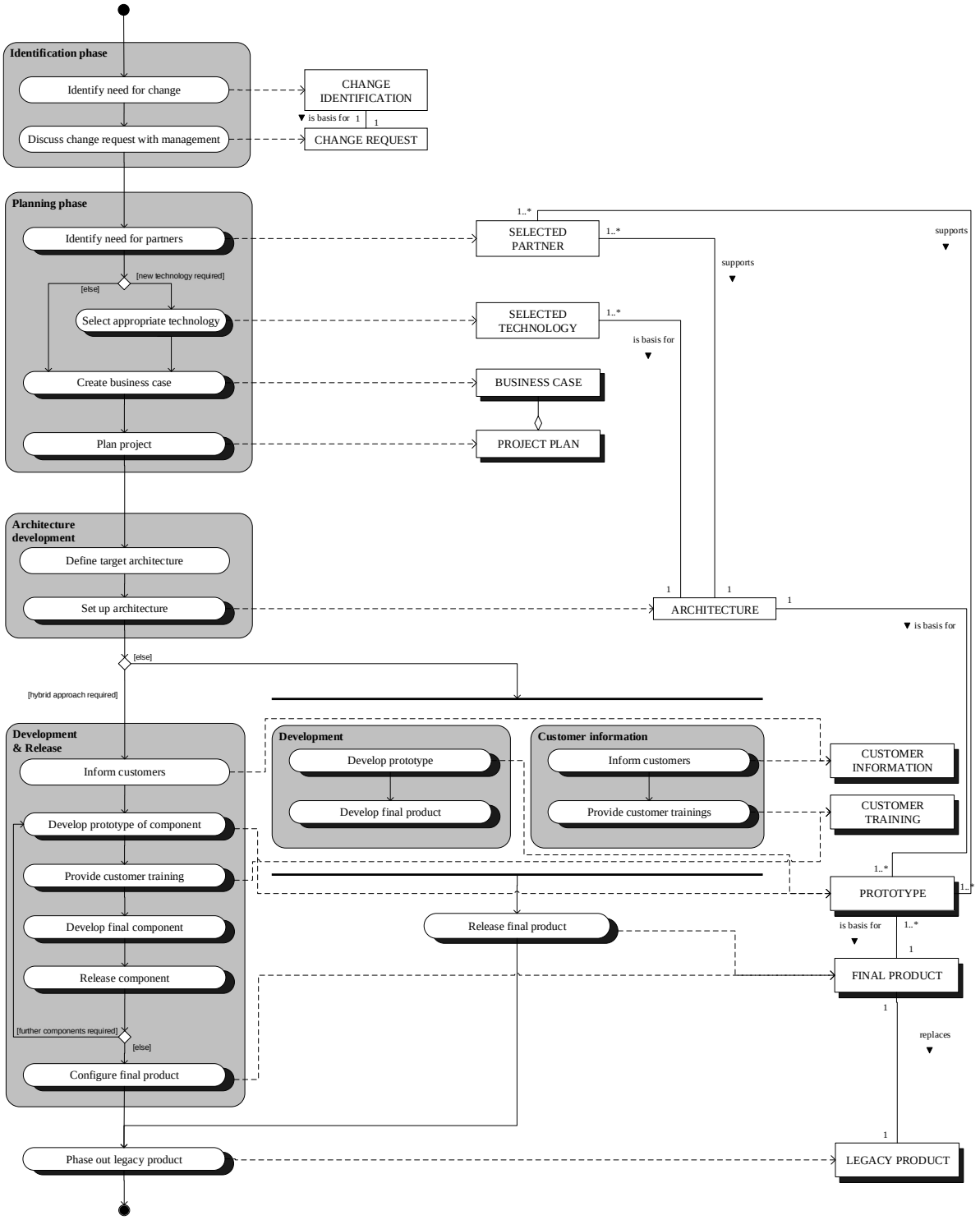


FIGURE 6.5: The process deliverable diagram of the major software product overhaul phases.

In the following, the major phases illustrated in figure 6.5 are described in more depth:[1]

Identification phase: The first step of a software product overhaul is to identify the need for change, which means that the overhaul of the product is discussed internally. In this phase, *"[...] there is a lot of discussion going on around the product internally, also with the sales people and the marketing people"* (Appendix D, Company B, 40). This need for change as well as the need for new technology should be discussed with internal and external stakeholders (Appendix D, Company D, 32).

Planning phase: Once the need for change has been confirmed, a product software overhaul needs to be thoroughly planned. This planning phase first contains the identification of possible partners of the project. In the context of a software product overhaul, it is often the case that software companies are not experienced in such processes or they lack knowledge in specific activities. Thus it is helpful to consider the support by partners. *"It's a new technology for us, so we don't have the knowledge in our company itself, so we started of with what companies are in the market that can help us with that transition"* (Appendix D, Company D, 32).

Besides the corporation with partners, the selection of appropriate technology appears to be essential for software companies before they are concerned with a product overhaul. *"We made a lot of research about what technology we will use and what will be valid for the common 10 years, then we step by step we chose these technologies"* (Appendix D, Company H, 62).

The final activity of the planning phase is constituted by creating three kinds of plans: a technical migration plan, an organizational change plan and a customer implementation plan (Appendix D, Company D, 33). One major task in this respect is furthermore the establishment of a business case, which contains any kind of information regarding financial forecasting.

Development of architecture: First step of the development of the architecture is to define the target architecture. The planning of the set-up of the architecture is defined as follows:

> *"[The architecture is] first set up by the architect, but the architect is not the one, who is putting all the stones on stones. That's what the developers are doing. It's the same process of more or less building a house. First have a good idea about the architecture and have a good fundament" (Appendix D, Company A, 154).*

Setting up the architecture is then done by *"[...] taking parts from the old one and rebuilding them in the new architecture"* (Appendix D, Company E, 49).

Development and release: After the architecture of the product is set, the further procedure depends on the overhaul approach, the company follows. When applying the hybrid approach, companies first inform their customers about the change. Then all

[1]Some of the mentioned phases might involve further activities. The purpose of the illustration however was to provide a high-level overview of the major phases, not a detailed list of any possible activity.

components of the software product are developed and finally delivered to the customer in an incremental way. When applying the big-bang approach, the final product is only delivered to the customer when finished.

An essential part of this phase is the close interaction with the customers. On the one hand, this entails the communication of the software product overhaul by providing the customers with *"[...] marketing materials, webinars, videos [...]"* (Appendix D, Company C, 54). On the other hand, the customer needs to be prepared for the new product in terms of trainings. A possible structure of such trainings was suggested as follows:

> *"For the bigger customers, there is gonna be onside training[...], for the medium customers, there is central trainings, for the small ones, there's gonna be online trainings" (Appendix D, Company E, 123).*

Eventually, the software product needs to be configured and deployed for the customer, which also lies within the area of responsibility of the software company (Appendix D, Company C, 126).

Phase out of old product: The final phase of a software product overhaul consists of phasing out the legacy product. This can be done, if every customer of the company switched to the new version of the product. Since this is a time-consuming activity, it can take companies several years.

6.6 Financial dimension, the market & the customer

> *"That's difficult because it's not only software development, but also project management, configuration, training and implementation" (Appendix D, Company F, 33).*

Generally, a software product overhaul requires big investments. However, as the statement in the beginning of this section indicates, it is diffiult for companies to provide financial estimations due to the fact that there are several activities besides the development of the new product.

In order to get software product overhaul project started, a company needs to be profitable or backed by a large shareholder (Appendix D, Company A, 95). Except from Company F, which is a rather young company, none of the projects was funded by external partners, but purely from the revenue of the company itself. This company chose the collaboration with one of their key customers as source of funding. The key customer funded the project to some extent, whereas in return he was more involved in the project than any other customer.

As some of the overhaul projects were quite complex and delayed, profitability is always important in order to avoid running out of money. That is why it is essential to concentrate on both performing the overhaul but also on maintaining the current product of a company. Most of the time, the current products of the companies were so-called

cash cows. So the company made revenues with their current products, which they could reinvest in the overhaul project. Only Company E had financial problems in the course of their project. However, the reason is not that the company was not profitable anymore, but they did not clearly decide about starting the project. This planning phase costed the company a lot of time as well as money.

> *"[. . .] our changes abroad [. . .] will become much bigger if we have our platform in the cloud computer ready" (Appendix D, Company A, 97).*

As introduced in section 6.2, companies grasped the opportunity to enter new market segments with their new product, which is also implied by the above mentioned statement. In around 75 percent, the companies targeted new customers, whereas two third of the companies were focusing on international markets. This implies that product software companies are concerned with several customer-related challenges such as the satisfaction of current customers as well as the acquisition of new customers. Companies thus need to be aware of what the challenges are and how they can be approached (cf. section 6.8). A central factor in this respect is also the close interaction and communication with the customer. An essential task for software companies during the software product overhaul is to inform their customers about the future changes and convince them of the benefits. If that is not done by software companies before and during a software product overhaul, they risk the loss of customers and turnover. This is also emphasized by the following statement:

> *"[. . .] you are losing your customers, if you say: [. . .] you are working, working, working [. . .] oh, yeah, customer, I have something new and you have to switch over [. . .] then the customer says: oh why?" (Appendix D, Company A, 105).*

6.7 Management of the organization & its personnel

Some of the case companies chose to outsource either minor or major parts of their project (cf. figure 6.6). Company A for example was outsourcing a minor part of the development activities to India, which turned out not to work well. Another example for outsourcing major activities is Company D, which was supported by an experienced consulting company in the field of software migration. In their case, the consulting company was helpful as they had been doing such projects several times before.

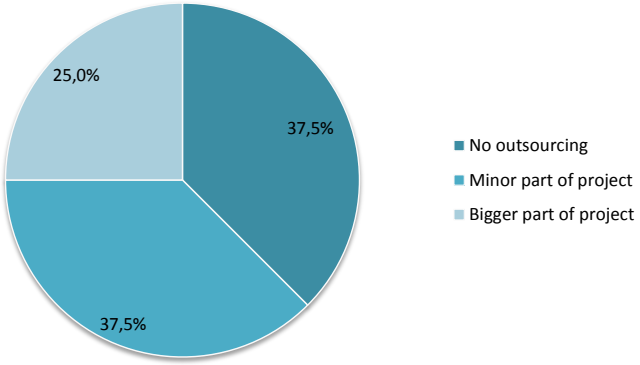


FIGURE 6.6: Overview of whether companies outsourced parts of the overhaul project.

Organizational change was furthermore a relevant topic during the overhaul project (cf. figure 6.7). A part of the companies did not have to change anything in their organizational structure. Another part however has undergone either minor or fundamental changes regarding the organizational structure. Whereas minor changes were related to changes in the development team for instance, major changes entailed the restructuring of the business unit structure of the company or the management board. The reasons for these changes were mainly linked to the fact that the company was moving towards another direction with their new product, for example a new market segemnt or a new target customer.

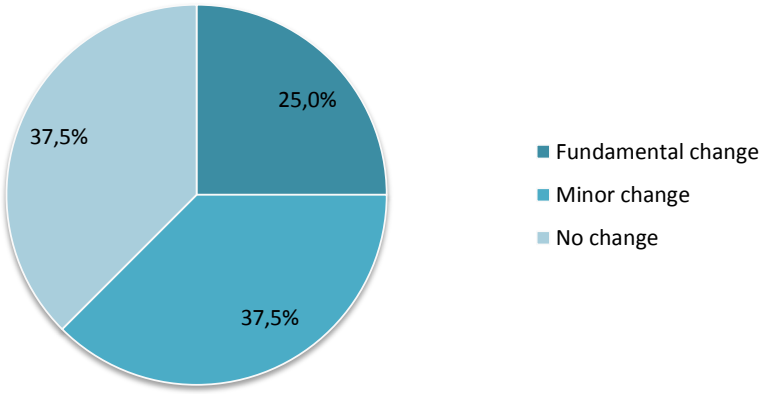


FIGURE 6.7: Overview of whether companies changed their organizational structure during the overhaul project.

