Mikko Kuhalampi

SOFTWARE PRODUCT LINES AND COMPONENT REUSE – IMPACT ON CAPABILITIES AND COMPETITIVENESS OF AN ORGANIZATION

JYVÄSKYLÄN YLIOPISTO
INFORMAATIOTEKNOLOGIAN TIEDEKUNTA
2019

ABSTRACT

Kuhalampi, Mikko
Software product lines and component reuse – impact on capabilities and competitiveness of an organization
Jyväskylä: University of Jyväskylä, 2019, 73 pp.
Information systems science, Master's Thesis
Supervisor(s): Halttunen, Veikko

This thesis evaluates the impacts of the utilization of Software product lines (SPL) and component reuse on capabilities and competitiveness of an organization. The SPL method is closely linked to new product development and the ability of a company to manage software processes. While writing this paper, the author was working in a company offering SaaS-based products in B2B market. The project group aims at achieving competitive advantage to the firm through growing its product portfolio and to ensure that the customers will stick as customers in the future as well. The competition in software business is fierce, and the companies are forced to create new ways to do business in order to keep up with the development. Solutions really need to bring value to its customers and bind them tightly to the provider. In this thesis, software product lines were approached as an asset in the software product process – the research questions being: How the utilization of Software product lines and component reuse affects organizations' capabilities and competitiveness, what are the benefits and shortcomings of the method, what is the impact of component reuse on the efficiency of new product development, and how the companies utilize the methods. The software development process itself is crucial for the success of a company in keeping up with the constant change. In the thesis, the terms of SPL and new product development were explained, as well as the relationship that they have. Also, the link between capabilities, competitiveness and software product lines was explained. In the empirical part, several companies working with different software as a service – products and development projects, were interviewed about the usage and possibilities of SPL and reuse. This was done through executing semi-structured theme interviews, where the respondents of five different companies were interviewed. The results showed, that the efficient utilization of these methods require commitment throughout the company. Implementing SPL and reuse gives the company benefits in development efficiency, movement of workforce and product quality, for example. The goal of this research was to find out the benefits and shortcomings of the method and discover the impacts that the utilization of the method has on organizations' capabilities and competitiveness.

Keywords: software business, SaaS, software product lines, new product development, software components, component reuse

TIIVISTELMÄ

Kuhalampi, Mikko
Ohjelmistotuotantolinjat ja komponenttien uudelleenkäyttö – vaikutukset organisaatioiden kyvykkyyteen ja kilpailukykyyn
Jyväskylä: Jyväskylän yliopisto, 2019, 73 s.
Tietojärjestelmätiede, pro gradu -tutkielma
Ohjaaja(t): Halttunen, Veikko

Tässä tutkielmassa tarkastellaan Ohjelmistotuotantolinjojen (Software product lines) ja komponenttien uudelleenkäytön (Component reuse) vaikutuksia yrityksen kyvykkyyteen ja kilpailukykyyn. Toimintatapa liittyy olennaisesti myös uuden liiketoiminnan luomiseen ja yrityksen kykyyn hallita ohjelmistoprosesseja. Teoreettisena pohjana tutkielmalle käytetään ohjelmistotuotantolinjoihin ja ohjelmistokomponenttien uudelleenkäyttöön liittyvää aiempaa tutkimustietoa. Tutkielman kirjoittamisen aikana kirjoittaja toimi osana rekrytoinnin SaaS-palvelua tarjoavan yrityksen projektia, jossa tavoitteena on tuotevalikoiman laajentaminen kilpailuedun saamiseksi markkinalla. Kilpailu ohjelmistoliiketoiminnassa on kiihtynyt niin kovaksi, että yritysten täytyy jatkuvasti etsiä uusia tapoja kasvattaa liiketoimintaansa ja sitouttaa asiakkaitaan. Yritysten täytyy pystyä tuottamaan asiakkaalle aitoa lisäarvoa tarjoamalla pitkälle kehitettyä palvelua ja sopivia tuotteita heidän tarpeisiinsa. Tässä tutkielmassa käytiin läpi ohjelmistotuotantolinjojen käytön merkitys ja aiempi tutkimustieto aiheesta, sekä pyritään selvittämään ohjelmistotuotantolinjojen sekä komponenttien uudelleenkäytön vaikutus yrityksen kyvykkyyteen sekä kilpailukykyyn. Tutkimuskysymyksinä toimivat: miten ohjelmistotuotantolinjat ja komponenttien uudelleenkäyttö vaikuttavat organisaatioiden kyvykkyyteen ja kilpailukykyyn, mitä hyötyjä ja haittoja näillä toimintatavoilla on, mitkä ovat uudelleenkäytön vaikutukset uusien tuotteiden kehitykseen, ja miten yritykset hyödyntävät näitä toimintatapoja. Empiriaosiossa haettiin vastauksia näihin kysymyksiin kvalitatiivisen haastattelututkimuksen avulla. Tutkimus suoritettiin puolistrukturoituna teemahaastatteluna, ja siinä haastateltiin viiden eri SaaS-palveluita ja ohjelmistoprojekteja tarjoavien yritysten henkilöstöä. Tutkimus osoitti, että tutkittujen toimintatapojen implementointi ja niiden hyödyntäminen vaatii koko organisaation sitoutumista. Ohjelmistotuotantolinjat ja komponenttien uudelleenkäyttö toimintatapoina muun muassa tehostavat yrityksen ohjelmistokehitystä, mahdollistavat helpomman työvoiman liikkumisen yrityksen sisällä, ja tuovat tuotteille luotettavuutta ja laatua. Toisaalta nämä toimintatavat voivat myös hidastaa yrityksen kykyä reagoida tapahtuviin muutoksiin. Tämän tutkimuksen tavoitteena oli löytää toimintatapojen hyödyt ja haitat, sekä ymmärtää niiden vaikutuksia yrityksen kilpailukykyyn ja kyvykkyyteen.

Asiasanat: ohjelmistotuotantolinjat, ohjelmistoliiketoiminta, SaaS, tuotekehitys, ohjelmistokomponentti, uudelleenkäyttö

FIGURES

Figure 1 - Software product lines (Hallsteinsen et al. 2008) 12
Figure 2 - Cycle of the software product line process. (Gomaa, 2005)..... 13
Figure 3 - Development of product lines (Northrop, 2002)..... 15
Figure 4 - Software component reuse process. (Forsell, 2002) 16
Figure 5 - Reuse metrics and models (Frakes & Terry, 1996)..... 18
Figure 6 - Component reuse based on Mansell (2006) 19
Figure 7 - Economics of software product line engineering based on Vander Linden, Schmid and Rommes, 2007. 20
Figure 8 - Expected NPV for architectural scenarios AS1 and AS2 and strategic scenarios SS1 and SS2 (Wesselius, 2006) 23
Figure 9 - The stage-gate system based on Cooper (1990)..... 27
Figure 10 - The five levels of software process maturity (Paulk, 2002) 31
Figure 11 - Effectiveness of capabilities important to competitiveness. (Lesser & Ban, 2016) 33
Figure 12- Benefits of SPL and component reuse..... 59
Figure 13 - Potential shortcomings of product line architecture 60

TABLES

Table 1 - Relationship between SPLs and DSPLs (Hinchey, Park & Schmid, 2012) 25
Table 2 - General information about the interviewed companies 39
Table 3 - Interviewed organizations and their views on SPL and reuse. 57

CONTENTS

ABSTRACT 2

TIIVISTELMÄ 3

1 INTRODUCTION 7

2 SOFTWARE PRODUCT LINES 11
2.1 Theoretical review of software product lines 11
2.2 Components and component reuse 15
2.2.1 Component reuse process 16
2.2.2 Strengths and difficulties of the organizations implementing reuse 18
2.3 Differences between software product line engineering and single system development 20
2.4 Business strategies and return on investment for SPL 21
2.5 Benefits and pitfalls of software product lines 22
2.6 Software product lines and component reuse in the future 24
2.7 New product development 26
2.7.1 Defining new product development 26
2.7.2 The process of new product development 27
2.8 Chapter conclusion 28

3 LINKING SOFTWARE PRODUCT LINES AND COMPONENT REUSE TO CAPABILITY AND COMPETITIVENESS 29
3.1 Organizational capability as a tool for success of an organization 30
3.2 Competitiveness 32
3.3 Impacts 34
3.4 Component reuse approach 34
3.5 Chapter conclusion 35

4 CARRYING OUT THE RESEARCH 36
4.1 Background and goals 36
4.2 Presenting the research method 37
4.3 Data collection 38
4.4 Data analysis 40

5 RESULTS 42
5.1 General information and interviewees' backgrounds 42
5.2 Software product lines 44

5.3 Component reuse 49
5.4 Linkage to competitiveness and capabilities 53

6 DISCUSSION 57

7 CONCLUSION 61

REFERENCES 66

APPENDIX 1 70

1 INTRODUCTION

Competition in software business is fierce between different kinds of service providers. In order to keep up with the development, service providers must continuously develop their products, and even more importantly – look for new possible business opportunities. "From being considered a configuration mechanism for electronic systems, software has become the core of most modern systems supporting individuals, companies and societies" (Bosch & Eklund, 2012).

Currently, this is a large phenomenon in the industry. Companies are constantly looking for ways to bring more value to their customers. Especially, in software as a service business (SaaS), the providers cannot be sure if the customer is planning to change the provider or not. Also, the sales process can be long, and it is not an easy task to get new customers fast enough. This brings an opportunity to increase efficiency by component reuse and creating software product lines (SPL).

This subject is very interesting, as many software companies suffer the problems of slow development processes and growing demands of the customers. This thesis was made while working in an organization that provides recruitment software and recruitment platforms for its clients, currently working on widening their product portfolio. In this thesis, software product lines are an enabling and resource-saving act in an organization. By using software product lines companies can achieve significant savings – by approaching the system development to consider a family of software products, not just one single system (Gomaa, 2005). Organizations that use SPL are widening the use of them to the whole organization and even over boundaries outside their own organization (Bosch, 2009). This thesis focuses on the impacts that the implementation of the approach has on companies' success. Capabilities and competitiveness play a large role in the success of a company, and therefore the link between them and SPL approach is evaluated. Many earlier studies have brought up potential benefits and shortcomings of the method, but the problem is the measurement and verification of the impacts is challenging, as the SPL impacts are more long-term results. This is why it is interesting to study the linkage to capabilities and competitiveness, as the earlier studies mostly have focused on individual bene-

fits of the method concerning software business. Quite few studies exist on the matter, especially with a straight linkage. Also, the concept of component reuse is an interesting topic, as it is widely used and is going through changes as the usage of external component libraries is constantly growing. The earlier studies have studied SPL from different perspectives but linking the usage of SPL and component reuse straight to capabilities and competitiveness is not widely studied.

There is quite a lot of research both on software product lines and component reuse. However, the results are not always in line with each other – some agree with the benefits and still suggest wide usage of product line architecture, and some promote agile methods over SPL (Ahmed & Capretz, 2010). Main sources of earlier research information concerning product lines and reuse are the studies by Käkölä and Duenas (2006), Clements & Northrop (2003), Forsell (2002) and Northrop (2002). The studies mentioned provided thorough information basis for product lines and its impacts on organizations' success. Numerous researches have studied the basis of software product lines and its process. Some research has also been made on the benefits and shortcomings of the method – however, it is problematic, as the measurement of the results is challenging. The results have offered separate benefits and shortcomings of the results, but there is a gap in connecting the impacts of SPL and component reuse to organization capabilities and competitiveness, and that is what this research aims at studying.

Also, the way of thinking in an organization has a massive impact on the results. To achieve the right spirit within the company, studying this subject can help in many ways. In order to create such atmosphere, the organization must decide to work in a significant way and stick to it, until it becomes a habit. Achieving effective and lasting results demands strong commitment from the entire organization (Ahmed & Capretz, 2010). This way, the base processes stay as they are, and the companies can keep constantly looking for new business opportunities in the field. Also, when an organization has products with similar technical backgrounds, if a problem is spotted or a better solution is invented, the solution can be taken into use in all the systems (Metzger & Pohl, 2014).