Personnel management stories

Figure 6.8 shows the difference between the planned and real allocation of personnel during the product overhaul project. Company B did not specifically mention a number. In all the other case companies, the number of personnel increased. In two cases (Company A, Company H), the number of employees was even doubled, which meant a huge increase of personnel that worked on the product overhaul project.

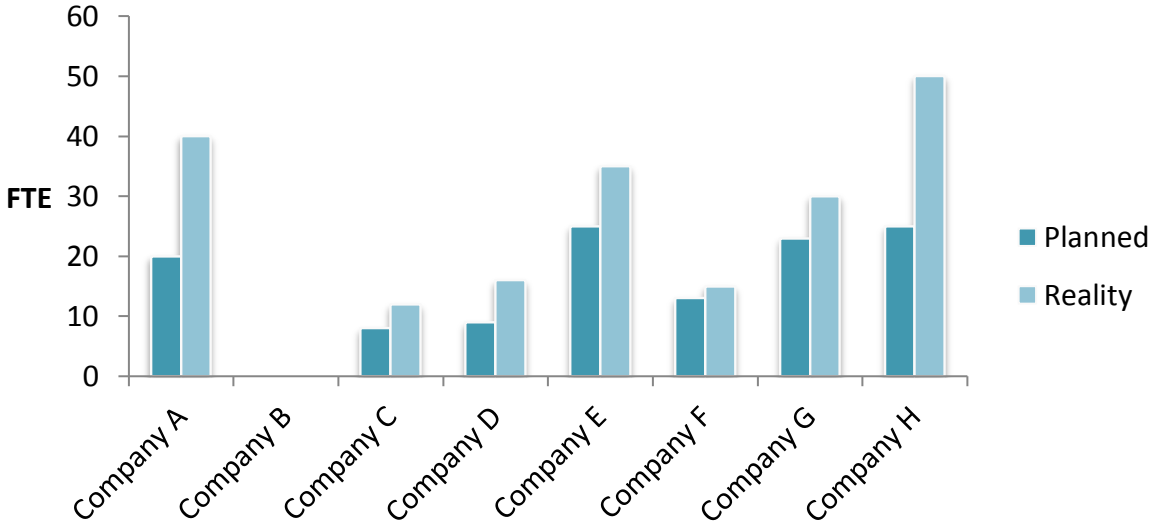


FIGURE 6.8: Overview of the planned and real allocation of personnel.

As the previous overview revealed, the management of personnel is often a challenge for product software companies during a product overhaul. Mostly, new technology comes along with the overhauled product. The consequence is that there are new challenges for software companies regarding the management of their personnel. Some of the case companies experienced some sort of "horror stories" concerning that process. These stories serve as negative examples of what can be challenging and what can go wrong during a product overhaul regarding the developers and other kind of personnel.

- **Story 1:** Company A is one example for such a "horror story". Although the company managed to succesfully finish their product overhaul, there were several hurdles to take in the course of the project. One major mistake that was done by Company A is that they started with a big-bang approach, which did not work out for the large scale migration of their product. Due to this bad decision, the company lost a lot of time and thus had to increase the pace of the project. In order to do so, the company hired extra personnel. Starting with 20 developers in the beginning, they hired further 20 developers in order to avoid a bigger delay of the project (Appendix D, Company A, 63). This way, the company could speed up their project, however they had further 20 developers to put on their payroll, which meant another huge investment for the company.

- **Story 2:** Company D cannot be considered as such a bad experience, however they faced several challenges regarding the management of their personnel. The company switched over from legacy platform called Progress to a Microsoft.NET environment. With such a big change, the developers of the company were challenged regarding their capabilities of the new technology. To test the knowledge of their employees, the company started a .NET assessment, which ended kind of disappointing for Company D (Appendix D, Company D, 109). Only a few of their employees participated in the test. Out of the small group, the company selected a

few developers, which finally became part of the project team. The company even had to hire extra personnel, as no more internal developers finalized the assessment with satisfying results. The fact that only few of their developers participated in the assessment had another consequence: what to do with them afterwards? So far, the company is still hiring those developers, as they are still busy with maintaining the old product. However, as the company plans to phase out their legacy product in the near future, they are facing a real challenge related to the management of the developer, which are not able to work with the technology of the new product.

- **Story 3:** Case company Company E faced another challenge regarding the management of personnel. This challenge was mainly driven by the fact that the company almost ran out of money. The only chance for the company to avoid bankruptcy was to fire personnel. The new operational manager that was hired back then thus fired 15 employees in about 2 months (Appendix D, Company E, 69). This represented an intense task for the new operational manager, however it was the only chance for the company to successfully finish their product overhaul.
- **Story 4:** Finally, Company H experienced another sort of "horror story". The problem for the company was that they wrongly estimated the dimension of the overhaul project. Thus also Company H doubled their personnel from 25 in the beginning to 50 in the end. The company did not have problems with developers, who were not capable of adapting to new technology. However, challenges such as the complexity and unpredictability of the project had the consequence that team leaders could not cope with the dimension of the project anymore. Thus, the company lost personnel during their project overhaul as well.

6.8 Challenges & risks

> *"I don't hope in 10 years, we have to make this overhaul again. Because otherwise we won't survive." (Appendix D, Company A, 140)*

The aforementioned statement emphasizes that a software product overhaul is a challenging endeavour, which companies do not want to execute often. There are several challenges, product software companies have to overcome, as well as risks that can occur during the product overhaul. The following section presents an overview of those challenges and risks and provides means to overcome the challenges and avoid the risks.

The challenges of a product overhaul can be categorized as follows:

- Technological challenges
- Organizational challenges
- Customer-related challenges

All of the individual challenges of the different categories are presented in a *challenges-countermeasures model*. The models are constructed in such a way that the challenges are

positioned in the center in blue boxes, surrounded by a a rectangle with a dashed frame. Some of the challenges have an affect on other challenges, which is represented by a dashed arrow. Other challenges can result in one or several risks, which is also indicated by a dashed arrow. Furthermore, the countermeasures are placed outside the rectangle with the dashed frame in green ellipses. It is indicated with green arrows, which challenges can be approached by which countermeasures in order to overcome these challenges.

Technological challenges

The technological challenges and the related countermeasures are represented by the *challenges-countermeasures model* in figure 6.9.

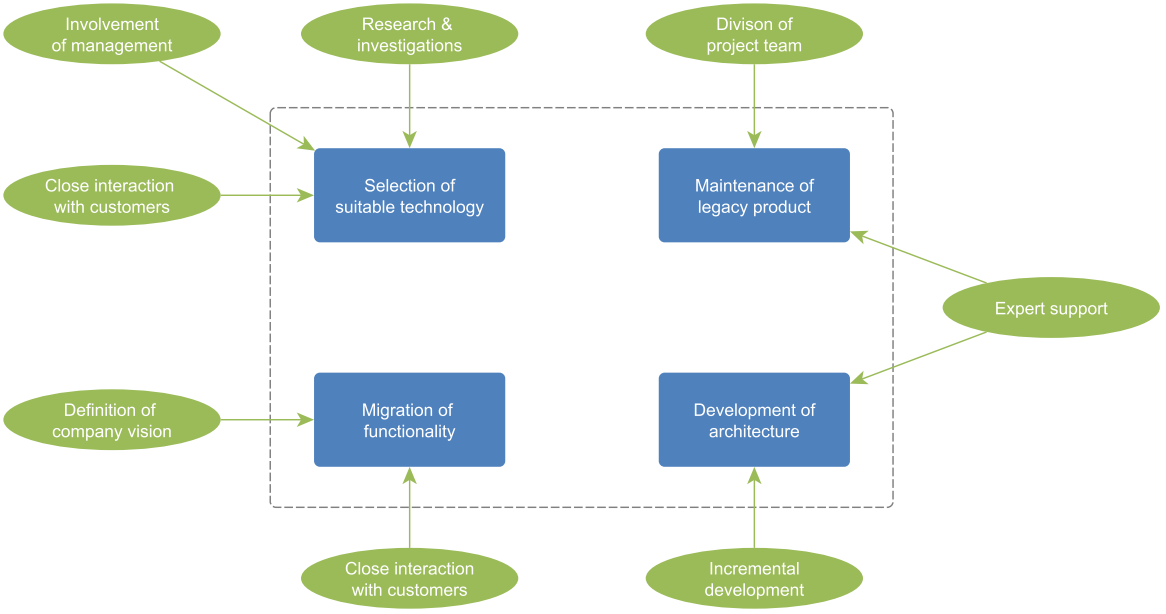


FIGURE 6.9: *Challenges-countermeasures model* for technological challenges.

- **Selection of suitable technology**
  - **Challenge:** Selecting a suitable technological framework is crucial for companies when they are concerned with the switch to a new technological environment. However, it is difficult for companies to determine, which technology suits well, which technology will last for a while and which technology will satisfy the customers' demands (Appendix D, Company H, 28).
  - **Countermeasure:** There are several countermeasures this challenge could be approached with. First of all, a company should do proper research and investigations about the preselected technology. Furthermore, the involvement of management plays an important role. It is the management's task to consolidate information about new technology from other companies. In this case, asking other companies about their opinion can be helpful. Finally, a close interaction with customers appears to be necessary. The technological decsions are not only important for the company itself, but also for its customers

as they have to adapt to the new technology as well. Thus, it is essential to interact with the customer - at least the key customers - in order to choose technology that satisfies both sides.

- **Maintenance of legacy technology**
  - **Challenge:** The challenge of maintaining the legacy product of the company is directly affected by the technological decisions they make in the beginning of the project. If the company is switching over to complete new technology, maintenance of their legacy product can be challenging (Appendix D, Company A, 77). A company needs personnel that is capable of maintaining the product, which is embedded in legacy technology. Most of the personnel in a company however is interested in working on the new product, rather than maintaining the legacy product. Until the legacy product is still in work and not phased out, it is a real challenge for software companies to find people, who are willing to work with that kind of obsolete technology.
  - **Countermeasure:** One countermeasure for this challenge would be to get external support by experts. Company A for instance hired developers, who were responsible for the maintenance of their legacy product only. Furthermore, another countermeasure is to divide the project team into groups, whereas one group is responsible for the production of the new product, the one other for the maintenance of the current version of the product (Appendix D, Company H, 44).
- **Development of architecture**
  - **Challenge:** Another essential challenge for the beginning of each overhaul project is setting up a new technological architecture. According to Company A, product software companies are generally not used such a big activity as the basic architecture has generally been set already.
  - **Countermeasure:** Again expert support such as from an experienced software architect could be helpful in this case. Furthermore, an incremental way of developing the architecture is seen as a proper countermeasure.
- **Migration of functionality**
  - **Challenge:** Companies have the challenge to set limitations of which functionality is delivered to the new version of the product.
  - **Countermeasure:** There are actually two countermeasures that could be helpful for the migration process. Primarily, the company needs to be aware of its vision. The functionality of the product depends on what the purpose of the product is and which market they want to target with the new product. Furthermore, the interaction with the customer could be helpful in this case as well as the customer can indicate, which functionality he needs and which not.

Organizational challenges

The organizational challenges and the related countermeasures are represented by the *challenges-countermeasures model* in figure 6.10.