This study is set to consider software businesses and SaaS-based products. The impact that software product lines have on the capability and competitiveness of a company from the point of view of component reuse, is evaluated. In the empirical part the focus will be on examining the potential of SPL use and the reuse of components. SPL use brings many possibilities in software product development. It is also examined how the companies use this method and do they see SPL as a potential option in the future as well.

Software product lines work as theoretical background to this thesis. The idea behind software product lines is closely related to new product development. The aim of this study is to highlight the impacts of software product lines approach in to organizations' capability and competitiveness. Usually when companies start using product lines, they aim at decreasing development costs, reducing product time to market, expanding their product portfolio or to

achieve commonality in different products from the point of view of user experience (Bosch & Bosch-Sijtsema, 2010). The process of New product development (NPD) is also reviewed as it is closely related to SPL. The process is considered from the point of view of SaaS-based products – when answering to customer needs, how to evaluate whether to build new features to the existing product or develop a completely new product.

Getting deeper into the subject, there is a need for definitions for the key terms. A *software product line* is a portfolio of similar software-based systems or products that are produced from an internally shared set of software assets while using common means of production (Clements & Northrop, 2003, Clements, 2002). *Software product line engineering* is defined as "an industrially validated methodology for developing software products and software-intensive systems faster, at lower costs, and with better quality" (Käkölä & Duenas, 2006). *Organizational capability* is the ability of a company to manage their resources effectively and thus gain advantage over competitors. This usually means effective management of employees and following customer demands. (Grant, 1996). Organizational capability and competitiveness are closely linked to each other – competitiveness can be gained through a high level of organizational capability. Lee (2001) states, that the attractiveness of an organization and its ability to create competitive advantage can be seen as factors of organizational capacity. *Competitiveness* means the ability of an organization to survive in the market competition. One can measure competitiveness for example by price, marketing or internal knowledge.

Needless to say, that in the rapidly changing world of software business, the organizations competing in it must be able to react to change. Therefore, all the resources that can be saved in the current development processes are worth a lot. The research question of this study goes as follows:

- How the utilization of software product lines and component reuse affects organizations' capabilities and competitiveness in Software business?

Alongside the research question, a set of sub-questions are presented:

- What are the benefits and shortcomings of software product line approach?
- What is the impact of component reuse on efficiency in the process of new product development?
- How do software companies utilize product lines and component reuse?

The objective is to detect the benefits and shortcomings that using SPL brings to the organization. Also, this study tries to find answers how to measure these impacts. It is important to remember, that this thesis does not consider

agile methods or its impacts on capabilities or competitiveness. Focus is on software product lines and component reuse. Bringing agile methods to this study would make it difficult to get any reliable outcomes from the study.

Especially new product development is a process that takes a lot of development resources and therefore is closely linked to software product lines engineering. The research subject itself is important, because the cost-savings the organizations can make are significant, and the number of SaaS-based products is very large and is continuously growing. This means, that there will be a huge demand in the future for ways to reduce the resources needed in time-consuming development.

The structure of the thesis goes as follows: First, the subject is approached through a wide literature review to find the theoretical basis that is behind software product lines and component reuse. After that, the habits of Finnish software companies are approached through conducting a semi-structured theme interview, to get a thorough picture of the benefits and shortcomings of reuse and find out the characteristics of different use cases and their results. While conducting a qualitative research, theme interviews work as a reliable source of data. The topics and themes are discussed beforehand, but the main questions are asked only in the interview.

2 SOFTWARE PRODUCT LINES

In this section, software product lines are presented as a theoretical basis of this thesis. First, software product lines (SPLs) are reviewed from the theoretical point of view. After that, the differences between other approaches is evaluated. Also, possible business strategies for using SPL are reviewed as well as potential benefits and pitfalls of the method. After this, a brief look into the future of software product lines and software product line engineering is provided.

2.1 Theoretical review of software product lines

Software product lines play a large role in daily actions of companies. The traditional way to develop software is to develop single systems—which means developing each system individually. For software product lines, the development approach is a lot larger – it considers a whole family of software systems. This approach involves analyzing which features of the software product family are common, optional or alternatives. (Gomaa, 2005). SPL is a very efficient way to enhance new product development and widen the whole product family through product lines. The differences between software product line engineering and single system development will be reviewed in the next chapter. According to Clements and Northrop (2003), "A product line is a set of products that together address a particular market segment or fulfill a particular mission". Bosch (2009) states, that software product lines can be seen as the most successful approach to intra-organizational reuse. Numerous companies have been able to make their R&D processes a lot more efficient and to widen their product portfolio through software product lines. Product lines offer a great level of configurability that can only be reached by using shared software components in the systems. Through this kind of development, also customer experience can become a lot better and consistent. Needless to say, software product

lines have a significant impact on a company's business, when the way of working is executed right. (Bosch, 2009).

Product lines bring along several kinds of benefits for the following areas: Requirements, architectural design, components, modeling and analysis, testing, planning, processes and people. (Clements & Northrop, 2003). For example, it can make these processes more effective by reusing the existing architectures and solutions of the company. Commonly used components also make testing a lot easier. In planning, usage of SPL can make it easier to evaluate time estimates more realistically. From the requirements perspective, using product lines lowers the workload, as most of the requirements are common with other products. Therefore, time is saved, as there is no need for further requirements analysis. (Clements & Northrop, 2003). From the architectural point of view, the earlier products help a lot, as there is no need for using all the normal effort to design, as the base already exists. Components share detailed designs between existing and new products. They can easily be reused, as well as the documentation can be eased. There is no need to start from zero. By using product lines, analysis and modeling can also be made a lot easier. According to Hallsteinsen et al. (2008), product lines have a significant impact on efficiency and the teams feel a lot more confident about the time frames they put in to projects. Also, in terms of testing the teams save a lot of time by using already existing patterns that are partly or completely tested. Data reuse and common ways of working make it easier for organizations to move employees between projects, as they have a good information on the product, even though they worked on another one completely.

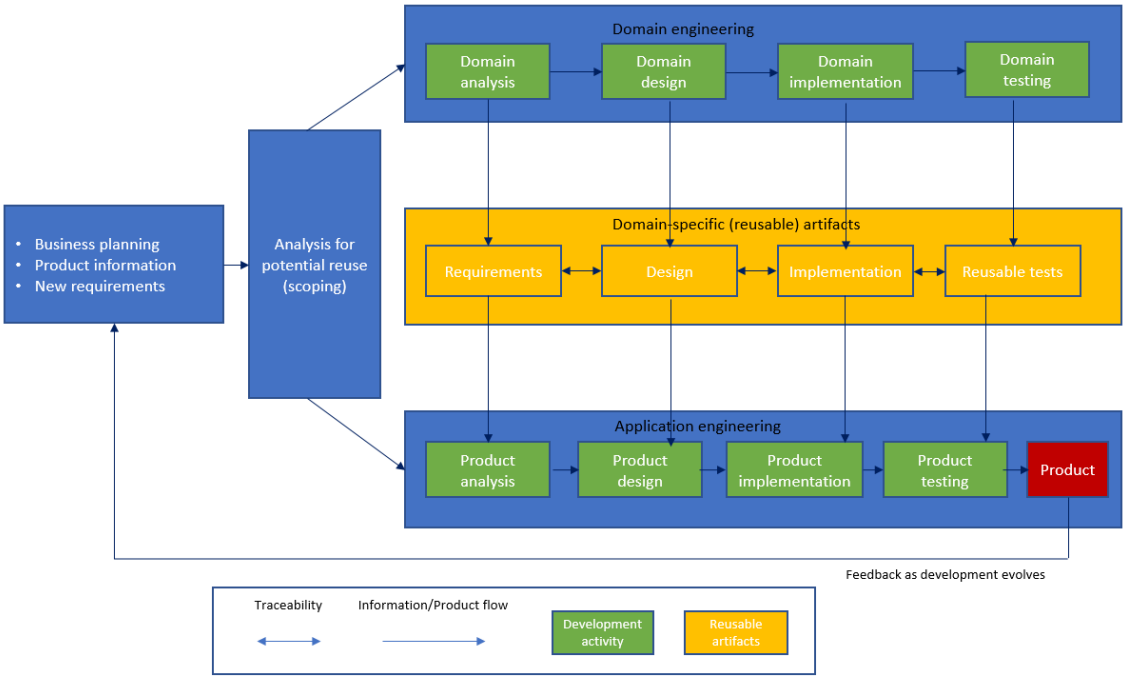


Figure 1 - Software product lines (Hallsteinsen et al. 2008)

In Figure 1, Hallsteinsen et al. (2008) present a figure to characterize software product lines. They employ a two-life-cycle approach that separates domain and application engineering.

Gomaa (2005) describes the process of software product lines and product line engineering in various ways. A good example of the product line process in is presented in Figure 2. Gomaa (2005) posits, that most of the application requirements come from outside the process. The product line engineering starts with product line analysis models, architecture and reusable components from the existing products. After that, process moves from product line reuse library to application engineering. In this phase, it is examined whether the application is ready for further development or if there still are unsatisfied requirements or errors in the application. In this thesis, more attention will be given to components and component reuse. Components are explained further in the next chapter.

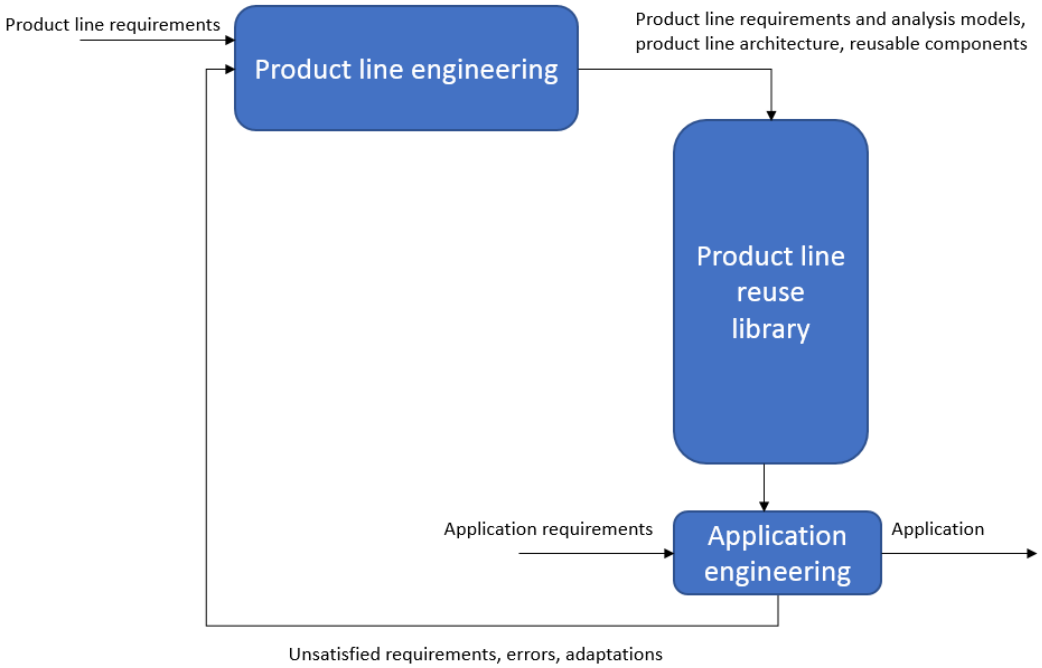


Figure 2 - Cycle of the software product line process. (Gomaa, 2005)

According to Gomaa (2005), the goal of SPL usage is to design a software architecture for the product line, which has common components, optional components, and variant components. Common components are needed by all members of the product family, optional components by part of the family and variant components are widely used by different members of the family, but demand modifications before usage. By doing this, the organization can focus on adapting and configuring the existing architecture to fit the new needs, instead of starting with nothing.

There has been a lot of research on SPLs in the 21st century, and it is seen as a process that is able to improve efficiency in software development and new

product development. Käkölä and Duenas (2006) divide the research areas of software product line engineering and management to five different areas:

- Product line management
- Product line requirements engineering
- Product line architecture
- Product line testing
- Specific product line engineering issues

In this thesis, the focus is on the overall process of software product lines, and its impacts on organizations' development processes and new product development. In today's research and conversations on software product lines, the term Dynamic Software Product Line (DSPL) comes up quite often. DSPL's will be explained further in section 2.1.5.