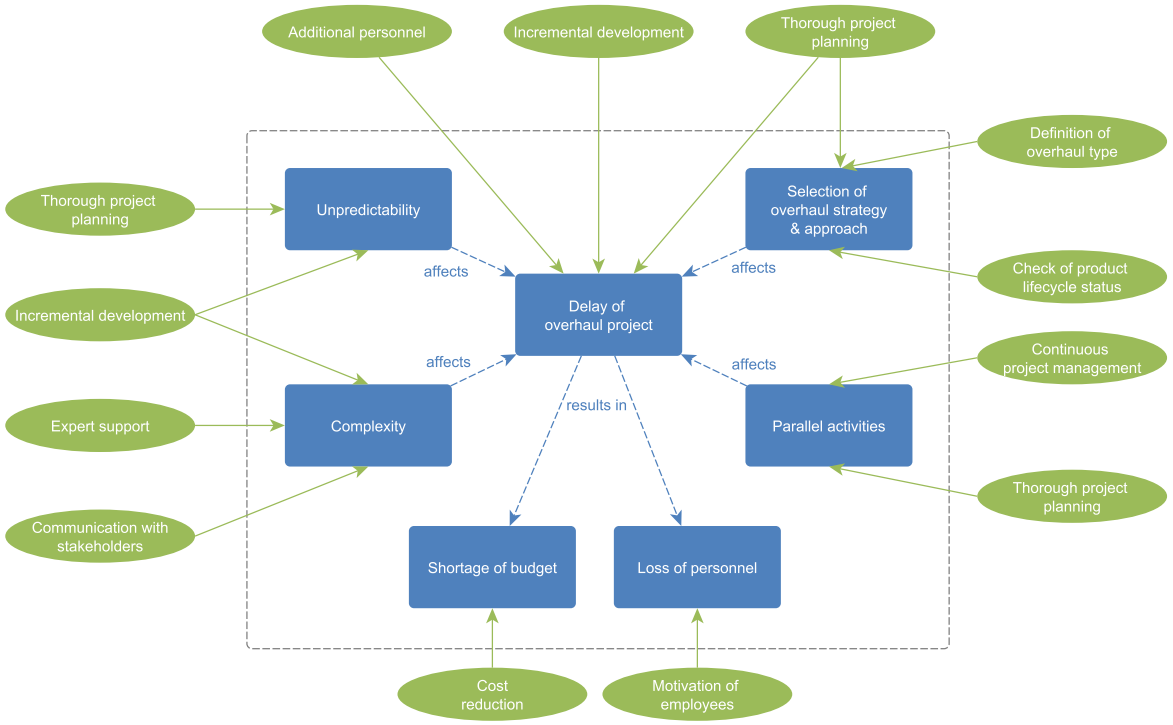


FIGURE 6.10: *Challenges-countermeasures model* for organizational challenges.

- **Unpredictability**
  - **Challenge:** An overhaul project is often a leap in the dark (Appendix D, Company A, 38). The problem with a product overhaul in general is that it is not predictable in the beginning (Appendix D, Company C, 50). There are several uncertainties such as the uncertainty about the new technology - does it work well or not -, but also the uncertainty about the time and capacity you need (Appendix D, Company H, 73).
  - **Countermeasure:** There are actually two countermeasures, with which this challenge can be approached. First of all, it is important for companies to have a clear vision of the product overhaul from the beginning on, thus planning the project thoroughly. Furthermore, the approach of incremental development contributes to overcome this challenge. By means of a step-by-step development of the product, a company sets short-term goals rather than having unpredictable long-term objectives. Thus, a huge delay in the overhaul project can be avoided as well.

- **Complexity**
  - **Challenge:** A product software overhaul is a huge step for a company (Appendix D, Company D, 142). It is considered as a project, which is hard to manage and handle for typical product software companies (Appendix D, Company A, 16). First of all it is difficult to get it running smoothly, as there are several aspects to consider such as the decison on the technological components. Secondly, there is a lot of day-to-day pressure to overcome since a company is not only occupied with the overhaul project, but has other engagements and projects as well. Finally, an overhaul project often entails new technology, a company is not experienced with.
  - **Countermeasure:** One means to overcome the complexity of an overhaul project is to hire external experts, who concentrate on those processes, a company is not experienced themselves (Appendix D, Company A, 16). Furthermore, an incremental approach of developing the new product is also beneficial as the development of smaller pieces appears to be leass complex as bigger ones. Eventually, it is crucial for product software companies to have stakeholders involved in the project. This is especially important when it comes to the planning of the overhaul project in order to eliminate any kind of uncertainties from the beginning, which could increase the complexity of the project in later phases.
- **Selection of overhaul strategy & approach**
  - **Challenge:** Another element of an overhaul project is constituted by the right selection of the strategy and the approach. As in the case of Company A and Company H, applying the hybrid approach can be highly dangerous. Whereas Company A switched over to the hybrid approach, Company H had trouble with delivering the new product to their customers. With the big-bang approach, "[. . .] it wasn't easy to bring [the product] to the customer. That took us much more time than expected [. . .]" (Appendix D, Company H, 34).
  - **Countermeasure:** The selection of the right strategy and the right approach can be challenging, however there are several methods of how to overcome this challenge. First, a company needs to be aware of the lifecycle status of their product. For younger companies, whose product is rather less advanced in the product lifecycle, the big-bang approach is rather an option than for a company with a mature product (Appendix D, Company A, 178). A mature product, which is further developed and evolved with a complex structure, is easier to overhaul when applying a hybrid approach. Moreover, a company needs to clarify the type of overhaul, they are concerned with. A redevelopment of the UI for instance is in most cases not possible to approach in a hybrid way as any interface of the UI needs to be ready with the same technology.
- **Parallel activities**
  - **Challenge:** A huge challenge for product software organizations can be described as follows: *"You're changing the boat you're sailing on while you're sailing on it"* (Appendix D, Company B, 22). Or as Company D stated: *"De*

winkel blijft tijdens de verbouwing open" (Appendix D, Company D, 21). This means that product software companies cannot close down for a couple of years to only concentrate on the overhaul project. Besides the project, there are several other activities going on in the company and the main development line is still running.

  - **Countermeasure:** Two things that could help to overcome the challenge is to plan the project properly in the beginning as well as to manage the project thoroughly during the overhaul.

- **Delay of overhaul project**

  - **Risk:** A challenge or rather a risk that is affected by the aforementioned challenges is the delay of the overhaul project.
  - **Countermeasure:** There are several countermeasures in this respect, some of which were mentioned previously (incremental development, thorough project planning, expert support). If a company is profitable enough or backed by an investor, hiring additional personnel to speed up the process also appears to be possible to avoid the delay of an overhaul project.

- **Shortage of budget**

  - **Risk:** One consequence of a delayed overhaul project is indeed the shortage of budget. Running out of money is one of the most important challenges for product software companies during an overhaul project.
  - **Countermeasure:** A means, which was used by Company E is to cut costs in terms of firing people. Another countermeasure is to get investors involved in the project.

- **Loss of personnel**

  - **Risk:** Another consequence of running over time is the risk of losing personnel. Although this was the case only once, it definitely needs to be mentioned as a major risk to avoid.
  - **Countermeasure:** As an overhaul project can be time-consuming, it can also have a demotivating effect for the employees. Thus a company needs to keep their employees motivated. Furthermore, the countermeasures that were mentioned in the previous are important to consider. Avoiding the loss of personnel is furthermore crucial in order to avoid knowledge drain of employees.

Customer-related challenges

The customer-related challenges and the related countermeasures are represented by the *challenges-countermeasures model* in figure 6.11.

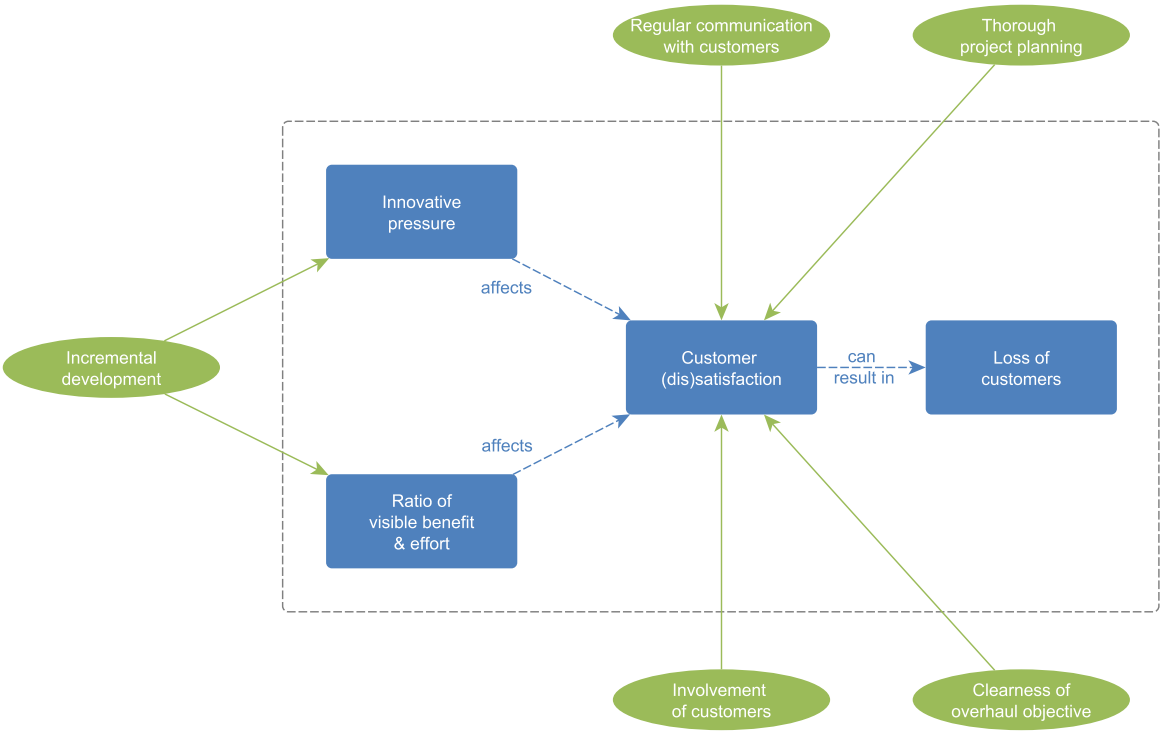


FIGURE 6.11: *Challenges-countermeasures model* for customer-related challenges.

- **Innovative pressure**
  - **Challenge:** Software companies are constantly under pressure to deliver innovations to their customers. In case of a product overhaul, the customers of the company have some expectations regarding the new product. They expect a company to fundamentally improve their product regrading aspects such as the design or performance.
  - **Countermeasure:** One reasonable countermeasure against the challenge is the hybrid approach, which means the development of the new product in a step-by-step manner. By means of that approach, the product is constantly evolved and new features and module can be delivered to the customers constantly.
- **Ratio of visible benefit & effort**
  - **Challenge:** A customer is not impressed by the new architecture the company surrounded the product with. Customers are rather interested in how well the new product is functioning compared to the old one and what kind of new functionality it has got. *"[. . .] you put 1 year in refactoring the code to make*

it more maintainable for example. You have an extremely huge expense and you have no visible benefit. I think that's the biggest challenge, if that balance between visible benefit and effort is out of balance. [...] Because there would be no improvements for the customer, no new features" (Appendix D, Company B, 45-46).

  - **Countermeasure:** Again for this challenge, an incremental way of developing the new product is beneficial. With this kind of approach, the customer receives updates of the product periodically. He is thus not concerned with a renewal, which had taken the company years to complete without having any benefit for the customer.

- **Customer (dis)satisfaction**

  - **Challenge:** A huge concern of product software companies during a product overhaul is to satisfy their customers as they are the ones, who acquire their products.
  - **Countermeasure:** The satisfaction or dissatisfaction is dependend on wehther the company successfully handles the two aforementioned challenges. Furthermore, involving the customers in activities such as the determination of the future technology or the feature set of the new product appears to be important. A company should be also clear in the beginning when communicating the project's objective to their customers. By means of a clear communication, missunderstandings are avoided from the very beginning.

- **Loss of customers**

  - **Risk:** One risk that the dissatisfaction of customers can result in, is the loss of customers, which can be considered as some kind of worst-case-scenario for a software company during a product overhaul.
  - **Countermeasure:** It is important to consider the aforementioned countermeasures.