Product lines are also widely used in industrial fields. There has been a lot of discussion about the differences and commonalities in the approaches of industrial product lines and software product lines. Rabiser, Schmid, Becker, Botterweck, Galster, Groher and Weyns (2018) found, that even though the industrial and academic software product lines focus on different parts of the process, the basis of product line thinking stays the same. According to Rabiser et al. (2018), recent research topics in academic research such as entire software ecosystems and multi-product lines, or dynamic product lines are not covered as much in recent industry research. This is natural, as software product lines bring along more options to add in.

Northrop (2002) presents the development of existing product lines in the following Figure 3. SPL development is divided to three different parts: Core asset development, product development and management. Core asset development is about establishing production capability for products, and product development is implementing the requirements for individual products. Management is about handling the process as a whole. Northrop (2002) mentioned, that management both at the technical and organizational levels must be strongly committed to the software product line effort to ensure success in implementing product lines.

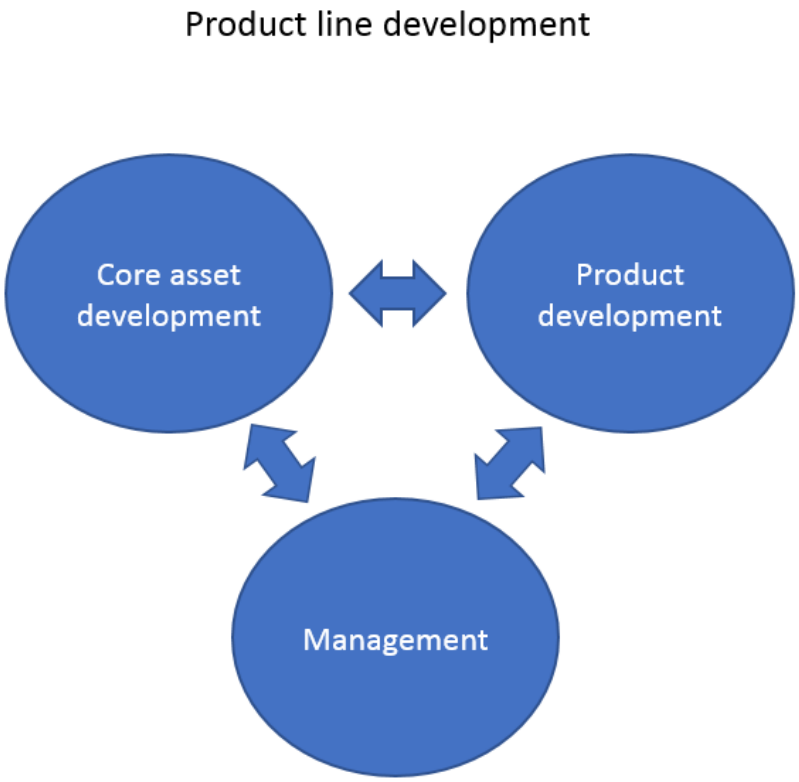


Figure 3 - Development of product lines (Northrop, 2002)

2.2 Components and component reuse

In this chapter, the importance of component reuse is highlighted. First, the reuse process and its phases are discussed. The usage of software product lines enables component reuse on a whole new level - reusable components are building blocks that can be used more than once to build a new system. According to Forsell, Halttunen and Ahonen (2000) a component is a common term for reusable piece of software. In this thesis, components are both software components and business-related components. Considering the business-related impacts of software product lines on capability and competitiveness, it is necessary to also consider the business components as potential elements of reuse. According to Frakes and Kang (2005) "Software reuse will only succeed if it makes good business sense". This is true, as the main goal of software reuse is to improve quality and make the processes more efficient - thus, create more profit.

2.2.1 Component reuse process

To understand the way component reuse, it is needed to get familiar with the its process. Component reuse process consists of many different phases (Forsell et al. 2000). Reuse process can be divided to four main activities:

- managing the reuse infrastructure
- producing reusable assets
- brokering reusable assets
- consuming reusable assets
  (Lim, 1998; Forsell et al. 2000; Forsell, 2002).

In this case, Lim (1998) originally defines components as "assets" to emphasize that components are more than just code. First of the main activities, managing the reuse infrastructure, is the most important one. It plans and drives forward the next three phases – therefore, it manages the whole process. Forsell (2000) presents the following figure to illustrate the process of software reuse. In the model, the producing, brokering and consuming components are tied with the entire software development process. The main thing is, that the reuse process is a key part in the overall software product line process – and it is there to stay. The model considers only the most important parts of reuse-oriented development, but it gives a good image of how the process relates to the development process.

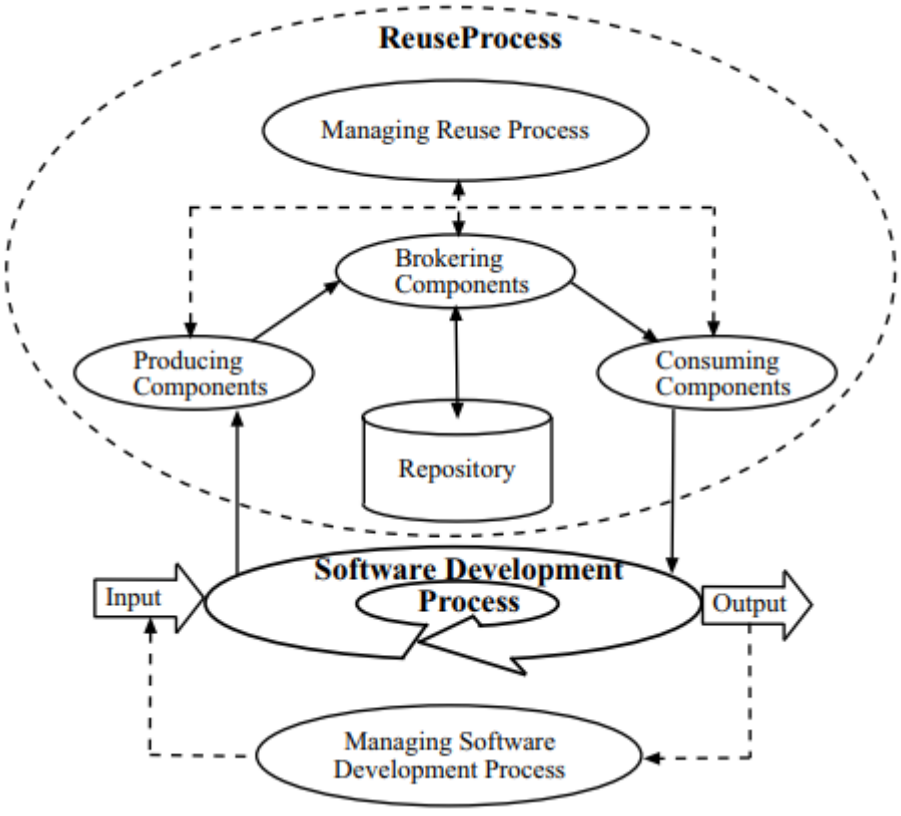


Figure 4 - Software component reuse process. (Forsell, 2002)

Succeeding in the reuse process naturally demands efficient management of the process. This means, that the management should be the first thing to consider when launching a reuse program in a company. The management is seen as a critical aspect in the process (Lim, 1998). The planning of the process and defining the reusable components and reuse in the organization. Managing the products means managing the components. The quality of the components plays a key role in making the process profitable. Forsell (2002) divides management of the reuse process to three parts: management of the process, management of the products and management of the people. By combining these tasks, it is possible to spread the reuse process to the whole organization.

In Figure 4, Forsell (2002) describes producing components, brokering components and consuming components as parts of the reuse-oriented software development process. Next, those terms are reviewed shortly.

Producing components holds in the design and producing of the components. It involves two main phases – domain analysis and component creation. By analyzing the domain, it is possible to identify all the reusable components. Software components are created in different software projects and systems. The same methods and techniques are valid also in component producing, but usually reuse perspective sets more criteria for the creation phase in order for the components to be reusable. Naturally some maintenance and fixes are needed, but the workload is significantly smaller than starting with nothing. (Forsell, 2002)

Brokering components means giving the components to the whole organization to use. This means for example component reuse across different projects. Brokering also creates trust to existing components and thus they are used more widely. Lim (1998) mentions five tasks to brokering components: assessing, procuring, certifying, adding, and deleting components. Brokering is more than just maintaining the component repository – it also includes a lot of validation and verification of components before they are put into the repository.

Consuming components is usually defined as finding, understanding, modifying and integrating components. Also, documentation holds an important role as part of component consuming. Documentation makes the process more efficient than it was before. An important thing before consuming the components is to identify the system we are working with and plausible components to it (Forsell, 2002).

Frakes and Terry (1996) mention, that a reuse program and process must be planned, deliberate and systematic in order to maximize the profits produced. They have studied the metrics and models of reuse a lot and highlight the importance of reuse impact measurement. If the impacts cannot be measured, it is difficult to explain the potential of reuse and product line thinking to the management. In Figure 5, Frakes and Terry (1996) present the main metrics and models of reuse. The main metrics are categorized into following types: reuse cost-benefit models, maturity assessment, amount of reuse, failure modes, reusability and library metrics.

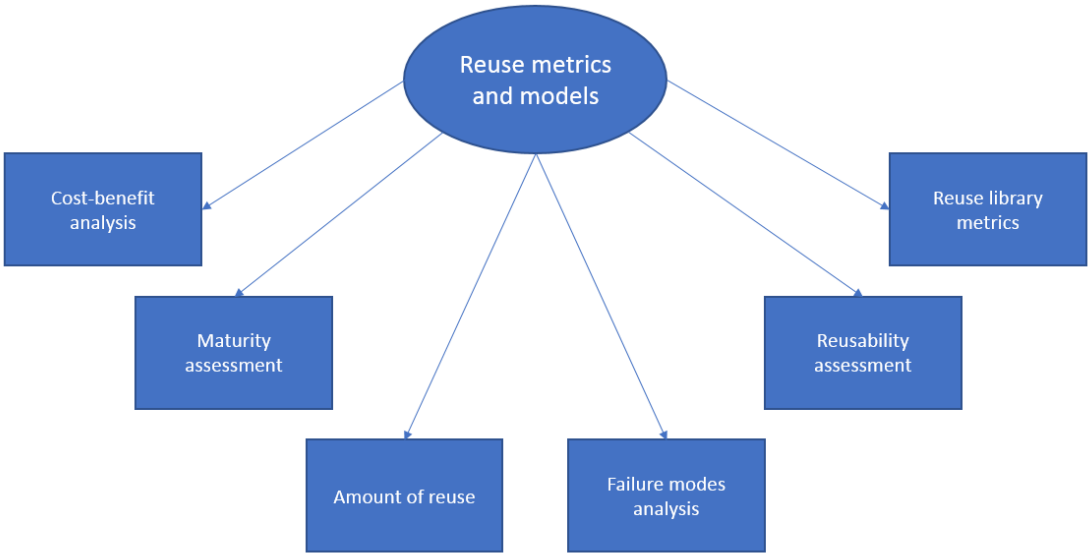


Figure 5 - Reuse metrics and models (Frakes & Terry, 1996)

2.2.2 Strengths and difficulties of the organizations implementing reuse

In this chapter, the characteristics of organizations that are likely to succeed with reuse, and the ones are not, are presented. Certain characteristics are demanded when an organization is going to implement systematic reuse. Mansell (2006) presents aspects that reuse-positive organizations usually have:

- A reuse technique already, and new development reuses components.
- An organizational structure that supports reuse, and a repository.
- A common view on reuse and its benefits for the whole organization.
- A reuse process that is continuously managed, and a process for quality management exists.
- A common way to do reuse in other sectors too, such as documentation, design and analysis.
- A habit, that all stakeholders are a part of the project definition and planning.

This list of characteristics that the organizations usually has is in line with other research papers too, but nicely concludes the most common and beneficial qualities. Mansell (2006) also presents main difficulties that organizations have in implementing reuse:

- Developers themselves are responsible for maintenance and support of the assets, leads to reduced motivation.
- Benefits or reuse are not identified.
- The developer and the asset share a dependency.
- The development of reusable assets has unclear guidelines.