6.9 The product software overhaul framework - A guideline for product software companies

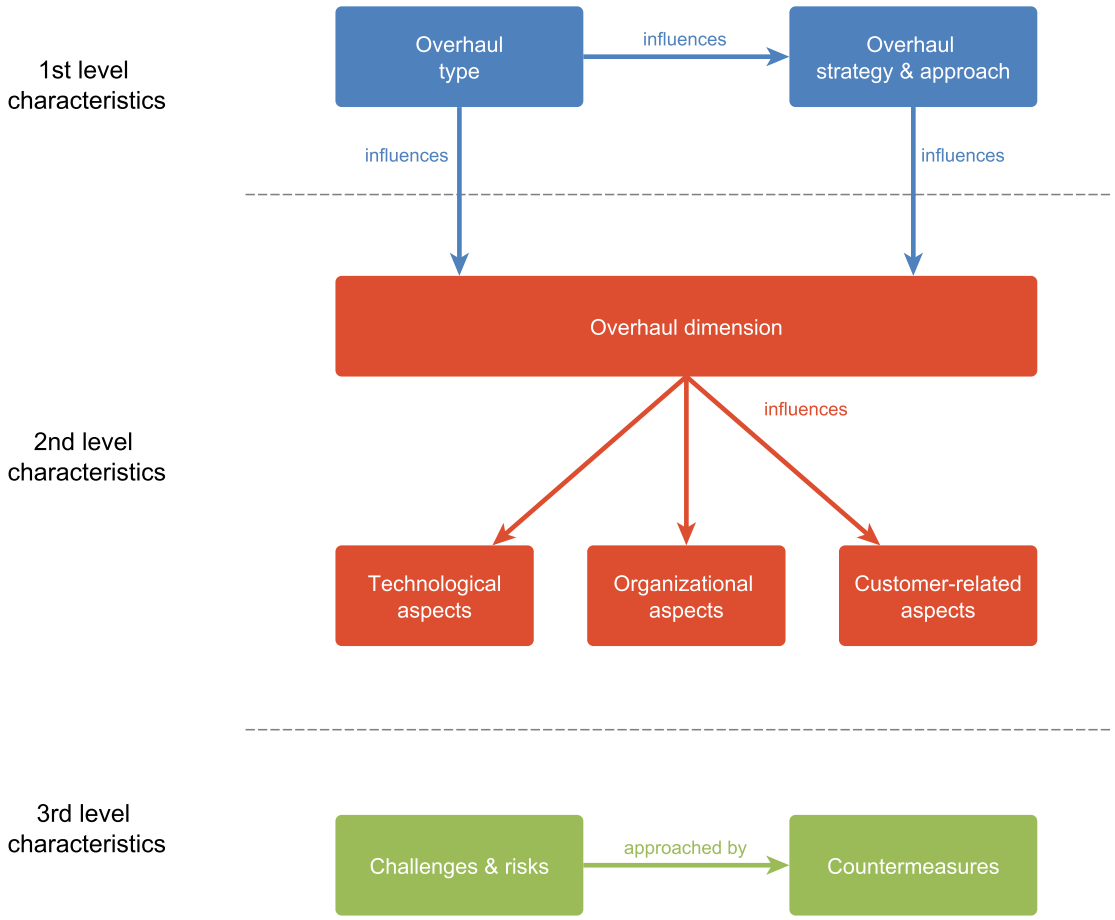


FIGURE 6.12: *The product software overhaul framework.*

The *product software overhaul framework* (cf. figure 6.12) shows that a software product overhaul has three types of characteristics. 1st level characteristics are considered as basic aspects that every product software company is concerned with, namely a specific type of product overhaul as well as a strategy to approach the overhaul. These characteristics have an influence on the 2nd level characteristics, which is primarily the overhaul dimension: duration, investment, and personnel. The dimension of such an overhaul again affects three aspects: technological aspects, organizational aspects, and customer-related aspects. Finally, the challenges that occur during a software product overhaul are considered as 3rd level characteristics. The *product software overhaul framework* constitutes some kind of guideline, which contains any aspect that should be considered by software companies before and during a product overhaul.

Chapter 7

Discussion

The central goal of this thesis was to provide a better understanding of the overhaul process in practice. In this chapter, some limitations of the research are being discussed in order to elaborate on the strentghs and weaknesses of the research. Furthermore, an outlook for future research is provided.

7.1 Limitations of research

Naturally, every research has its limitations. Thus, also this research has to be evaluated regarding its limitations, weaknesses and threats to validity. Most of the following mentioned limitations are caused by the lack of resources and time of the researcher.

One limitation of this thesis is concerned with the selection of the case companies. As many companies do not want to share much negative experiences, it was hard to get access to suitable case companies. Consequently it was not possible to select product software companies from one specific area. Generalizing any results of the research to one specific sort of companies is therefore difficult. Repeating the same research with a focus on one specific group of companies could increase the validity. Furthermore the inclusion of more case companies would increase the validity of the research. However, this was not possible due to time constraints and a lack of financial resources.

Another problem that occured during the data collection process of this research was that some of the companies were not finished with their overhaul projects yet. This influenced the data collection in such a way that some of the interviewees could not talk about their experiences regarding several aspects as they were not advanced that far in the overhaul process yet. Another issue that occured during the data collection process, more specifically the expert interviews, is that the interviewees recorded any answer from their memories. This can constitute some sort of bias, as some of the interviewee could have forgotten important information during the interview. This problem could be overcome by conducting a mixed-method research such as a case study combined with a quantitative approach in terms of a survey.

Moreover, some of the case companies are privately owned, which means that they do not publish any information about their financial situation. Thus, this thesis lacks a thorough investigation of the finanical investments and the development of revenues during a software product overhaul.

Furthermore, there was another limitation with respect to the data collection process of the research. In order to increase the validity of the research, multiple sources of evidence were used. However, in most of the cases there was barely documentation about the overhaul process available. Only one company provided a power point presentation containing information about their overhaul project. Any other kind of documentation consists of general information about the company itself, which was helpful to avoid misspellings of company specific terms.

Some other limitations are mostly caused by the fact that the research has been conducted by a single researcher. This implies limitations for the data collection process as well as the data analysis. All of the interviews, the main sources of evidence of this research, were conducted by one researcher only. This involves the possible risk of personal bias of the researcher. Having a second researcher during the interviews would increase the validity of the research as well as the reliability of the results. A second researcher could for example come up with further questions that were forgotten by the other researcher. The same goes for the coding process during data analysis.

7.2 Future research

The present thesis has presented a theoretical framework on software product overhaul processes. Besides this theoretical examination, first empirical findings about practical, real-life projects and experiences were generated. However, as the central research questions have been answered, the findings of this research have opened up new relevant fields of interests again, which could be the objective of future research.

First of all, there is the option of performing a single case study in order to investigate several parts of an overhaul process in more depth. In this context, an interesting area is constituted by the phases and activties of such a process. So far, a clear overview of each phase of the overhaul process is lacking in literature as well as in practical contexts. In a single case study, these phases could be investigated over a period of several months during an overhaul project. By means of this investigation, a method could be established, which facilitates the performance of such a process.

Besides that, another interesting topic for future research is the sustainability of such overhaul projects. One important aspect during an overhaul project is the decision about the new technology. However, many companies struggle with deciding on technological components. When making these decisions, companies have to anticipate possible future developments, so that the chosen technological components are not obsolete again after a short period of time. Thus it is a challenging task for software companies to make such farsighted decisions. Consequently it could be interesting to find out, how companies deal with the notion of sustainability in the context of a software product overhaul.

Finally, there is the option to extend the study internationally to compare practices in different countries. Besides extending it internationally, it could also be extended nationally in terms of the number of participating case companies. Another option would be to conduct the same research with product software companies that are active in the B2C-business. So far, the research has focused on product software companies that serve other companies rather than a multitude of private end-users. With such a research, one could determine similarities and differences between those two business areas.

Chapter 8

Conclusion

The major goal of the present thesis was to contribute to a better understanding of the overhaul process of product software in a practical context. To reach this goal, a multiple case study was conducted to answer the main research question, which was stated in chapter 2 as follows:

RQ: *"What are the characteristics of the major product overhaul process in software companies?"*

To solve the main research question of this research, four sub research questions were considered. Each of the four sub research questions is answered in the following.

SQ1: *"How can a product overhaul process be defined and where can it be positioned in the domain of product software?"*

To determine, where the term product overhaul can be positioned in the domain of product software, a broad overview of the software business in general was provided. In the software business, the term migration is widely used when talking about the change of software (Rouse, 2005). Many researchers apply the term migration for explaining the phenomenon of legacy system migration, which means the transition of large organizational computer systems (K. H. Bennett, 1995; Bisbal et al., 1999; Brodie & Stonebraker, 1995). Large organizational computer systems are however not the same as product software, which is a highly standardized product, manufactured by a product software company and offered to a specific market (Buxmann et al., 2011; Popp, 2011; Xu & Brinkkemper, 2005, 2007). When talking about the development of the follow-up version of such a software product, another term appeared to be necessary: the product software overhaul process. In the domain of product software, the overhaul process is closely linked to the areas of software product management, software maintenance and software evolution. A product software overhaul implies change for a software company's product. A defintion of the term software product overhaul has not been given yet. The outcomes of a multiple case study however have shown that product software companies agree with the selection of this term when talking about a fundamental change of software product.

Figure 8.1 illustrates, which overhaul types have been revealed by the software companies that participated in this research.

FIGURE 8.1: Overview of the types of a software product overhaul.

The figure emphasizes that a software product overhaul is not only concerned with the migration of a software product to a new technological environment, but also with the redevelopment of certain components such as the UI. In all cases, an overhaul was charaterized as a process that can lead to major changes - for the product, for the organization as well as for the customers, who use the product.

Considering the aforementioned facts, a first definition of the software product overhaul process can be formulated as follows:

> *"A software product overhaul can have different natures. It can mean the fundamental redevelopment of a product entailing all kinds of architectural changes such as the database management system or the programming language. Besides, it can also mean the migration of an application to a new technological environment. Generally the sum of all processes that are related to a major change of a current software product that leads to a new version of that software product - while keeping its basic functionality - is referred to as a software product overhaul."*

SQ2: *"What are the reasons for product software companies to fundamentally overhaul their products?"*

It was important to find out why companies aim to overhaul their software products in the first place. In literature as well as practice, several reasons were revealed. Besides Haines (2009), Jansen et al. (2011) categorized such reasons into product strategy, platform changes as well as portfolio decisions. The reasons, mentioned in literature, are presented in figure 8.2:

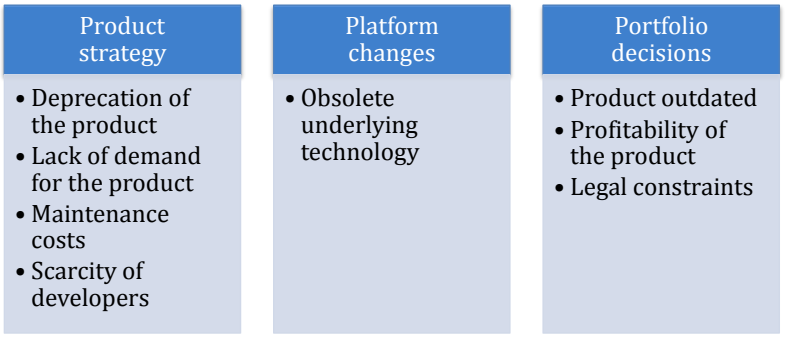


FIGURE 8.2: Overview of the triggers of overhaul projects in theory.

In comparison to those reasons, the major triggers mentioned by the case companies are listed in figure 8.3.

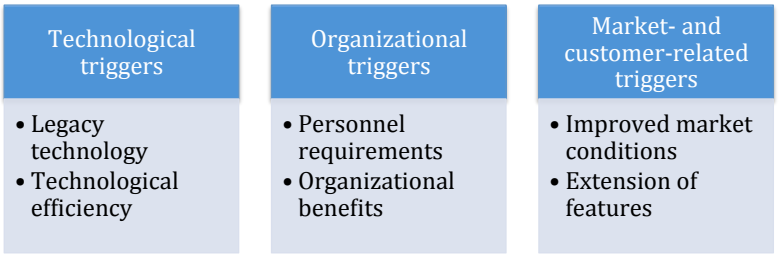


FIGURE 8.3: Overview of the triggers of overhaul projects in practice.

None of the investigated case companies started their overhaul project without having any specific reason. In comparison to what is illustrated by scientific literature, the overhaul project in companies was barely triggered by portfolio decisions. One factor that was often mentioned by practitioners, but was not specifically named by academic literature, is the trigger *improved market conditions*.

SQ3: *"What are the strategies that product software companies apply to approach the overhaul process?"*

Generally there were three strategies named by academia and two ways of how to approach an overhaul project. There are the strategies migration, wrapping as well as redevelopment from scratch (Bisbal et al., 1999; Brodie & Stonebraker, 1995; Gimnich & Winter, 2005). Besides, a product overhaul can be approached by a big-bang approach or a smooth migration.

Only two of the three proposed strategies were commonly applied by software companies, namely migration and the redevelopment from scratch. Most often the strategy of the companies was not a 1-to-1 migration of the product, but they rather redeveloped modules of the product from scratch, which they then migrated to the new technological environment. The migration and redevelopment strategy mostly have an interwoven nature, as migration entails redevelopment to some extent and the other way around.

Regarding the overhaul approach, science and practice revealed similar findings. The two most widely applied approaches of the case companies were the big-bang approach as well as the smooth migration, which was commonly referred to as the hybrid approach. Besides those two approaches, a third approach was mentioned by one company, namely the evolutionary approach. This approach is similar two the hybrid approach, however it rather aims to improve the product incrementally without setting up a large overhaul project.

The selection of a suitable strategy and approach was actively discussed amongst the case companies. Clearly selecting the right overhaul strategy and approach was crucial to product software companies. Whereas the big-bang approach was considered to be rather risky and only suitable for smaller projects, the hybrid approach was prefered by companies as it entails several advantages, particularly regarding both development and customer communication.

SQ4: *"What aspects affect the product overhaul process in software companies and what are the most significant challenges?"*

The overhaul process was affected by the following major aspects:

- **Technology:** In most cases, the overhaul led to an extension of the functionality of the product. In a few cases, the extension of functionality and feature completeness was even considered as a major trigger of the overhaul process. Besides the functionality of the product many parts of the technological environment were affected by the overhaul of the product: the architecture, the development environment, the DBMS, the operating system as well as the programming language. Software companies tended to replace obsolete technology by state-of-the-art technology. The architecture for example was often affected by the software product overhaul as companies migrated their products to a new technological environment such as the cloud, which required a different architectural setting than before.
- **Temporal aspects:** The product overhaul process appeared to be a complex and time-consuming project. Except for one case company, the overhaul project was

always delayed. In some cases the originally planned duration was even doubled. The reasons for such a delay were manifold. On the one hand, short delays happened as the company did not specify concrete goals for the overhaul project from the beginning on. On the other hand, the product overhaul project turned out to be tremendously delayed. Such delays could not simply be explained by inaccurate planning and management of the project. A main factor in this respect was the selection of the overhaul approach. The big-bang approach for example implied several problems for those companies, which decided to redevelop their product first and to migrate it to a new technological level. One example for such a problem was that the customers struggled with the project, as they were concerned with a big change all of a sudden as well.

- **Financial aspects:** The overhaul process mostly required large investments. As the product overhaul was mostly initiated as a huge project, it was a real challenge for companies not to run out of money. Only profitable companies or companies that were supported by external funding could overcome this challenge. Another financial aspect to consider was the business or revenue model. When migrating from client-server technology to the web, companies usually changed their business and delivery model. This involved a change from the classic license/maintenance model to SaaS-solutions with subscription based revenue model.

- **Organizational aspects:** Organizational change was often a consequence of the software product overhaul. There is a difference between minor organizational change, such as the addition of personnel, and major organizational change. Such major changes involved the restructuring of the business unit structure or even of the management board.

- **Market and customer aspects:** The customer played an important role during the overhaul process. Companies had two major challenges during the overhaul process regarding the customers. First, companies had to overhaul their products while they still served their current customers with the old solution. Furthermore, they needed to communicate to their customers and convince them of the new version of the product. At the same time, companies tended to enter new market segments, which forced them to satisfy both current customers as well as prospective ones.

Challenges

A product overhaul project is highly complex and time-consuming. Thus, by definition there were several challenges that came along with the software product overhaul. The challenges of a product overhaul can be divided into three categories (cf. figure 8.4).

FIGURE 8.4: Overview of the challenges of overhaul projects in practice.

The overview of the challenges reveals that the different categories are interrelated to each other. The migration of functionality for instance, which is considered as a technological challenge, is also affecting the satisfaction level of customers. Satisfying customers is also easier, if the overhaul project is not heavily delayed.

All challenges that can appear during a product overhaul as well as the countermeasures that can be used to approach those challenges were presented in *challenges countermeasures models* (cf. section 6.8). These models shall serve as a source of inspiration for product software companies, which are concerned with the overhaul of their product.

RQ: *"What are the characteristics of the major product overhaul process in software companies?"*

The *product software overhaul framework* (cf. section 6.9) showed that a software product overhaul generally has three types of characteristics. 1st level characteristics are considered as basic aspects that every product software company is concerned with, namely a specific type of product overhaul as well as a strategy to approach the overhaul. These characteristics have an influence on the 2nd level characteristics, which is primarily the overhaul dimension: duration, investment, and personnel. The dimension of such an overhaul again affects three aspects: technological aspects, organizational aspects, and customer-related aspects. Finally, the challenges that occur during a software product overhaul are considered as 3rd level characteristics. The *product software overhaul framework* constitutes a guideline, which contains any aspect that should be considered by software companies before and during a product overhaul.

All in all, this research provided insights of the product overhaul process in software companies. The empirical results show that the product overhaul of software is a challenging and time-consuming activity that most companies consider as a major project. Although

technology is one of the main factors that affects a product overhaul, it is not the only one. There is a multitude of organizational challenges and customer-related challenges, a product software company needs to overcome in order to avoid a shortage of budget, running over time or the loss of personnel and customers. This research reflected on these aspects and opens up a path into further research about the phenomenon of software product overhaul processes.

References

Andrikopoulos, V., Binz, T., Leymann, F., & Strauch, S. (2013). How to Adapt Applications for the Cloud Environment. *Computing*, *95*(6), 493–535.

April, A., Hayes, J. H., Abran, A., & Dumke, R. (2005). Software Maintenance Maturity Model (SM mm): The software maintenance process model. *Journal of software maintenance and evolution: Research and Practice*, *17*(3), 197–223.

Artz, P., Weerd, I. V. D., Brinkkemper, S., & Fieggen, J. (2010). Productization : transforming from developing customer-specific software to product software. In *Proceedings of the 1st international conference on software business (icsob 2010)* (pp. 90–102).

Bekkers, W., Weerd, I. V. D., Spruit, M., & Brinkkemper, S. (2010). A Framework for Process Improvement in Software Product Management. In *R. o'connor, s. tichkiewitch, & r. messnarz (eds.), systems, software and services process improvement* (pp. 1–12). Springer, Berlin-Heidelberg.

Benbasat, I., Goldstein, D. K., & Mead, M. (1987). The Case Research Strategy in Studies of Information Systems. *MIS quarterly*, *11*(3), 369–386.

Bennett, K., Rajlich, V. T., & Wilde, N. (2002). Software evolution and the staged model of the software lifecycle. *Advances in Computers*, *56*, 1–54.

Bennett, K. H. (1995). Legacy Systems : Coping with Success. *Software, IEEE*, *12*(1), 19–23.

Bennett, K. H., & Rajlich, V. T. (2000). Software Maintenance and Evolution : A Roadmap. In *Proceedings of the conference on the future of software engineering* (pp. 73–87).

Berander, P., & Andrews, A. A. (2005). Requirements Prioritization. In *A. aurum & c. wohlin (eds.), engineering and managing software requirements* (pp. 69–94). Springer, Berlin-Heidelberg.

Bezemer, C. P., & Zaidman, A. (2010). Multi-Tenant SaaS Applications : Maintenance Dream or Nightmare? In *Proceedings of the joint ercim workshop on software evolution (evol) and international workshop on principles of software evolution (iwpse)* (pp. 88–92).

Bisbal, J., Lawless, D., & Grimson, J. (1999). Legacy information systems: issues and directions. *IEEE Software*, *16*(5), 103–111.

Brodie, M. L., & Stonebraker, M. (1995). *Migrating legacy systems: gateways, interfaces & the incremental approach*. Morgan Kaufmann Publishers Inc., San Francisco.

Buxmann, P., Diefenbach, H., & Hess, T. (2011). *Die Softwareindustrie: Ökonomische Prinzipien, Strategien, Perspektiven*. Springer, Berlin-Heidelberg.

Carlshamre, P., & Regnell, B. (2000). Requirements lifecycle management and release planning in market-driven requirements engineering processes. In *Proceedings of the 11th international workshop on database and expert systems applications* (pp. 961–965).

Carrière, S. J., Woods, S., & Kazman, R. (1999). Software architectural transformation. In *Sixth working conference on reverse engineering* (pp. 13–23).

Chan, F., Chan, M., & Tang, N. (2000, November). Evaluation methodologies for technology selection. *Journal of Materials Processing Technology*, *107*(1), 330–337.

Corbin, J. M., & Strauss, A. (1990). Grounded Theory Research : Procedures , Canons , and Evaluative Criteria. *Qualitative sociology*, *13*(1), 3–21.

Cordy, J. R. (2006). The TXL source transformation language. *Science of Computer Programming*, *61*(3), 190–210.

Correia, R., Matos, C., Heckel, R., Koutsoukos, G., & Andrade, L. (2006). Software Reengineering at the Architectural Level: Transformation of Legacy Systems. *Department of Computer Science, University of Leicester, UK*.

Cusumano, M. A. (2003). Company Character and the Software Business. *Communications of the ACM*, *46*(10), 21–23.

Cusumano, M. A. (2004). *The Business of Software: What Every Manager, Programmer, and Entrepreneur Must Know to Thrive and Survive in Good Times and Bad*. Free Press, New York.

Cusumano, M. A. (2010, April). Cloud computing and SaaS as new computing platforms. *Communications of the ACM*, *53*(4), 27–29.

Darke, P., Shanks, G., & Broadbent, M. (1998, October). Successfully completing case study research: combining rigour, relevance and pragmatism. *Information Systems Journal*, *8*(4), 273–289.

D'souza, A., Kabbeldijk, J., Seo, D., Jansen, S., & Brinkkemper, S. (2012). Software-As-A-Service : Implications For Business And Technology In Product Software Companies. In *S. pan & t. cao (eds.), proceedings of the 16th pacific asia conference on information systems (pacis 2012).* Ho Chi Minh, Vietnam.

Dubey, A., & Wagle, D. (2007). Delivering software as a service. *The McKinsey Quarterly*, *May 2007*.

Ebert, C. (2007, June). The impacts of software product management. *Journal of Systems and Software*, *80*(6), 850–861.

Ebert, C. (2014). Software product management. *Software, IEEE*, *31*(3), 21–24.

Eisenhardt, K. M. (1989). Building theories from case study research. *Academy of management review*, *14*(4), 532–550.

Flick, U. (2009). *Sozialforschung: Methoden und Anwendungen*. Rowohlt Taschenbuch Verlag, Reinbek bei Hamburg.

Fois, S., & Lysonick, R. (2012). *Analyzing the Global Software industry: Trends, challenges and evolution in the business model* (Tech. Rep. No. October). Sia partners, Paris.

Forite, L., & Hug, C. (2014). FASMM : Fast and Accessible Software Migration Method. In *Proceedings of the 8th international conference on research challenges in information science (rcis 2014)* (pp. 1–12).

Fricker, S. A. (2012). Software Product Management. In *A. maedche, a. botzenhardt, & l. neer (eds.), software for people: Fundamentals, trends and best practices* (pp. 53–81). Springer, Berlin-Heidelberg.

Gimnich, R., & Winter, A. (2005). Workflows der Software-Migration. *Softwaretechnik-Trends*, *25*(2), 22–24.

Gorchels, L. (2006). *The product manager's handbook: the complete product management resource*. McGraw-Hill, New York.

Guo, C. J., Sun, W., Huang, Y., Wang, Z. H., & Gao, B. (2007). A framework for native multi-tenancy application development and management. In *Proceedings of the international conference on e-commerce technology (cec) & the international conference on enterprise computing, e-commerce, and e-services (eee)* (pp. 551–558).