- Reuse is not considered as an institutional issue.
- Reuse is more ad-hoc than systematic.

These aspects define very well the characteristics that are not beneficial for the organization when implementing component reuse or product lines. The requirements for the organization are even more strict with product lines. This means, there are certain characteristics the organization should and should not have when implementing the approach. They have a large impact on the outcome of the implementation.

Figure 6 demonstrates a schema of common identified reuse scenarios. In systematic reuse, the components are developed with an idea that it will be reused later. In other projects or products, the existing component can be found either from the repository or another solution when a function is called, it is found from the repository. Repository stands for a library of reusable assets which is established usually in the beginning of implementing reuse. Management usually controls the usage and current state of the repository. When a component is reused from project to another unplanned, Mansell (2006) calls it "ad-hoc reuse". Sometimes reuse can happen also between different development environments both systematically and ad-hoc. No specific identified location means, that there is no mechanism or certain place that the components are put in.

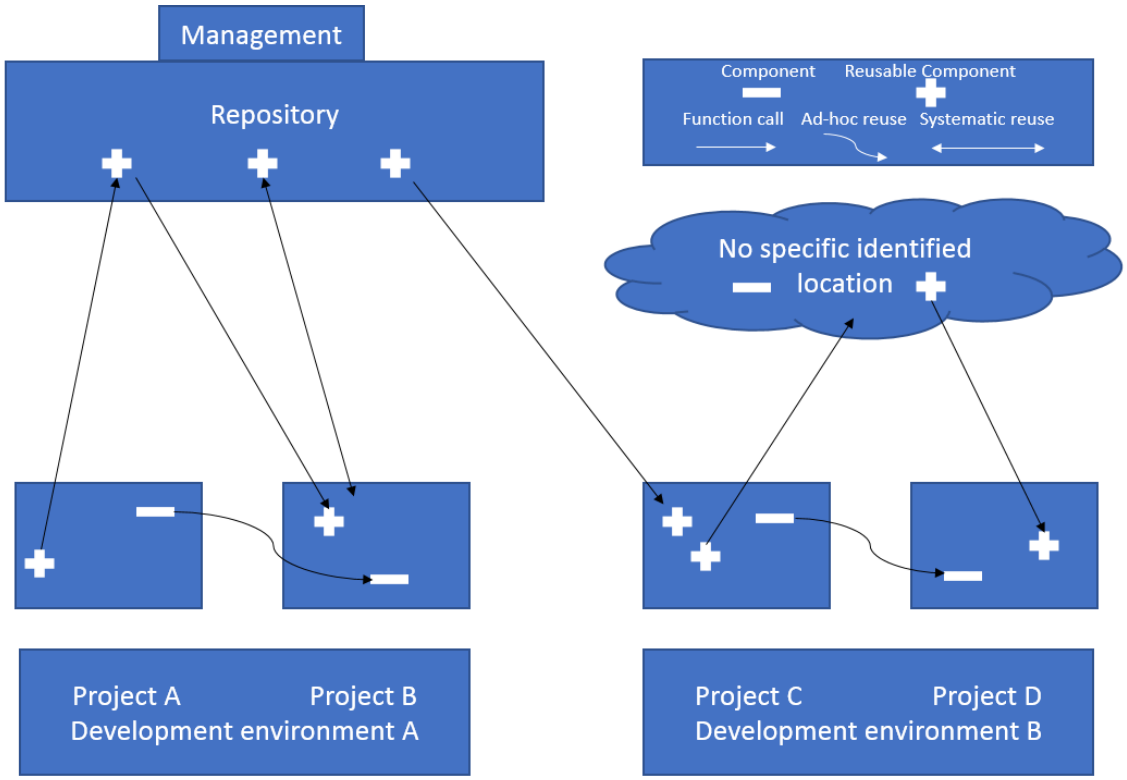


Figure 6 - Component reuse based on Mansell (2006)

2.3 Differences between software product line engineering and single system development

To understand the reasons and factors behind these approaches, we must go deeper into characteristics of them. Käkölä and Duenas (2006) present two primary ways that SPL engineering differentiates from developing single systems: First, they mention that SPL engineering demands two distinct software development processes: domain engineering and application engineering. In this case, domain engineering characterizes the variability of the product line, and by that establishes a widely-used software platform to develop high quality applications in a short time frame in the product line. Shortly, application engineering exploits the product line. (Käkölä & Duenas, 2006)

Second, they mention, that SPL engineering needs to define and manage variability in the whole product family. In this case, variability is approached in every domain artefact, such as models, components, test cases etc. This way the customers can have tailored solutions precisely to their needs (Käkölä & Duenas, 2006). Van der Linden, Schmid and Rommes (2007) present in Figure 7 the differences of single systems development and product lines. As it shows, the up-front investment is larger in the beginning with SPL, but costs start to get significantly lower when the organization has three or more systems, that share the same technical platform.

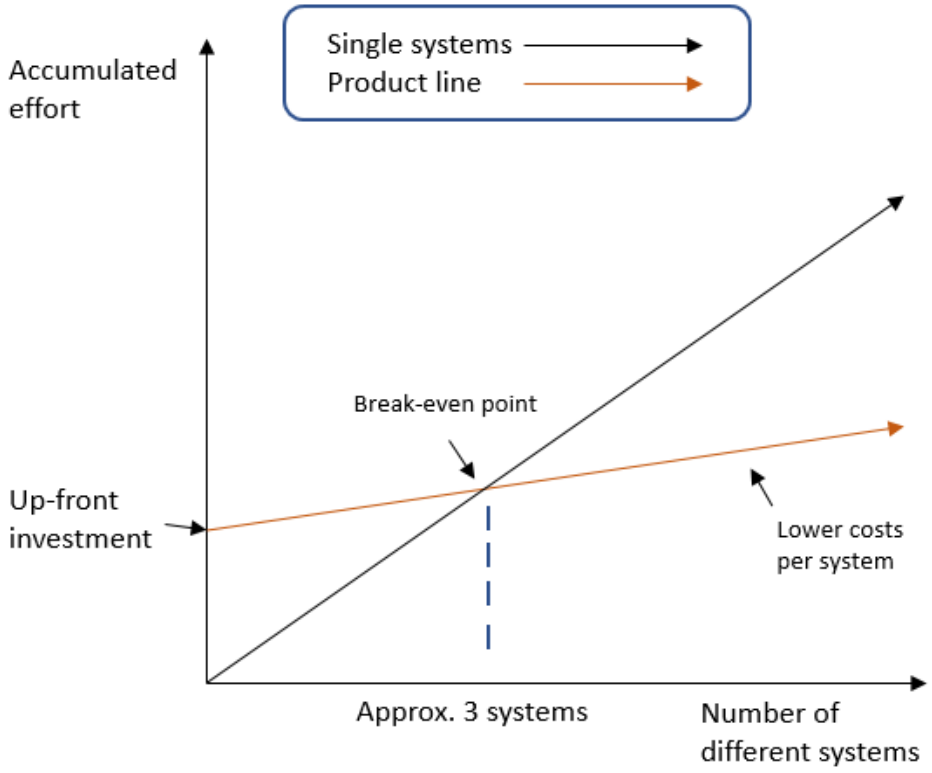


Figure 7 - Economics of software product line engineering based on Vander Linden, Schmid and Rommes, 2007.

Organizations do not end up using product lines by accident. Usually some kind of evaluation is done to increase efficiency in the future. According to Clements, Kazman and Klein (2001), the primary benefit of architecture evaluation is, that it uncovers problems that cannot be ignored -else, they would be orders of magnitude more expensive to modify and correct later. Architecture evaluation produces better architectures. In these evaluations, product lines usually look tempting, but they demand higher investments in the beginning than other solutions. Therefore, the decision makers need to be convinced about the possible outcomes in the future. The importance of the architecture in software systems is highlighted, because it defines the modifiability, performance, security, availability and reliability of the system in the future. If the base is not working properly, there are no development tricks to write some qualities out of the system. (Clements et al. 2001)

2.4 Business strategies and return on investment for SPL

Implementing SPLs can be a major success for a company. Bosch and Bosch-Sijtsema (2010) mention, that in some cases adopting software product lines allowed reducing development expenses by 50% or more and decreasing defect density with significant results. Successful software businesses focus on one of three following strategies: Operational excellence, Customer intimacy or product innovativeness. Firms aiming at operational excellence try to give their clients the best total cost of using the product. The main goal is to reach higher productivity and lower overall costs. Customer-intimate organizations want to have a high-level complete solution they can provide the customer with. They constantly seek for long-term relationships with customers and usually customize the products for their customers to keep them satisfied and tied to the product. Product-innovative companies aim at providing customers with the best products and they target mass-markets. They want to act rapidly and grab new business opportunities before others. This way they gain the major part of potential customers before others can provide such products. (Käkölä, 2003)

Böckle, Clements, McGregor, Muthig and Schmid (2004) have researched the potential return on investment (ROI) of using SPL engineering. According to the authors, managers are constantly asking for evidence of the results to defend their vision of using software product lines. ROI calculations are a way for managers to argue on behalf their decisions. They came up with a model, that can calculate the costs and benefits that can be expected from various product development situations. Böckle et al. (2004) suggest, that organizations "should engineer your products in a way that takes advantage of their commonalities while controlling their differences". Software product lines are usually the more economical way in the long run.

2.5 Benefits and pitfalls of software product lines

In this chapter, the research focuses on benefits and shortcomings of software product lines usage presented by earlier research. The first part presents the benefits and the second part presents the shortcomings. Software product lines have a major set of benefits, but one cannot forget the potential shortcomings of applying the approach in to use.

SPL usage can bring a lot of benefits and positive effects in an organization. For example, one of the largest and clearest outcomes is the reduced time-to-market. The main thing is, by using SPL organizations can reduce the timeframe between identifying the market need and bringing the product to market significantly. This is mainly because of the large building blocks made while implementing SPL, which generates the opportunity of reuse of product architecture. (Wesselius, 2006). Of course, the organization must also consider the time that is needed to build such a product line architecture. The planning and building of the structure take time in the beginning. This may postpone the generated income to result. By using SPL, companies reduce the time-to-market, and thus speed up the income.

Figure 8 (Wesselius, 2006) presents well the net present value differences in the cases where a product line platform is built and not built. In Architectural scenario 1 a platform is not built, and the products are developed one by one. In Architectural scenario 2, development is started by building a software platform. Wesselius defines the strategic scenarios as follows:

> "SS1: products will be demanded that can be developed on the basis of the platform until at least 2013
>
> SS2: products will be demanded that can be developed on the basis of the platform until 2009. From 2011 onwards, the products will require features that require entirely different platform."

By looking at the results, it is clear that the most beneficial scenario is AS2 in case of SS1. This points out, that when we can assume the platform to work as a base also in the future, the best result will follow. In this case, the probability of SS1 was seen as almost 100%, so the this supports decision making a lot: choosing to build the platform is beneficial in the long term, but has costs in the beginning. (Wesselius, 2006)

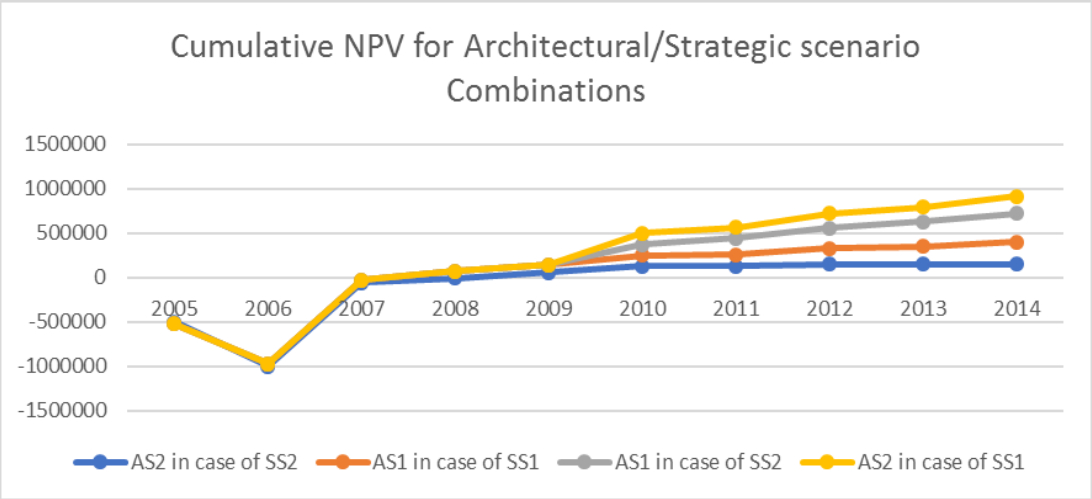


Figure 8 - Expected NPV for architectural scenarios AS1 and AS2 and strategic scenarios SS1 and SS2 (Wesselius, 2006)

Another clear benefit caused by SPL usage is cross-product compatibility. Product line engineering can bring many opportunities to sell upgrades to different products, especially the ones with long lifetime. Upgrades can be for example service packs, newer versions of the system or new functionalities or improved performance (Wesselius, 2006). Often these upgrades are things that the clients are willing to pay for and this can be a very beneficial business model for the company to add functionalities and respond to customer needs. This will raise the level of customer experience and bring revenue to the organization. (Clements & Northrop, 2003). When several different systems are built on the same platform, by responding to customer needs and creating new functionalities, the organization has to do the basic work only once, and then the solution can be copied to the other systems too. This also lowers significantly the overall development costs needed. If the systems do not have the same base platform, the upgrade sales profit numbers are going to be remarkably smaller. The situation where the company can benefit from reusing the upgrades and versions in several systems, can be approached by defining strategic scenarios for customer needs in terms of upgrades and defining the expected reduction of costs of reusing these upgrades. (Clements & Northrop, 2003).