Haines, S. (2009). *The product manager's desk reference*. McGraw-Hill, New York.

Henderson, B. D. (1973). *The Experience Curve - Reviewed; IV. The Growth Share Matrix or The Product Portfolio* (Tech. Rep.). The Boston Consulting Group. Retrieved from `http://www.bcg.com/documents/file13904.pdf`

Hietala, J., Kontio, J., Jokinen, J.-p., & Pyysiäinen, J. (2004). Challenges of Software Product Companies : Results of a National Survey in Finland. In *Proceedings of the 10th international symposium on software metrics* (pp. 232–243).

Hoch, D. J., Roeding, C. R., Punkert, G., Lidner, S. K., & Muller, R. (2000). *Secrets of software success: management insights from 100 software firms around the world*. Harvard Business Press, Boston.

Hunold, S., Korch, M., Krellner, B., Rauber, T., Reichel, T., & Rünger, G. (2008). Transformation of Legacy Software into Client/Server Applications through Pattern-Based Rearchitecturing. In *32nd annual ieee international conference on computer software and applications (compsac'08)* (pp. 303–310). Ieee.

IEEE. (1998). *IEEE Std. 1219: Standard for Software Maintenance* (Vol. 1998). IEEE Computer Society Press, Los Alamitos.

Jalali, S., & Wohlin, C. (2012). Systematic literature studies: database searches vs. backward snowballing. In *Proceedings of the acm-ieee international symposium on empirical software engineering and measurement* (pp. 29–38).

Jamshidi, P., Ahmad, A., & Pahl, C. (2013). Cloud Migration Research: A Systematic Review. *IEEE Transactions on Cloud Computing*.

Jansen, S., Ballintijn, G., & Brinkkemper, S. (2005). A Process Model and Typology for Software Product Updaters. In *Proceedings of the ninth european conference on software maintenance and reengineering (csmr 2005)* (pp. 265–274). Ieee.

Jansen, S., & Brinkkemper, S. (2006a). Definition and Validation of the Key process of Release , Delivery and Deployment for Product Software Vendors : turning the ugly duckling into a swan. In *Proceedings of the 22nd ieee international conference on software maintenance (icsm'06)* (pp. 166–175).

Jansen, S., & Brinkkemper, S. (2006b). Ten misconceptions about product software release management explained using update cost/value functions. In *International workshop on software product management (iwspm'06)* (pp. 44–50).

Jansen, S., & Brinkkemper, S. (2008). Applied multi-case research in a mixed-method research project: Customer configuration updating improvement. *Information Systems Research Methods, Epistemology and Applications*, 1–29.

Jansen, S., Brinkkemper, S., & Finkelstein, A. (2009). Business Network Management as a Survival Strategy: A Tale of Two Software Ecosystems. In *Proceedings of the 1st international workshop on software ecosystems* (pp. 34–48).

Jansen, S., Popp, K. M., & Buxmann, P. (2011). The Sun also Sets : Ending the Life of a Software Product. In *Software business* (pp. 154–167). Springer, Berlin-Heidelberg.

Karlsson, J. (1996). Software requirements prioritizing. In *Proceedings of the second international conference on requirements engineering* (pp. 110–116).

Karlsson, J., & Ryan, K. (1997). A cost-value approach for prioritizing requirements. *Software, IEEE*, *14*(5), 67–74.

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

Khadka, R., Batlajery, B. V., Saeidi, A. M., Jansen, S., & Hage, J. (2014). How do professionals perceive legacy systems and software modernization? In *Proceedings of the 36th international conference on software engineering (icse 2014)* (pp. 36–47).

Khadka, R., Reijnders, G., Saeidi, A., Jansen, S., & Hage, J. (2011, September). A method engineering based legacy to SOA migration method. In *27th ieee international conference on software maintenance (icsm)* (pp. 163–172).

Khadka, R., Saeidi, A., Idu, A., Hage, J., & Jansen, S. (2012). Legacy to SOA Evolution : A Systematic Literature Review. In *A. d. ionita, g. lewis & m. litoiu (eds.), migrating legacy applications: Challenges in service oriented architecture and cloud computing environments: Igi global (in press).*

Khadka, R., Saeidi, A., Jansen, S., Hage, J., & Haas, G. P. (2013). Migrating a Large Scale Legacy Application to SOA : Challenges and Lessons Learned. In *Proceedings of the working conference on reverse engineering (wcre)* (pp. 425–432).

Kitchenham, B. (2004). Procedures for Performing Systematic Reviews. *Keele, UK, Keele University2*, *33*.

Kittlaus, H., & Clough, P. (2009). *Software product management and pricing: Key success factors for software organizations.* Springer, Berlin-Heidelberg.

Kontio, J., Jokinen, J.-p., Mäkelä, M. M., & Leino, V. (2005). Current Practices and Research Opportunities in Software Business Models. *ACM SIGSOFT Software Engineering Notes*, *30*(4), 1–4.

Krebs, J. J. (2008). *Agile Portfolio Management.* Microsoft Press, Redmond.

Lehman, M. M., & Ramil, J. F. (2003). Software Evolution – Background, Theory, Practice. *Information Processing Letters*, *88*(1), 33–44.

Lethbridge, T. C., Sim, E. S., & Singer, J. (2005). Studying Software Engineers : Data Collection Techniques for Software Field Studies. *Empirical Software Engineering*, *10*(3), 311–341.

Lippoldt, D., & Stryszowski, P. (2009). *Innovation in the Software Sector.* Organisation for Economic Co-operation and Development.

Maimbo, H., & Pervan, G. (2005). Designing a case study protocol for application in IS research. In *P. chau (ed.), proceedings of the ninth pacific asia conference on information systems (pacis'05), hong kong* (pp. 1281–1292).

Malton, A. J. (2001). The Software Migration Barbell. In *Proceedings of the aserc workshop on software architecture.*

Natt och Dag, J., Regnell, B., Gervasi, V., & Brinkkemper, S. (2005). A linguistic-engineering approach to large-scale requirements management. *Software, IEEE*, *22*(1), 32–39.

O'Connor, R. V. (2012). Using Grounded Theory Coding Mechanisms to Analyze Case study and Focus Group Data on the Context of Software Process Research. In

M. mora, o. gelman, a. steenkamo, & m.s. raisinghani (eds.), research methodologies, innovations and philosophies in software systems engineering and information systems.

OECD. (2002). *Highlights of the OECD Information Technology Outlook 2002* (Tech. Rep.). The Organisation for Economic Co-operation and Development.

Ojala, A., & Tyrväinen, P. (2007, February). Business models and market entry mode choice of small software firms. *Journal of International Entrepreneurship*, *4*(2-3), 69–81.

Osterwalder, A., Pigneur, Y., & Tucci, C. L. (2005). Clarifying business models: Origins, present, and future of the concept. *Communications of the Association for Information Systems*, *16*(1).

Pohl, K., Böckle, G., & Van der Linden, F. (2005). *Software Product Line Engineering: foundations, principles and techniques.* Springer, Berlin-Heidelberg.

Popp, K. M. (2011, July). Software Industry Business Models. *IEEE Software*, *28*(4), 26–30.

Rajala, R., Rossi, M., & Tuunainen, V. K. (2003). A Framework for Analyzing Software Business Models. In *Proceedings of the 11th european conference on information systems engineering (ecis 2003)* (pp. 1614–1627).

Rajlich, V. (2014). Software evolution and maintenance. In *Proceedings of the future of software engineering (fose 2014)* (pp. 133–144).

Regnell, B., & Brinkkemper, S. (2005). Market-driven requirements engineering for software products. In *A. aurum & c. wohlin (eds.), engineering and managing software requirements* (pp. 287–308). Springer, Berlin-Heidelberg.

Rivera, J., & van der Meulen, R. (2014). *Gartner Says Worldwide IT Spending on Pace to Reach $ 3 . 8 Trillion in 2014.* Retrieved 17.04.2014, from `http://www.gartner.com/newsroom/id/2643919`

Rouse, M. (2005). *Definition: Migration.* Retrieved from `http://searchcio.techtarget.com/definition/migration`

Ruhe, G., Eberlein, A., & Pfahl, D. (2002). Quantitative WinWin – A New Method for Decision Support in Requirements Negotiation. In *Proceedings of the 14th international conference on software engineering and knowledge engineering (seke 2002)* (pp. 159–166).

Runeson, P., & Höst, M. (2008, December). Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering*, *14*(2), 131–164.

Sawyer, S. (2000, March). Packaged software: implications of the differences from custom approaches to software development. *European Journal of Information Systems*, *9*(1), 47–58.

Sawyer, S. (2001). A market-based perspective on information systems development. *Communications of the ACM*, *44*(11), 97–102.

Seale, C. (1999, December). Quality in Qualitative Research. *Qualitative Inquiry*, *5*(4), 465–478.

Seaman, C. B. (1999). Qualitative Methods in Empirical Studies of Software Engineering. *IEEE Transactions on Software Engineering*, *25*(4), 557–572.

Standish Group. (2013). CHAOS MANIFESTO 2013. Retrieved from `http://versionone.com/assets/img/files/ChaosManifesto2013.pdf`

Staudenmayer, N., Graves, T., Marron, J., Mockus, A., Siy, H., Votta, L., & Perry, D. (1998). Adapting to a new environment: How a legacy software organization copes with volatility and change. In *5th international product development conference, como, italy.*

Torchiano, M., Penta, M. D., Ricca, F., Lucia, A. D., Lanubile, F., & Torino, P. (2008). Software Migration Projects in Italian Industry : Preliminary Results from a State of the Practice Survey. In *Proceedings of the 23rd ieee/acm international conference on automated software engineering workshops (ase workshops 2008)* (pp. 35–42).

Vähäniitty, J., Lassenius, C., & Rautiainen, K. (2002). An Approach to Product Roadmapping in Small Software Product Businesses. In *Conference notes, center for excellence finland (ecsq2002)* (pp. 1–13).

van de Weerd, I., Bekkers, W., & Brinkkemper, S. (2010). Developing a maturity matrix for software product management. In *P. tyrvinen, s. jansen, & m.a. cusumano (eds.), software business* (pp. 76–89). Springer, Berlin-Heidelberg.

van de Weerd, I., & Brinkkemper, S. (2008). Meta-modeling for situational analysis and design methods. In *I. van de weerd, & s. brinkkemper (eds.), handbook of research on modern systems analysis and design technologies and applications* (pp. 35–54). Mankato, Minnesota State University.

van de Weerd, I., Brinkkemper, S., Nieuwenhuis, R., Versendaal, J., & Bijlsma, L. (2006, September). Towards a Reference Framework for Software Product Management. In *14th ieee international requirements engineering conference (re'06)* (pp. 319–322). Ieee.

Wiegers, K. E. (1999). First things first: Prioritizing requirements. *Software Development*, *7*(9), 24–30.

Xu, L., & Brinkkemper, S. (2005). Concepts of Product Software : Paving the Road for Urgently Needed Research. In *Caise workshops (2)* (pp. 523–528).

Xu, L., & Brinkkemper, S. (2007, October). Concepts of product software. *European Journal of Information Systems*, *16*(5), 531–541.

Yin, R. K. (2009). *Case Study Research: Design and Methods.* SAGE, London.

Appendix A - Case Study Protocol

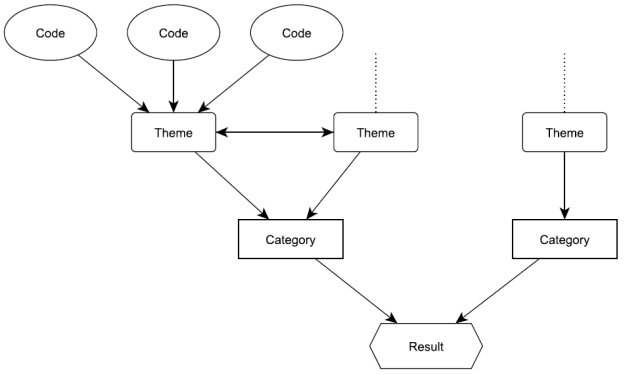
See following pages.