Software product lines can make a remarkable difference in efficiency, but it can also bring different kinds of pitfalls, especially when the process is not managed properly. Software product lines can become victims of their own success and challenges start to pile up costs. These challenges can be for example relatively high coordination costs, slower release cycles and high system-level error density (Bosch & Bosch-Sijtsema, 2011). Wesselius (2006) presents the following shortcomings in their research: Platform over-design and perfectionism, short-term focus, lack of vision and decision making. Platform over-design means, that the management can end up perfectionating "the perfect product line architecture", which demands a lot of time and resources. Also, it will make the development period longer for the product line platform, which is not ideal.

As the development period takes longer, the return-on-investment will start later than in a reasonably developed architecture. By drifting into perfectionism, the organization can also end up creating features to the architecture, that are never used. Needless to say, that is simply a waste of resources. Wesselius (2006) posits, that "the firm cannot afford to be prepared for everything".

Short-term focus can also be a setback in terms of long-term plans. This simply is a consequence of setting the time horizon too near and causing the management to miss potential long-term business opportunities. Wesselius (2006), points out, that on the other hand it can also lead to ignoring possible future costs. An important thing to understand is, that investments in software product line engineering demand time to be profitable – by applying these methods, organizations will probably not see rapid change. The results will take time. The decision, whether to trust the payback to come, depends on the organization itself. It is easier for the management to set short-term goals and see if they work or not. If the payback is a long-term process, they are also seen as more risky options. (Wesselius, 2006). The uncertainties about the future results in long-term can be tackled by net present value calculations and challenging the organization to constantly think also about the larger change that is about to happen in the organization.

This is closely related to another pitfall for SPL use - lack of vision and clear decision making. Especially in organizations that are not holding on to their own priorities, executing product line architecture is a challenging mission. The management must stick to their decisions in order to make SPL architecture work. The demands and vision cannot change constantly. This shortcoming can be prevented by defining the strictly the strategic scenarios and visions of the future. This way the organization can be sure that everyone is aware of what will happen next.

Many earlier studies on SPL mention few benefits and shortcomings of the method, but they do not take the problems in testing in to account – testing both SPLs itself and their impacts is a hard task. Testing the product line itself can help to explain the level of results achieved. Engström and Runeson (2011) summarize the challenges in SPL testing in to three main challenges: how to handle the large number of tests needed, how to balance the effort made for reusable components and how to handle the variability within a product line.

2.6 Software product lines and component reuse in the future

Software product lines have been studied a lot in the 21st century. In this chapter, the future of SPL engineering is going to be evaluated -what kind of challenges will it face and some variations of a normal software product line. As the field evolves, there will also be changes in the procedures. Software product line engineering will be needed in the future and it will keep up with the development of the field (Gomaa, 2005).

As earlier mentioned, the term of Dynamic Software Product Lines (DSPL) is becoming more common. According to Hinchey, Park and Schmid (2012), dynamic software product lines extend the existing product line approaches by moving the capabilities to runtime, and helping to ensure system adaptations to lead to desirable properties. In table 1, the authors present the main relationships and differences of SPLs and DSPLs. Nowadays DSPLs are also seen as possible beneficial alternative as an architecture model. The largest benefits that DSPL usage brings, is possible systematic engineering foundations that it can provide to adaptive and self-adaptive systems.

Classic software product lines	Dynamic software product lines
Variability management describes different possible systems.	Variability management describes different adaptations of the system.
Reference architecture provides a common framework for a ser of individual product architectures.	DSPL architecture is a single system architecture, which provides a basis for all possible adaptations of the system.
Business scoping identifies the common market for the set of products.	Adaptability scoping identifies the range of adaptation the DSPL supports
Two-life-cycle approach describes two engineering life cycles, one for domain engineering and one for application engineering.	Two life cycles: the DSPL engineering life cycle aims at systematic development of the adaptative system, and the usage life cycle exploits adaptability in use.

Table 1 - Relationship between SPLs and DSPLs (Hinchey, Park & Schmid, 2012)

High usage of product lines has also raised research on larger contexts. Bosch (2008) highly recommends the companies to add to their software product line thinking also a bigger picture, Software ecosystems. Software ecosystems bring the original software product line and the existing platform to a larger context. Bosch (2009) states, that once an organization has decided to make its own platform available from outside the organizational boundaries, the existing software product lines can be modified to software ecosystems. By offering these kinds of ecosystems, it is easier for the company to conquer new clients as they seem to follow the ones that seem to be further in their actions than others. More value can be brought to customers, attractiveness is raised for new customers and a platform is a bigger process to change than just a mere software product.

Both dynamic software product lines and software ecosystems are good examples of the development of the original SPL method. The original software product line engineering will stay as a valued method for improving efficiency. Nowadays, reuse is not only done in own repositories of an organization, but

code is also reused from open source libraries. Software developers obtain private benefits from writing components and sharing their code, and collectively contribute to the development of software overall (Haefliger, Von Krogh & Spaeth, 2008). They also mention, that developers benefit from project-external components and reduce overlapping development significantly. Several studies concerning reuse also mentioned open source libraries as a possible resource of large-scale software reuse.

2.7 New product development

In this chapter, the term of New product development is explained and how it is seen in different organizations. First, the term is introduced in more of a general setting, and afterwards moving to IT perspective and its possibilities in achieving competitive advantage.

New product development is very closely related to software product line engineering. A large amount of the advantages achieved by using SPL is caused by increased efficiency and savings in the process of new product development (Van der Linden et al, 2007). In this chapter, the process of new product development is reviewed and it is approached from the point of view of software product lines. The term new product development is explained thoroughly, and the process of NPD is presented.

2.7.1 Defining new product development

New product development is defined as the process of bringing a new product to market. It includes idea generation and idea screening, concept development and testing, business analysis, prototype and market testing, technical implementation, and plans for product commercialization and launch. (Pavlou & El Sawy, 2006). According to Clark & Fujimoto (1991), new product development is a strategic process in which firms integrate inputs from R&D scientists, engineers, and marketers to jointly develop and launch new products. New products play a very important role in the ability of a company to accomplish competitive advantage. The results can contribute to the growth of a company and profitability and have a significant impact on them (Veryzer, 1998). Including IT in the process of new product development will help in data-analysis, enable more efficient communication and better problem solving, and achieve much higher levels of integration (Nambisan, 2003).

New product development is also a lot about innovation. There are multiple different possibilities for software ventures in terms of innovation and business model and product design. Two venture groups are presented here: Traditional startups that are funded and corporate software ventures that work as a part of large software companies but act as their own entity. There are various pros and cons for software ventures in new product development and software

innovation. Startups are more agile than large software companies, because their life-cycles are small, they do not have the same bureaucracy. Usually they can introduce simple and modern systems quite rapidly, as they do not have old legacy-systems and clients slowing their product development. They can also focus on a lot smaller part of the process and gain competitive advantage through that. (Käkölä, 2003).

2.7.2 The process of new product development

Over the years, the new product development process has evolved a lot, and will continue doing so. Various models have been suggested to manage the overall process of NPD, but the progression is similar in every one of them (Veryzer, 1998).

In the early stage of the process, the idea or the concept is evaluated, if a customer need and a market opportunity really exist in the case. After the product is seen as a potential new product the concept needs more refinement and examination of technical feasibility before design phase starts (Ulrich & Eppinger, 2000). Cooper (1990) has proposed a seven-phase process for the product from the idea to launch. He introduces seven stage-gates which divide the innovation process to predetermined set of stages. The gates work as quality control checkpoints that require certain criteria to proceed. Each stage aims at managing risks and increasing efficiency by bringing structure to the overall process and making the key questions in the beginning of the process (Veryzer, 1998). Even though there are many different approaches for managing the process, such as quality function deployment and value proposition process, the early stages of these processes remain the same – idea generation, preliminary market and technical assessment, business analysis and strategy planning. This all happens prior to the development of the new product (Cooper, 1990).

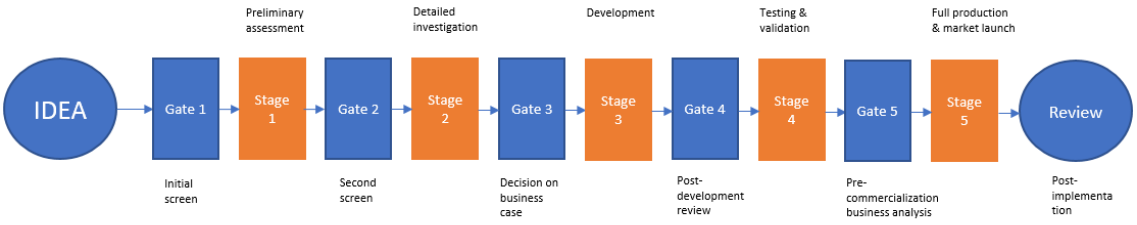


Figure 9 - The stage-gate system based on Cooper (1990)

Cooper (1990) describes the stage-gate system as follows The entrance to every stage is a gate, and the gates control the overall process. They work as quality control checkpoints for the production process. Each gate is characterized by deliverables or inputs, exit criteria, and an output. In this picture, each stage is more expensive than the previous one. Stage by stage, the outcome is better and risks are managed more widely. This sketch of the stage-gate system presents well the fundamental phases of the NPD process.

There is no doubt that product development is a critical cornerstone to success. In many industries, the existence of companies is increasingly determined by their success in new product development (Cooper, 2001). According to Griffin (1997), more than one-third of a corporation's revenue comes from new products that did not even exist five years ago. From IT perspective, this number is certainly higher in IT-related organizations. Ulrich and Eppinger (2000) posit, that new product development contains the whole process of starting with a perception of a market opportunity and after that, making the transitionto the production, sales, and delivery of a product.

2.8 Chapter conclusion

In this chapter, the key observations made in entire chapter 2 are presented. First, the theoretical views on software product lines and reuse were reviewed. As mentioned earlier, a product line is a set or a family of products that together address the product portfolio of a company. (Clements & Northrop, 2003). Component reuse is systematically reusing the existing parts of the solutions that already exist. Reusable components are seen as building blocks that can be used more than once to build a new system.

After getting more familiar with the terminology and the theories, software product line approach was differed to single system development and it was told, how it differentiates from other development strategies. Then, the business strategies and impacts were discussed.

An important part of the theoretical part was the evaluation of the benefits and shortcomings caused by product lines and reuse. The previous research offered versatile views on the matter, and there has been a lot of research on the matter. Also, the approach was discussed from the point of view of new product development – which is largely affected by component reuse and SPL. Component reuse is often seen as a part of software product line approach. Northrop (2002) states, that software product line practices often involve strategic reuse, which tells us that software product lines are as much about business practices and reuse of business components as they are about technical practices.

In Chapter 3, the terms capability and competitiveness are explained, and it is studied, how they are linked to software product lines and reuse.

3 LINKING SOFTWARE PRODUCT LINES AND COMPONENT REUSE TO CAPABILITY AND COMPETITIVENESS

In this chapter, the terms of capability and competitiveness are defined and their linkage to software product lines is viewed. The hypothesis is, that by using software product lines and implementing software product line engineering, organizations can improve their capabilities, competitiveness and efficiency significantly. To understand those impacts better, it is necessary to understand the reasons behind the phenomenon.

Earlier studies do not connect the impacts straight on capabilities and competitiveness that much, but more on single details that benefit the software development process. Many of these single details have an impact on capabilities of an organization. For example, the benefits that have been reported are shorter time-to-market, efficiency in development, common goals and shared vision within the company (Mansell, 2006, Clements & Northrop, 2003). The method's linkage to capabilities and competitiveness is easier understood, when the situation is turned the other way around. If an organization does software development without concerning systematic reuse of components at all, they are not performing as well as they could in the market. Same goes for product line architecture – SPL bring along a lot of good things, such as common knowledge of the architecture between employees and the ability to move personnel better between projects (Northrop, 2002). If an SPL approach architecture does not exist, the processes of the organization are not as clear as they are with them.

The expectations are, that the impacts can be both direct or indirect, and help the company towards a more successful future. The lack of studies on the linkage between the methods and impacts, and the possibilities to understand the way software companies exploit these methods support the need to study this subject from this point of view.

The structure of this chapter goes as follows: first, organizational capability is defined as a tool for success. Then, competitiveness is described and

linked to SPL and component reuse. After that, their impacts on capabilities and competitiveness of an organization are presented.

3.1 Organizational capability as a tool for success of an organization

As earlier mentioned, organizational capability is often referred to the ability of a company to manage their resources effectively and thus gain advantage over competitors (Grant, 1996). On the market today, this often means effective management of employees and following customer demands. Today, unstable market conditions are caused by innovation, intensity and diversity of competition. During a long transition, organizational capabilities have become the primary basis for organizations' long-term strategies.

Organizational capability is often measured with Capability maturity model (CMM). According to Paulk, Curtis, Chrissis and Weber (1993), it is necessary to design a path that increases an organizations' software process maturity in stages. By doing this, organizations can achieve lasting results in process improvement processes. The model itself has worked as a basic framework for the field for decades. The capability maturity model for software provides organizations with helpful guidance on gaining control of their processes for development and maintenance of their software. It also tells us, how to evolve towards a culture of efficient software engineering and excellence in management (Paulk et al. 1993). The idea is to guide software business companies in selecting process improvement strategies. CMM does this by determining the maturity of the current process and identifying the issues that are the most critical to the quality of the software and process improvement. Software process capabilities of an organization can be strengthened by focusing on a set of activities focusing on making them more efficient. This naturally also affects positively to the overall software process (Paulk et al. 1993). Lesser and Ban (2016) present three levels of maturity in software development:

- Foundational software organizations
- Intermediate software organizations
- Advanced software organizations

Whereas Paulk (2002) presents five different levels for Software maturity based on the capability maturity model. The levels are: Initial, Repeatable, Defined, Managed and Optimizing. In Figure 10, he demonstrates the levels and their differences.

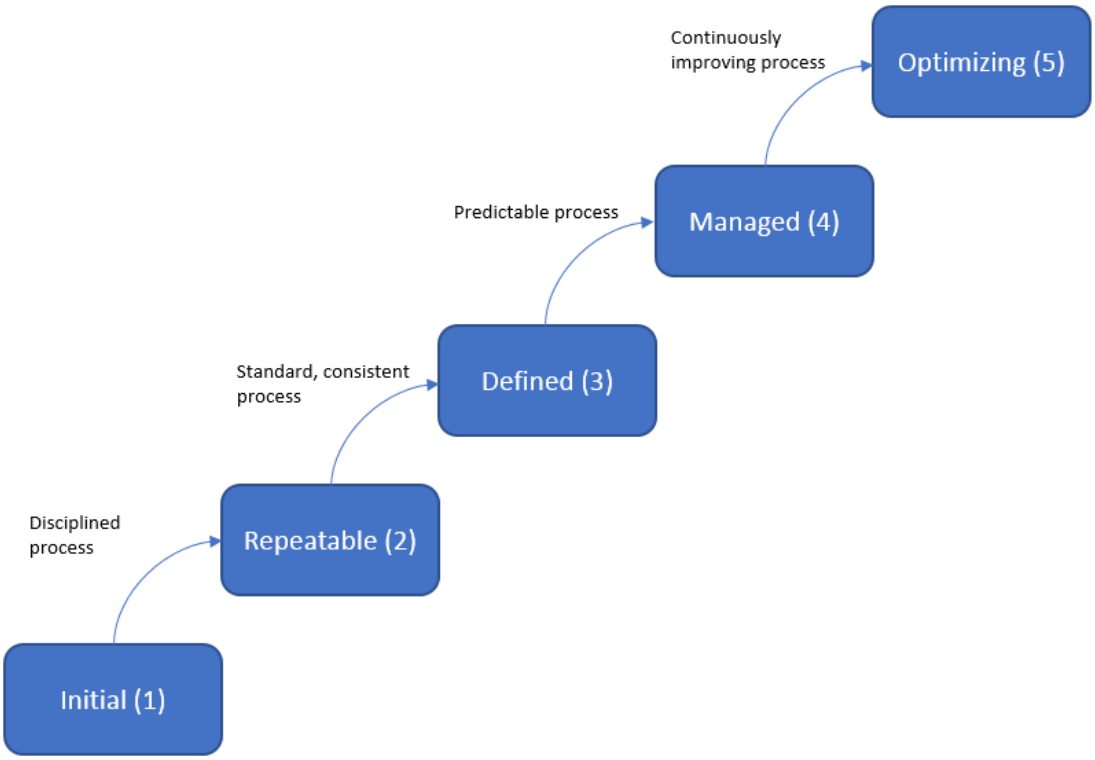


Figure 10 - The five levels of software process maturity (Paulk, 2002)

On Initial level, the software process is ad-hoc, and the processes are mostly undefined. On repeatable level, basic project management processes are established, and they track cost, functionality and schedule. It is possible to repeat earlier success from other projects. On defined level, the process is documented and standardized for both the management and the development. The development is more efficient. In managed level, detailed measures of the overall software process and the quality of the product are collected. The optimizing level means that there is continuous improvement in the development process, and testing is done to evaluate new ideas and innovations.

On level 3, the processes are defined throughout the organization. Paulk (2002) mentions also, that level 3 organizations are stable both in management and engineering activities. They control costs, schedule and functionalities through established product lines. Also, the quality of the software is tracked. Organization that operate on this level are on a good level of capabilities, as they understand the processes roles and responsibilities throughout the defined process. From this, we can conclude that on Paulk's scale the organization implementing software product line approach should be at least on the "defined" level to have the characteristics that the company needs to succeed in implementing product lines. Also, from these views, we can find a link between SPL and capabilities – organizations being on higher maturity levels usually have product line architecture in place and reuse is done systematically.

On higher levels, the usage of the product lines is evaluated continuously. Paulk (2002) states, that Capability maturity model and the maturity levels have an impact on the success of the development process. From this, we can conclude that the maturity level of the organization is connected to the success in implementing software product lines. Software product lines, on the other hand, affect the overall level of maturity. When an organization has defined product lines throughout the organization, it has an impact on the maturity levels, because this usually means that the processes are documented, and reuse is favored. To enhance organizational capabilities the companies must pay attention to their existing capabilities and resources.

The data collected by the company and its products is part of their organizational capability and is seen as an asset for the company. By using the data, the company is able to find solutions to their problems and make their processes more efficient. Therefore, data can be a major asset for increasing the capability of an organization. According to Holmström and Bosch (2013), data collection in information systems can be divided into Pre-deployment data collection and Post-deployment data collection. Data collection is more common, when the system owner also owns the data. If the data belongs to a customer, a company first has to ask for permission to use the data from the client and its customers.

In a way, component reuse and software product lines are reusing information. The amount of resources and information is combined into a component, which is then reused. The share of knowledge is seen as the most important part of knowledge management (Liao, Fei & Chen, 2007). By reusing components and implementing software product line architecture the organization spreads the existing knowledge throughout the organization. This raises the level of absorptive capacity in the company, which has a direct impact on successfulness and innovation in within the company. Absorptive capacity is seen as the organizations' ability to recognize the value of new information, to assimilate it, and apply it to commercial ends (Liao et al. 2007).

3.2 Competitiveness

Increasing competitiveness and gaining competitive advantage is part of every organizations' strategy. Yee and Eze (2012) define business competitiveness as "company business performance and competitive advantage in the marketplace". Only the ways to reach it are different. According to Lesser and Ban (2016), companies have finally started to realize that effective software development is crucial to achieving competitiveness – all the way from ideation to delivery. Stonehouse and Snowdon (2007) claim that every organization has the potential to improve the effectiveness of their software development practices and therefore increase their competitiveness. "Establishing an enterprise capability for accelerated software delivery can help differentiate companies in the

market" (Lesser & Ban, 2016). This is also the case with product line architecture – it brings more possibilities to the development and can make it faster.

The impact of capabilities to competitiveness is also emphasized, and the link between them is clear. The next figure shows how usability, flexibility, speed, global integration, reliability and business insights are seen to affect in the competitiveness of an organization. It seems, that advanced organizations are more effective at the development capabilities considered most important to competitiveness. (Lesser & Ban, 2016)

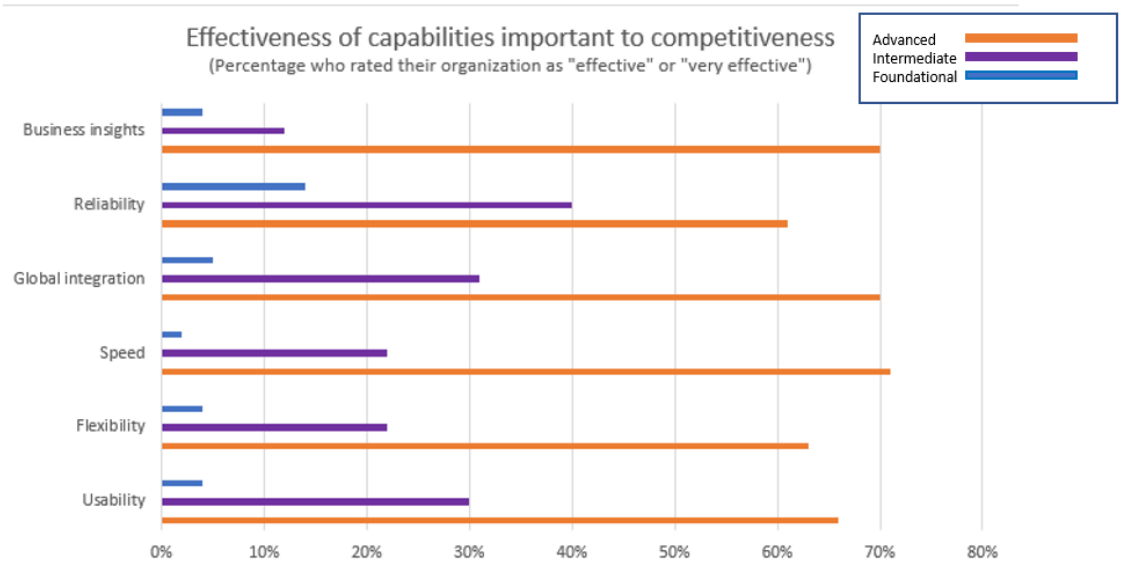


Figure 11 - Effectiveness of capabilities important to competitiveness. (Lesser & Ban, 2016)

Delivering competitive advantage in a firm depends on the firm's value chain and its ability to support the strategy in adding value to its products and services than the competitors. The value chain includes all the activities the firm does to make their final product or service better (Stonehouse & Snowdon, 2007). Naturally, implementing software product lines is also a part of a value chain, and therefore their impacts can also be seen in the whole value chain. According to previous research, core capabilities of the firm can be seen as a major factor for success and profitability of a firm. This, for example, has more effect on the profitability than the choice of the industry the firm makes. (Stonehouse & Snowdon, 2007).