Preamble	This case study protocol contains essential information about the following research project, performed by the graduate student Martin Raub and supervised by the assistant professor Dr. Slinger Jansen: ***New Software required?*** *Experiences of software companies during the major product overhaul process - A multiple case study research.* **Purpose** This protocol defines the aims of the research project, specifically focusing on the explanation of the case study design in order to avoid confusion and conflict in future. Thus, the document is kept the same for and shared with all participants of the research project. It contains crucial information regarding the background of the project, the research questions as well as the methodological approach such as data gathering and data analysis methods. First, the general procedure as well as data collection and data analysis methods are explained in order to illustrate the involvement of the case companies. Eventually, the theoretical background of the research project is explained in more depth to demonstrate its relevance and importance. **Non-disclosure agreement** All the information that is obtained while visiting the participating case companies is handled confidentially and is primarily used by the University of Utrecht. This particularly concerns the data that is collected during interview sessions (cf. Research Instruments). The case companies are informed about any possible publication of the results of the research project. Results will then only be published with consent of the interviewees and the organizations they work at.
Short introduction	**Objective** To my best knowledge, there is no study, which has explored the phenomenon major product overhaul from an organizational point of view. Thus this research aims to gain insight on how different product software companies deal with the challenges and key decisions during a product overhaul. It is of interest to investigate what companies experience during a product overhaul and thus to find out what goes well and what goes wrong. This will result in a framework that contains the major challenges, the most significant decisions as well as several advices that companies provide regarding the researched phenomenon. This framework will help product software companies to better cope with the process of a major product overhaul, strengthening their knowledge as well as increasing their awareness of decision-making activities. This research will furthermore serve as a source of inspiration for product software companies, which are constantly concerned with the replacement of legacy products. This leads to the following main research question, which will be answered in the course of the research project: *RQ: What are the characteristics of a major product overhaul process in software companies?* **Approach** In order to solve the aforementioned research question, a multiple case study approach is chosen. Case studies are explicitly suitable to examine a phenomenon in its natural setting, to learn about the state of the art of a specific process as well as to generate theories from practice. Multiple cases are going to be part of the study as several sources of information

	deliver more robust results than a single case study does. Furthermore, multiple case studies have the main advantage of revealing similarities and differences of certain companies, which is necessary to develop best practices. As case examples, Dutch product software companies are selected, which have either recently undergone a product overhaul or are facing this challenge presently.
Procedures	Several example cases will be part of this research. Solely Dutch product software companies are approached, which have experienced the process of a product overhaul already. All participating case companies will receive an invitation letter (cf. Appendix), which is sent to a contact person of the company by e-Mail. In addition, this case study protocol is attached to the e-Mail in order to provide an in-depth overview of the research. Field visits are planned on a flexible basis, which means that appointments are scheduled when a participant is available. Field visits will take place at the participant's location and will last for approximately 1.5 hours. During that time, relevant data is collected from several sources of evidence (cf. Research Instruments). Field visits and any other related activity are performed by the lead researcher. In order to increase the validity of the research, this case study protocol is assessed and refined by the research supervisor, who has the experience of having performed numerous case studies before.
Research Instruments (Data collection)	This research aims at collecting qualitative data, which is done by using multiple sources of evidence: document study and interviews. **Document study** Digital or paper-based documentation and presentations present one source of information. Such documents are provided to the researcher in addition to the information gathered by performing interviews. Documents usually contain more information about the background of the case company. Furthermore, this will help to prove the correctness of specific terms that are being used in the interviews. **Expert interviews** Expert interviews are known to be highly interactive and thus mostly result in high quality outcomes. It is aimed to conduct one extensive interview at each case company, preferably consulting an executive manager, software product manager or risk assessment manager. The expert interviews are semi-structured, containing a set of predefined questions along with open-ended questions, leaving enough space for a discussion in the end. **Case study database** A case study database is established to avoid the loss of any kind of data. The database consists of material that has been provided by the case organization (e.g. documents, presentations etc.) on the one hand, material that was actually obtained by the researcher (e.g. recorded interviews, interview notes, etc.) on the other hand. The database is made accessible for other researchers in a cloud storage service in order to comprehend, where data was collected from. 

Data analysis guidelines

Data is analyzed following the same specific procedure. To simplify data analysis and enhance its validity, each of the interviews is digitally recorded. In a first step, these records need to be transcribed. Afterwards, specific codes are worked out, which are based on words, sentences or paragraphs of the transcribed interview notes. One code can include information from different interviews. Every code represents a specific theme that is closely related to this research. Different themes again can be summarized in a specific category. Thus, conclusions are finally drawn from the categories.



Analyzed data is illustrated in an individual case report of each case company. Furthermore, cross-case conclusions are drawn, pointing out the similarities and differences between several case studies. The main goal is to eventually create a framework that contains the most important findings, which have been obtained by the cross-case analysis.

Validity

To ensure high quality and increase validity of the research, some tactics are followed in different stages of the research. The following table provides an overview of these tactics.

Tests	Case Study Tactic	Phase of research
Construct validity	use of multiple sources of evidence	data collection
	establish a chain of evidence	data collection
	review of case study outcomes by participating case companies	composition
Internal validity	use of coding approach for data analysis	data analysis
External validity	selection of typical software product companies	research design
	use of replication logic	research design
Empirical reliability	develop case study protocol & interview guideline	data collection
	develop case study database	data collection

Background	**General Introduction** The ICT industry is flourishing. During the last decade, global ICT spending almost doubled from 2.1 trillion US dollar (USD) in 2002 to 3.7 trillion USD in 2013. A remarkable amount of the global market is spent on software, valued at 299 billion USD in 2013, having a constant growth rate of about three up to four percent per year. Hence, the software market is one of the most rapidly growing sectors in ICT (Fois, S. & Lysonick, R., 2012; OECD, 2002; Rivera, J. & van der Meulen, R., 2014). An increasing part of software is aimed at being offered to an open marketplace rather than to one specific customer (Regnell & Brinkkemper, 2005). This type of software – in this research named product software – has the main advantage that the product can be reproduced with almost zero costs, enabling product software companies to offer millions of copies to the market (Buxmann, Diefenbach & Hess, 2011). On the other hand, product software does have a significantly shorter life cycle than traditionally manufactured goods, which implies a real entrepreneurially-oriented challenge for product software companies, particularly for software development as well as software product management, which entails portfolio management (Lippoldt, D. & Stryszowski, P., 2009; Sawyer, S., 2000). According to Kittlaus and Clough (2009), the software producing industry is growing faster than any other industry. Technology and thus platforms are emerging rapidly and product software companies are constantly concerned with the improvement of their product portfolio. Activities of such an improvement can range from updating a current product version and adding features to simply fixing bugs over different releases. Once in a while however, when managing a product portfolio, companies need to respond to major changes in the environment of the product by migrating the software to a new platform, since maintaining the product in the old environment would become too expensive (Malton, 2001). Migration activities can therefore range from changing hardware platforms, the operating system and the architecture to changing the client interface (Torchiano et al., 2008). Besides responding to platform-related changes, product companies need to constantly review the performance of their current products in order to stay competitive in the market. It is important to question, whether a product still meets the organization's business objective and, if not, to rebalance and rationalize the portfolio (Haines, 2009; Jansen, Popp, & Buxmann, 2011). Rebalancing and rationalizing a portfolio also means to phase out a product and replace it by a follow-up product, which appears to be an inevitable part of successful portfolio management. As it is relatively easy and cheap to enter the product software market, competition is always a hurdle for software product companies in terms of market shares. So not only the rapid technological development, but also increasing maintenance costs and decreasing market shares should encourage software organizations to reconsider the introduction of a new product instead of constantly updating the old one (Haines, 2009). *This phenomenon – the sunsetting of a legacy product and its replacement by a follow-up product – is in this research described by the term "major product overhaul".* **Problem** A major product overhaul is an essential task for product software companies every few years, mostly entailing a large-scale transition from one technological platform to another such as from Windows to client-server, from client-server to cloud. With such a migration, companies can improve their product using state-of-the-art technology, possibly enter new market segments and thus reach new customers – leading to an increase in market shares and profits – as well as decrease maintenance costs.

	However, such a process also appears to be a heavy burden for companies in terms of investments and loss of time (Kittlaus & Clough, 2009). The late introduction of a follow-up product can cost a company its position in the market. A delay of new product introduction can decrease a firm's market value by 5.25 percent on average (Hoch et al., 2000). According to several researchers, migration projects are prone to failure (Hoch et al., 2000, Jansen et al., 2011). Many companies already struggle with phasing out a legacy product before replacing it by a new one. Often, companies have to extend the phase-out timeline, since required personnel was wrongly estimated as well as technology was changing rapidly during transition phase (Jansen et al., 2011). Despite these facts, it is surprising that only little research has been done regarding product software migration. Many researchers have focused on corporate legacy system migration, investigating the technological aspects and challenges of such a migration mainly (Khadka, R., Reijnders, G., Saeidi, A., Jansen, S., & Hage, J., 2011; Khadka, R., Saeidi, A., Idu, A., Hage, J., & Jansen, S., 2012). Besides having many technological challenges, a software product migration also involves organizational and business related issues. In a study of Bergey, Smith, Tilley, Weidermann, and Woods (1999), some of the most important reasons for migration project failures were mentioned as related to the organization itself. This might concern the organization's strategy, its personnel that is tied to old technology or even the management that lacks in long-term commitment. The process of a software product overhaul is a real entrepreneurial challenge for product software companies. First, it concerns the questions: "what do we keep, what do we evolve, what do we stop?" (Ebert, C., 2007). Furthermore, it involves decisions regarding investments (how much do we invest at the right moment in time?), positioning of the product (do we keep on serving the same market segment?), internationalization (shall we go international?), the personnel (what are we doing with the developers, who are not aware of the new technology?) as well as the business or delivery model (is it worth to consider a software-as-a-service solution?) (Dubey, A., & Wagle, D., 2007). These are just some examples revealing that a product overhaul involves several organizational challenges and also requires decision-making before actually moving to a new product.
References	Bergey, J., Smith, D., Tilley, S., Weidermann, N., & Woods, S. (1999). Why reengineering projects fail. *Software Engineering Institute, Carnegie Mellon University, Pittsburgh*. Buxmann, P., Diefenbach, H., & Hess, T. (2011). *Die Softwareindustrie: Ökonomische Prinzipien, Strategien, Perspektiven*. Springer, Berlin-Heidelberg. Dubey, A., & Wagle, D. (2007). Delivering software as a service. *The McKinsey Quarterly*, *May 2007*. Ebert, C. (2007). The impacts of software product management. *Journal of Systems and Software*, *80*(6), 850–861. Fois, S., & Lysonick, R. (2012). *Analyzing the Global Software industry: Trends, challenges and evolution in the business model*. Haines, S. (2009). *The product manager's desk reference*. McGraw-Hill, New York. Hoch, D. J., Roeding, C. R., Punkert, G., Lidner, S. K., & Muller, R. (2000). *Secrets of software success: management insights from 100 software firms around the world*. Harvard Business Press, Boston. Jansen, S., Popp, K. M., & Buxmann, P. (2011). The Sun also Sets : Ending the Life of a Software Product. In *Software Business* (pp. 154–167). Springer, Berlin-Heidelberg. Khadka, R., Reijnders, G., Saeidi, A., Jansen, S., & Hage, J. (2011). A method engineering based legacy to SOA migration method. In *27th IEEE International Conference on Software Maintenance (ICSM)* (pp. 163–172). Khadka, R., Saeidi, A., Idu, A., Hage, J., & Jansen, S. (2012). Legacy to SOA Evolution : A Systematic Literature Review. In *A. D. Ionita, G. Lewis & M. Litoiu (Eds.),*

Migrating Legacy Applications: Challenges in Service Oriented Architecture and Cloud Computing Environments: IGI Global (in press).

Kittlaus, H. ., & Clough, P. . (2009). *Software product management and pricing: Key success factors for software organizations*. Springer, Berlin-Heidelberg.

Lippoldt, D., & Stryszowski, P. (2009). *Innovation in the Software Sector*. Organisation for Economic Co-operation and Development.