According to Kusunoki, Nonaka and Nagata (1998), competitive advantage in new product development is achieved by concurrently achieving product effectiveness (quality and innovativeness) and process efficiency (time to market and low cost). Calantone and Di Benedetto (1988) have studied the actions and consequences in the NPD process, and found that great performance with respect to market intelligence and marketing give a relevant chance for the company to perform better and therefore be more successful with the new product. The organizations that take care of the single steps and pay attention to the factors that impact successful process deployment, manage better in the results.

3.3 Impacts

As mentioned earlier, the usage of software product lines and component reuse can have significant results on capabilities and competitiveness of an organization. Van der Linden et al. (2007) divide the impacts of product lines to product qualities and process qualities. Product qualities mean the characteristics of the product itself, such as the reliability and the usability of the product. Process qualities are the reduced costs and reduced time-to-market, for example. These are key aspects on growing the competitiveness of the firm and its capabilities.

Figuereido et al (2008) found, that organizations using software product lines tend to have more stability in design, especially when changes are done either to optional features. This is also an important remark and supports the positive view over the product lines.

Northrop (2002) states: "Organizations can benefit tremendously through product lines". Other research papers support this statement: Mohagheghi and Conradi (2007) point out positive impacts of software reuse, such as organizing the work of developers – more experienced coders can develop the reusable components and leave the easier tasks for less experienced ones. The valuable time of the experienced developers do not go to waste. They also reported that reusable architectures lead to clearer abstraction of components within the organization, and defects and errors in the software are significantly reduced. As mentioned in Chapter 2, component reuse and product lines are nowadays connected to open source software libraries. According to Sojer and Henkel (2010), developers of open source software perceive efficiency and effectiveness as the main benefits of code reuse. Both benefits are linked to capabilities and competitiveness of a company, strengthening the abilities of the company. Also, developers see code reuse as a way to kick-start new projects as it helps them deliver a first version of the wanted solution a lot faster, which lets them focus more on it as it can help to differentiate from other competing projects and keep the focus on the current project (Sojer & Henkel, 2010).

3.4 Component reuse approach

So far, we have processed the theoretical review of software product lines and its possibilities in the future. In the empirical part of this thesis, the impacts of SPL use on competitiveness and capability of a firm is approached from the point of view of components and component reuse. To understand the impacts and the reusable components better, an interview is made for companies doing business with different kinds of SaaS-products and offering software products for their customers.

In this study, reusable components are both technical components and business components. In the empirical part, it is also studied how often is it possible for SaaS-businesses to create services that fill different needs than the

existing products that already exist in the product line. More importantly, is the product line architecture beneficial for creating products for different markets. If this is the case, the firm is able to enter to a whole new market with significantly less costs. The aim is to find those components that usually are reusable even though the idea behind a product change.

As mentioned earlier, this thesis does not consider agile methods nor its impacts on the results. The focus is on product lines and common reusable components between different systems. Agile methods, however, are widely used and the method's success has taken a lot of attention away from product line architectures and systematic reuse during the years, but it does not guarantee better results than systematic component reuse. This thesis sees wide, systematic reuse as a solution for effectiveness in development and competitiveness.

3.5 Chapter conclusion

In this chapter, the main points of the entire chapter 3 are presented. The goal was to introduce the terms capability and competitiveness of an organization, and to find the links that SPL usage and reuse have to them.

First, the terminology was presented about capabilities and competitiveness. The capability maturity model was presented and a link from the maturity levels to succeeding with product line architectures and reuse was found. Then, the previous research views on data collection in information systems and the component reuse approach were briefly reviewed.

Overall, there is a link between product lines approach, competitiveness and capabilities of the organization. Next, in the empirical part, more answers to this are looked for.

4 CARRYING OUT THE RESEARCH

To support the earlier research, it is needed to make an empirical section of the study. In this chapter, the main points behind the research are presented, including the research method, background of the thesis and the goals. Also, the organizations participating in the interviews are defined. The results of the research are presented in Chapter 5.

4.1 Background and goals

The target organizations of the research are Finland-based companies producing IT-related products on-demand or as SaaS-products. To get a broad view of the matter, both members of the management and members of the developing teams were interviewed. Some of the interviewees were also technical people who had made their way from the development to management positions. They were also able to give thoughts about the matter from different perspectives, which gave a lot of good insights.

The background for studying this subject came from collaboration with a company that is actually implementing these practices, as they are working on bringing a new product to market to increase their turnover and grow the overall business. The idea came from their existing products and the technical infrastructure working as a base for the new product is reused from the existing solutions. Of course, the writer's motivation to this field of study played a large role – having worked in several IT-based firms, the writer is interested in finding new ways to increase overall efficiency in the firm, especially in the development. By developing their solutions more efficiently than others, organizations can seem a lot bigger than they actually are, when compared to others. Tools to increase competitiveness and capabilities in the firm are asked for, and there is still a lot left to discover. Therefore, this subject was chosen, and the semi-structured theme interview was selected as a method, to get new views from the people that are experts in their field.

As background for the thesis, a lot of research was reviewed and the aim was to make such an interview form, that would encourage the interviewee to give broad answers about the habits in the company and overall view of the suitability and future of reuse. The earlier research in the theoretical part revealed the main points of software product line thinking and the importance of reuse. Many views about the impacts of reuse were presented, and in the empirical part will highlight the impacts and hopefully find some new insights on the matter.

The goal of this study is to identify the level on which the interviewed companies execute software product lines and component reuse. Also, the empirical part aims at finding stronger evidence of the impact of SPL and component reuse on competitiveness and capabilities of the organization. By conducting an interview for different kinds of organizations working on IT-related products such as systems and software products, it is possible to characterize the situations when the methods fit for the organizations and when they do not. For example, project-oriented and product-oriented organizations might have different views about the usability of the methods studied in this paper. Also, new product development was talked a lot about and the aim was to find the motives that it brings to SPL and component reuse.

4.2 Presenting the research method

In this thesis, the empirical section was executed as a qualitative research, in which employees of different software business firms were interviewed to get a broad view of how organizations think about software product line engineering and are they actually implementing it. To reach as good and reliable results, a semi-structured theme interview is conducted.

A theme interview itself did not consist of precise, completely prepared questions, but more concentrates on primary themes that are picked for the interview. The interview can go deeper into some themes than others, and the themes and the overall structures are the same for each interviewee. (Hirsjärvi & Hurme, 2001). The interviewee is given the right to speak freely about the matter and questions are prepared to keep the interview on the right track, if needed. The chosen themes are processed with every participant, but the structure is not that strict. (Hirsjärvi & Hurme, 2001). This interview method was chosen, because the interviewees come from different organizations and positions. With theme interview it is possible to understand the interpretations and motives of the interviewees and their views can be understood better. This way it was possible to focus more on the themes they were able to give more comments on.

Theme interview as a method fits this study well. It works well, when it is not known what kind of answers the interviewer is going to get, or when the answers are based on the interviewees' own experiences on the matter (Metsämuuronen, 2005). The method is also used, when more information is

needed about the subject, even though there is earlier literature (Hirsjärvi & Hurme, 2001). Strength of this method is also, that there is no tight structure, but the interview is conducted more freely to capture the experiences and feelings that the interviewee has on the matter.

In the beginning of the interview, every participant got a short explanation of what is the main study about, and the meaning of SPL and reuse was explained through two core pictures in this thesis. The participants were aware of the subject of the study but did not know the questions or structure of the interview beforehand.

The interview itself consisted of three main themes: Software product lines, component reuse and their linkage to competitiveness and capability of the organization. This is a common approach in theme interview method. There were some questions in every theme, but they could change a bit, depending on the conversation with the interviewee.

The themes and the structure for the interview were built based on the results of literature review and the author's views on the importance of each section. The main themes of the literature review were put in writing – Software product lines, Component reuse and their linkage to capabilities and competitivity. After naming the main themes to the interview, approximately 5-8 questions were put under each theme. Some questions were mostly supporting the interview, if it was needed. The interview moved forward freely depending on the views of the interviewee – in many cases, the conversation continued without some of the questions in the structure very well. The main themes, questions and structure of the interview can be found in Appendix 1.

4.3 Data collection

The qualitative analysis was done by executing an interview, which was carried out with the interviewee during the session, which was done either as face-to-face interview or by phone conversation, if needed. The topics and themes were shared with the interviewee beforehand, but the final questions were new. As the subject of the research could mean different things to the organizations, the semi-structured theme left the topic more open and the data about different views could be gathered.

The organizations that took part in the interview, were chosen based on the following requirements: they provided SaaS-products or systems and they had at least a few different products or projects they were working on constantly. It did not make a difference whether the companies were implementing product line architecture or reuse at the moment. Also, SaaS-products and software projects were the main business of every company interviewed. Intentionally, both product- and project-oriented companies were selected. Three of the companies interviewed focused more on developing their own products and solutions, and two of the companies were offering custom projects for their customers. The interviewed organizations were mostly small- or medium-sized

companies. The interviewed companies, their orientations and main businesses are presented in Table 2. The amount of personnel in the interviewed companies varied between 20 and 80 employees.

Organization	Respondents	Business	Company's field of expertise	Personnel	Number of products/projects
Organization 1	Respondent 1	SaaS-products, software projects	Customer-centered software projects, testing and implementations	30	One SaaS product, many customer projects
Organization 2	Respondents 2&3	SaaS-products	Recruitment software, recruitment process tools	25	3 main products
Organization 3	Respondent 4	Software projects	Software projects, digitalizing customer's solutions	60	Multiple software solutions and projects
Organization 4	Respondents 5&6	SaaS-products	Mobile communications, mobile solutions	20	Over 10 products
Organization 5	Respondent 7	Software projects	Large-scale software projects, no specific field	80	Large amounts of customer projects

Table 2 - General information about the interviewed companies

The focus group was members of management in the company, but also few technical developers were interviewed to discover possible inconsistencies between the answers of the management and the developers. This way it was possible to see if the reuse is more of a thing that the management encourages to do, but the people working on it do not think it is executed in very high level. In some cases, the members of management used to work in the developing teams, so they knew very precisely what the situation in the firm is – from the developer teams to management – and how the views differed.

In total, the interviews consisted of seven interviews – and the interviewees represented the total of five different companies that took part in the study. The formed structure was based on the information gathered in the theoretical literature review. In the beginning of the interview, the participants were shown the main figures concerning the subject and told about the motives behind it. The basic information about the organizations' products were processed, and their mission in their field and other common information such as number of employees and annual turnover.

The three main themes consisted of software product lines and software product line engineering, component reuse, and the linkage between reuse and

SPL to competitiveness and capabilities of the firm. The aim was to achieve honest and reliable views on these matters and share thoughts about the phenomenon overall and its future. In the themes, I wanted to challenge the interviewee about the current ways they work or do not work with SPL and component reuse – to get in-depth answers about why they do it or why they chose not to. The important thing were the interviewees' views about the potential impacts of the methods and thoughts on how the impacts can be measured. These answers had a lot of value, since the measurement of the benefits and shortcomings is a challenging task. Considering the background and field of expertise of the interviewee, the thought was not to limit the questions only to the ones existing in the interview structure, but to follow the views of the interviewee and dig deeper on the main points they focused on.

The process of the interviews went as follows: First, the planned number of companies were selected, and contacted. Meetings were set up with each organization, and all the interviews were done within a three-week period. To capture and understand the main points brought up by the respondents, the whole conversations including the interviews were recorded. After all the interviews were done, the recordings were reviewed one by one. The interviews were transcribed basically word-by-word, to capture the respondents' views as clearly as possible. The interviews were done in Finnish, with the same questions and structure as the one shown in this paper. This decision was made to achieve as good results from the interviews as possible by making the interview as easy as possible for the interviewees. The quoted answers were translated to English, word-by-word.

What was also important to the questions, was to point out that by "components" one can mean both technical components and business components. Hopefully bringing this up in the interview it would expand the thinking of an employee – as mentioned in the theoretical part, business components can also be reused. It was an interesting task to try to solve, whether it is systematically and knowingly done or not.