Malton, A. J. (2001). The Software Migration Barbell. *Workshop Om Software Architecture*.

OECD. (2002). *Highlights of the OECD Information Technology Outlook 2002*.

Regnell, B., & Brinkkemper, S. (2005). Market-driven requirements engineering for software products. In *A. Aurum & C. Wohlin (Eds.), Engineering and managing software requirements* (pp. 287–308). Springer, Berlin-Heidelberg.

Rivera, J., & van der Meulen, R. (2014). Gartner Says Worldwide IT Spending on Pace to Reach $ 3.8 Trillion in 2014. *Gartner Inc.* Retrieved April 17, 2014, from http://www.gartner.com/newsroom/id/2643919.

Sawyer, S. (2000). Packaged software: implications of the differences from custom approaches to software development. *European Journal of Information Systems*, *9*(1), 47–58.

Torchiano, M., Penta, M. Di, Ricca, F., Lucia, A. De, Lanubile, F., & Torino, P. (2008). Software Migration Projects in Italian Industry : Preliminary Results from a State of the Practice Survey. In *Proceedings of Evol 2008: 4th International ERCIM Workshop on Software Evolution and Evolvability, September 2008, L'Aquila, Italy. IEEE CS Press*.

Appendix

Invitation letter to case companies

Subject: Request research project at Utrecht University

Dear sir or madam,

I am a Master student of business informatics at Utrecht University, currently working on my graduation project with the title: "*The characteristics of the major product overhaul process in software companies – A multiple case study research"*. The research project is supervised by assistant professor Dr. Slinger Jansen as well as Prof. Dr. Sjaak Brinkkemper, both renowned scholars in the field of information science.

First of all, I'd like to know, if you are generally interested in contributing to research projects of universities? To reach the goals of my graduation project, I need to interview experts (e.g. project manager, product manager, etc.) at product software companies, who were recently or are currently concerned with the process of a major product overhaul.

A major product overhaul can mean the redevelopment of an existing product and its publication as a new version, however it can also mean the adaption of a new user interface or the migration of an application into a new technological environment (such as the migration from desktop software to SaaS-based solutions). Everything that is related to a major change of a current product and that leads to a new version of a product appears to be relevant concerning the term overhaul.

The main goal of the research is to investigate the overhaul process from an organizational point of view. It is crucial to find out, how this process looks like in product software companies, so how they deal with the process regarding organizational, financial, technological and customer-related aspects. Besides, it is essential to find out, what challenges exist and what advices product software companies can give, who were already

	concerned with the process before. Since practical research about this phenomenon is scarce, my research will help to increase the awareness and knowledge regarding the non-technical aspects of a product overhaul as well as serve as source of inspiration for companies, who will have to consider about a product overhaul in the near future. It would be great, if you are willing to participate in my research. If you are interested in participation, I can send you further information about the project by e-Mail or you can contact me via phone (0657774944). Thanks a lot in advance for your attention and looking forward to hearing from you soon. Kind regards Martin Raub

Appendix B - Interview Guideline

See following pages.

<table>
<tr><td>Research Information</td><td>
Title of research: New software required?
Experiences of software companies during the major product overhaul process - A multiple case study research.

Researcher: Martin Raub
Master of Business Informatics
Department of Information and Computing Science

1st Supervisor: Dr. Slinger Jansen

2nd Supervisor: Prof. Sjaak Brinkkemper

Short introduction
• The overhaul process is an essential task for product software companies every now and then, as technology is changing rapidly as well as there is a continuous market pressure
• On the opposite, many companies struggle with that challenging process due to manifold reasons (e.g. too few resources, underestimated process, etc.)
• Despite these facts, it is surprising that only little is known about this process in literature
• Furthermore, no more practical research has been done investigating the process from an industrial point of view
• Thus, this research project is aiming at exploring the experiences of product software companies, who are currently or were recently undergoing an overhaul process, in order to provide valuable information around the phenomenon

Non-disclosure agreement
• All data is handled confidentially and is primarily used by Utrecht University
• Do you have any problems with mentioning your company in the thesis? (Internal publication only)
• In case of external publication, the name of your company will be given an anonymous name
</td></tr>
<tr><td>Company Information</td><td>Company name:
Sector:
Size:
Annual turnover:
Mission/Vision:</td></tr>
<tr><td>Participant Information</td><td>Participant name:
Experience/Background:
Position:</td></tr>
<tr><td>Interview Questions</td><td>General Information about the overhaul project
1) What is the name and the aim of the product that was undertaken an overhaul? (One product only or the whole product line?)
2) Where would you position this product in the product lifecycle? (E.g. constantly increasing demands or rather mature product?)
3) To what extent did you overhaul your product?</td></tr>
</table>

4) What was the reason for you to start the overhaul of that product?
    a. Decreasing demands (market reasons)
    b. Changes in underlying technology (technology reasons)
    c. Maintenance costs? (financial reasons)
    d. New product wished by customer? (customer reasons)
    e. Low efficiency of old product (efficiency reasons)
    f. Other
5) What strategy did you apply to overhaul the product?
    a. Redevelopment (development from scratch: new hardware platform, architecture, database, etc.)
    b. Wrapping (surround existing applications with new interfaces)
    c. Migration (moving the existing application to a new environment)
    d. Other
6) Why did you choose the previously mentioned strategy?
7) How did you approach the overhaul?
    a. Big-Bang? (Replacement all at a sudden)
    b. Hybrid/Incremental? (Step-by-step replacement)
    c. Other
8) Why did you choose the previously mentioned path?
9) Which layers of the product architecture were concerned?
    a. Presentation layer (e.g. UI)
    b. Business layer (e.g. functionality)
    c. Infrastructure layer (e.g. database system)

Technological Aspects

1) To what extent did the new product change compared to the old one?
    a. Did functionality/features change a lot?
2) What technological components did change along with the overhaul and why did you choose the new one?
    a. Hardware platform? (e.g. from Mainframe to Microsoft Application Platform)
    b. Operating system? (e.g. Microsoft Windows to Unix-like OS)
    c. Architecture? (e.g. from client-server to cloud)
    d. Run time environment? (e.g. new database management system (DBMS) such as Oracle or MySQL)
    e. Development environment? (e.g. programming language from COBOL to Java)
3) When moving to the cloud, were you afraid of any security issues that are related to cloud solutions?
4) Did you think of any mobile solutions yet?

Temporal aspects

1) How much time did you spend on the product overhaul?
2) Was there any delay in the project?
3) I'd like to illustrate the overhaul process in a more visual way.
   What were the main phases of your product overhaul and what was the main result of each phase? (E.g. business case → building new architecture → uploading data to new environment → building and testing application → informing customer etc.)

Interview guideline	University of Utrecht	Software product overhaul

Financial aspects

1) Did you perform any kind of business case before starting the project? So did you make financial forecasts etc.?
2) Did you consider any portfolio assessment model before investing in the product?
    a. Boston Consultancy Group Growth Share Model (Market growth potential/market share)
    b. McKinsey matrix (Business position/Industry attractiveness)
    c. Other
3) How would you position your product in any of these models?
4) Can you give me an indication of how much it did cost to transition to the new version of your product?
5) Did the amount of investments differ a lot from what was estimated before the overhaul?
6) How was the project funded?
7) Did you run out of money at a certain point in time?

Organizational Aspects

1) How many people did work on the product overhaul?
2) Was that what you had planned before the project?
3) Which departments were involved in the overhaul process?
4) Did you set up a specific project team with a project leader etc.?
5) Did you plan any regular meetings to discuss current topics?
6) How did the teams look like in terms of size?
7) Did you apply specific development principles/methods such as agile or Scrum?
8) Did you have to add manpower after a while? So staff that you have not planned in the beginning?
9) Did you hire new employees or add internal people?
10) Were there external people involved in the process? (e.g. did you outsource any activities such as development?)
11) Often, a product overhaul entails the use of new technologies. What did you do with the personnel that were not aware of the new technology (e.g. programming language, mobile applications, etc.)?
12) Did you offer any kind of trainings to personnel (e.g. developers, sales people, service employees) due to these changes?
13) Did the product overhaul lead to any change in the organizational structure (e.g. restructuring of a business unit)

Customer-related and market specific aspects

1) Did you perform a thorough market analysis before replacing the old product?
2) Compared to the previous version of your product, how did you position your product in the market? Did you choose to serve another market in addition or did you keep on serving the same customers as before?
3) Did you change anything in your business or delivery model? (e.g. transition to SaaS offering)
4) When and how did you inform your customers of the new product?
5) Can you briefly describe how the transition process looked like for your customers?
6) Did you have to adjust the product to some of your customers' needs?
7) Did you dissatisfy or even lose any of your customers?

	Challenges and lessons learned 1) How would you describe the process in general? Positive or negative? 2) What were the biggest challenges of your overhaul project? 3) What did you do to overcome these challenges? 4) What are the lessons learned that you derive from the overhaul? a. What were the things that went wrong during the overhaul? Did you make any remarkable mistakes? b. What were the things that went well during the overhaul? 5) Are there any advices that you would give to others, who are concerned with a product overhaul?
Remarks	Do you have any remarks that could help me to further improve the interview?

Appendix C - Documents

See digital appendix.

Appendix D - Interview Transcripts

See digital appendix.

Appendix E - Complete Category System of Data Analysis

- Overhaul type
  - Cloud migration
  - Web migration
  - Client/server migration
  - UI redevelopment
  - Product line overhaul
  - Fundamental change
  - Software evolution
- Overhaul trigger
  - Legacy technology
  - Market conditions
  - Feature extension
  - Personnel experience
  - Technological efficiency
  - Organizational opportunity
- Overhaul strategy
  - Redevelopment
  - Migration
- Overhaul approach
  - Hybrid approach
  - Big-Bang approach
  - Evolutionary approach
  - Mixed approach
- Technological aspects

  - Actual development
  - Mobile development
  - External expertise
  - Functionality
  - Architecture
  - Operating system
  - Hardware platform
  - Development environment
  - Database management system
  - Programming language
- Temporal aspects
  - Overhaul dimension
  - Overhaul delay
  - Overhaul phases
- Financial aspects
  - Business case
  - Funding
  - Financial investment
  - Profitability
  - Portfolio assessment
- Business model
  - Revenue logic
  - Billing process
  - Delivery model
- Project management
  - Agile development
  - Project planning
  - Outsourcing
  - Team size
  - Team structure
  - Training of personnel
- Organizational change
  - Business unit structure

  - Development of personnel
  - Training of personnel
  - Management structure
- Customer interaction
  - Customer training
  - Customer configuration
  - Customer communication
- Market aspects
  - Market orientation
  - Market analysis
- Overhaul challenges
  - Customer satisfaction
  - Architecture development
  - Backward compatibility
  - Complexity
  - Innovative pressure
  - Overhaul strategy selection
  - Visible benefit and effort
  - Maintenance of legacy product
  - Unpredictability
  - Parallel activities
  - Technological decisions
  - Migration of functionality
- Overhaul challenges
  - Security issues in the cloud
  - Shortage of budget
  - Overhaul delay
  - Loss of personnel
  - Loss of customers
- Countermeasures
  - Additional personnel vs. project delay
  - Cost reduction vs. shortage of budget
  - Customer interaction vs. loss of customers

  - Customer interaction vs. migration of functionality
  - Customer interaction vs. technological decisions
  - Expert support vs. architecture development
  - Expert support vs. complexity
  - Incremental development vs. complexity
  - Incremental development vs. innovative pressure
  - Incremental development vs. overhaul delay
  - Management involvement vs. technological decisions
  - Product lifeycycle status vs. overhaul strategy selection
  - Project planning vs. overhaul delay
  - Project planning vs. unpredictability
  - Recruiting vs. mainteance of legacy technology
  - Stakeholder communication vs. complexity
  - Testing vs. technological decisions
- Lessons learned & further advice
  - Importance of customer interaction
  - Importance of architecture
  - Importance of project size
  - Importance of project planning
  - Importance of project management
  - Clearness of overhaul project
  - Simplicity
  - Time reduction
  - Overhaul approach advantages
  - Overhaul strategy advantages
  - Sponsorship