4.4 Data analysis

In this chapter, the analysis and the process of data collection is presented. The interview structure and themes were built on the main topics of the theoretical part of this study. The aim was to support the theoretical part the best way possible.

The companies that took part in the interviews were chosen on several different requirements. All of them are IT service providers. They either provide SaaS-based services and/or custom products for their customers. At first, the idea was to find only companies offering SaaS-products, but during the process of making the theoretical part of this study, it became obvious that the project-based organizations could not be left out. To get a clear picture of the usage of software product lines, reuse and their impacts, both product- and project-

oriented organizations were needed. Also, the results and answers between these organization types could be compared to each other, which brought a lot of new insights to this study. The respondents were mostly members of the management, but also few developers were interviewed to understand the possible differences on the views between the development teams and the management. It is important to get answers from individuals representing different parts of the organization. The variety of different firms and the different roles presented in the respondents (CEO's, developers, managers, architects) that the different views are presented, and the results of the interview can be seen as valid and reliable.

All the interview results from each theme were coded and put on a same document to get a better view of the similarities and differences between the interviewees' answers. This approach enabled the thematic analysis of the results more clearly. As mentioned, the interviews were transcribed word-by-word, which allowed the usage of many citations in the results. The questions in each theme aimed at receiving honest answers to the research questions. The theme approach and the nature of the interview allowed the respondents to discuss the topics in their own words, free of constraints. The interviewees had quite a lot of insights on the subject, which resulted to a wide set of data from each interview. Assumptions and conclusions were made inductively from the results based on the data gathered.

The vision was to use quote their comments on the main themes and bring their views in to the study as they were. This also was made to ensure the validity of the respondents' views in the notes. This was important, as the views of the respondents varied a lot, and some of them had much stronger thoughts on behalf of or against the approach than the others. To capture the reactions and the respondents' true views on the matter, this approach was needed. By committing to the plans stated in this chapter, the quality of this research was ensured. The number of respondents could have been larger but taking in to account the size and subject of this study, the amount was seen as applicable.

5 RESULTS

In this chapter, the main findings of the research are presented and analyzed. The contents of the interviews and its main results are reviewed one theme at a time. This helps to understand, in which parts of the interview certain answers and notices are made.

Because this chapter consists of both the results of the research interviews and conclusions made from those results, they should be differentiated from each other. When a presented information is directly gotten in the interviews, it is always mentioned. Also, in some cases it is needed to mention, in how many interviews the specific information was brought up. The common way is to first present the answer, and after that to draw conclusions of it. In chapter 5.4, the content is mostly the writer's own conclusions of the research results.

In-depth theme interviews were an interesting part of the research. I was surprised by the level of knowledge about the matter and vision of the future. At this point I already want to thank each and every one of the interviewees for using their time to answer my questions. As mentioned earlier, the empirical part consists of seven interviews, representing five different companies.

Respondents and organizations are differentiated from each other by numbering the Organizations from 1 to 5 and numbering the respondents from 1 to 7.

5.1 General information and interviewees' backgrounds

Overall, the interviewees seemed to have quite good knowledge about software product lines and component reuse – even though all of them did not agree that the methods are used in the company. The interviewees had different kinds of backgrounds – in every case more or less technical. The people that took part in the interview worked for example in following positions: Senior Software Developer, CEO, Software/System Architect, Business director and software development team lead. The organizations interviewed were small- or medium-

sized companies, three of them having 20-30 employees and two of them having over 50 employees in total. The companies offered services such as SaaS-software solutions, on-demand software solutions and development resources, marketing automation software, mobile applications and system modernization.

Every interviewee seemed to have technical studies or engineer studies as a background, but only four of them mentioned having processed software product lines in their studies. Common answer was that it was mentioned quite many times during studies, but not went deeper into. Mainly the things learned about SPL and reuse have been in different stages of working career. However, every respondent mentioned, that the methods software product line and component reuse are familiar, and they have been used either in their current organization or the one before.

Six out of seven interviewees answered, that component reuse is closely related to their daily work. By this we can point out, that component reuse is commonly known and used method in different kinds of software businesses.

From the organizations interviewed, two of them offered only SaaS-products and their business was based on products more than custom-made projects. Two organizations interviewed mentioned customer-centered projects as the main source of revenue, and one did both SaaS-products and customer-oriented projects. One thing worth mentioning, is that already in this phase it could be noticed that organizations offering multiple SaaS-products and their business being more product-centered, felt that software product line architectures and component reuse is right at the core of their business. Organizations that leaned more on custom-made projects for the customer, did not have as strong beliefs on SPL or reuse being a part of their daily work and business. This clearly tells us, that product-centered organizations seem to benefit more from software product line usage and component reuse, than organizations offering custom-made projects for customers. This naturally has an impact on the beliefs they have on the methods.

Before moving to software product lines, the respondents were asked the following question: "Would you agree that the architecture in the products would be suitable for entirely a new kind of software service?". This question suited the product-oriented organizations better, but we got some good answers from the project-oriented organizations as well. The more product-oriented organizations 1, 2 and 4 were optimistic about possibilities in other fields with the same architecture. The product-oriented organizations 3 and 5 did not see changing the field as an option. Here are some views on the question:

> Respondent 4: "No, we could not do that. All of the final solutions that we provide to our customers are unique. In many cases, we could not even make the solutions work for another company working in the same field – the solutions are made to match the customer's business logic, data warehouses and technological implementations"

> Respondent 2: "It is possible, and we have thought about few different scenarios, but did not make the decision to go further with them, at least yet. This is something we should still consider as an option to create new business"

> Respondent 6: "Yes, they would. Today's reality is, that the customer's needs change rapidly. As a provider you must react to these changes. It has a major impact on our productivity as a small firm of 20 people. We have solutions in reserve, that we can use with the new products we come up with. This is architects' and coders' job these days."

From the answers of the respondents, we could conclude, that very generic solutions can be reused, but they do not bring the best value. Custom-made projects are very hard to be used on totally different areas of business, but common SaaS-software products seem to have that possibility according to the respondents.

5.2 Software product lines

The term *Software product line* seemed to be quite familiar to the interviewees, but most of them agreed, that it was helpful to process the basic idea behind it before going deeper into the interview. It helped them to go deeper into the subject and understand what was meant by SPL in this paper. Two organizations out of five answered that they have a formal software product line architecture. Organization 2 has three SaaS-products in their product line and Organization 4 has over ten different products, every single one of them being a part of the same product line. In these organizations, both the management and developers seemed to have the same thoughts on the matter, which is proof of the fact that they have planned this together and the SPL method has been transparently taken into use in the company.

> Respondent 5: "We have declared this with the management, that all the products are kept in the same product line. Our software product line has over 10 products, and each product is seen as a smaller product line inside the original one."

Both of the companies having a formal product line agreed, that it takes a lot of time and resources to establish a product since it has impacts on the whole organization's way of working: how development is done, what can be done within the architecture and how new products' developing process is done.

The respondents were asked, if they see SPL as an important part of development. The answers were both supporting its importance and saying it is not necessary. Five out of seven respondents saw this as an important part of development process.

> Respondent 1: "I would say it is important for us. Especially when producing multiple products. It helps to understand the bigger picture and it can be divided more clearly into smaller parts."

> Respondent 2: "Extremely important. Software design nowadays is more and more about combining functionalities than just coding. SPL as a method reduces flaws and errors in the code."

The project-oriented organizations were clearly the ones that disagreed with the importance of SPL. They did not see it as a backbone of their development.

> Respondent 4: "In project-oriented organizations I do not see it as important. It is about so concrete components, that implementing strict product lines is challenging. We are more about reuse and external libraries."

As a conclusion from the answers to this, it seems that many of the organizations have moved on from strict product lines to using agile methods, in order to respond to the changing customer needs faster. However, none of them said that implementing SPL would not be helpful at all. It has its flaws, but when implemented on the right level it can boost the productivity a lot. We are drifting from product lines to mixing reuse and agile methods. Respondent 7 also mentioned, that they tried using strict documentation and product line approach – but the results were not as good as expected. The developers that benefitted from this were the ones that made it themselves. The other teams could not use their work couldn't benefit from it as much. Also, the type of the produced product has an impact on the effectivity of the method.

Systematic use of product lines also received various answers. Two organizations mentioned that it is done systematically from the start to final product. The usual answer to this was that it differed between the development teams, but all of them agreed that it would be better if the way of working was the same throughout the company. Systematic or not, the respondents seemed to agree that it is beneficial to have a common view of this in the company. The method does not bring as much results, if only part of the organization is on board.

One of the most interesting questions in the interview was "What are the potential benefits and shortcomings of the SPL method?". It also gained a lot of answers and thoughts by the interviewees. First, the mentioned benefits are reviewed and the study continues to the shortcomings after that.

The one benefit that every interviewee mentioned was the efficiency that SPL brings – creating something new becomes a lot faster. Having existing architectures to start with speeds up the process of new product development significantly. Here are few examples that the respondents mentioned:

> Respondent 1: "The benefits are clear. Creating new solutions becomes faster. We have something to begin with. Let's say we have to produce a proof-of-concept product for a customer in order to win the case for us. By using software product lines, we can create prototypes for the potential customer really quickly and look a lot faster and bigger than we actually are"

> Respondent 2: "We would be nowhere near the situation we are in now with the new product, if we chose not to use the existing architectures and platforms. We had them already, and more importantly, we had the know-how already – we were familiar

> with the technical solutions. If we used open source solutions, for example, we would have to do the studying first before starting to work."

In conclusion – for small companies that are already low on developing resources, software product line architecture can bring them flexibility and efficiency when starting new processes.

Second benefit that was mentioned by the interviewees, was the reached benefits in testing and quality control. When the components are already familiar and they have proven to be beneficial, it does not take time to undergo them in terms of testing and seeing the quality of the code. As Respondent 3 put it:

> "When the components are used several times already, the developer can rely on the code - They have the knowledge about what it can and what it can't do".

It seems, that especially the technical employees seem to value the fact that the code is similar and does not require studying in order to use it. This might be one of the things that management fails to notice – they see the potential in moving workforce easily from project to another but cannot relate to the feelings of a coder.

On the benefits side, it was also mentioned that SPL brings stability to the developer teams and keeps ongoing things in sync. The people working on the solutions are familiar with the architecture. Respondent 7 mentioned also, that there is only one product line to be managed and a main way of doing tasks in every team. This brings also common rules, common release dates and common testing protocols. All of these save the working hours of the workforce to something else more important. As also mentioned by the interviewees, for the developers it promotes efficiency a lot when there is less things to worry about. By reducing the amount of different tasks, the developers have to worry about, they are given the peace to focus and concentrate on their main job. Efficiency is promoted also in the situation where the product line of the company is commonly known by every employee. It reduces the time needed to check different things or asking for permissions from others, when there is a one known way of proceeding with a new product, for instance.

It was also brought up, that naturally the key employees in the project have a large impact on the final outcome. Respondent 2 mentioned:

> "In terms of product lines, product managers have a large role in making the decisions, whether to proceed with existing solutions or create something new, which always demands more resources. Also, it is important to make sure, that when new parts are produced, it is produced in a way that also our other products might benefit from it. If the products grow too far apart from each other, we might lose the synergy."

Even though the project-oriented organizations did not see product lines as beneficial in their business, they mentioned some benefits also. Respondent 4 mentioned the following:

> "The benefit of a software product line is the fact, that especially in larger organizations it brings ability to transfer people between products and between projects. There are organizations, where this is not even nearly possible – the teams do as they see best."

In these situations, when movement of workforce is needed to finish a certain project or sprint – a common way of working and proceeding saves a lot of time. There is no need to familiarize the architecture or solutions made – the employee knows the basics already. This naturally raises the level of capability of the organization in the form of internal knowledge. This can be a key characteristic of an organization that it can rely on. If there is a project or a large change in the product that needs to be done as soon as possible, they can grab backup from the know-how of the whole organization.

Alongside the benefits, quite many potential shortcomings were also mentioned. It was clear, that some kind of shortcomings might also follow by implementing software product lines. Some of them were clear, and some of them were practical observations that the earlier research did not mention. These observations were usually made by the developers themselves, when they were asked about the product lines in practice. For example, Respondent 1 mentioned, that implementing too strict product lines, the work of a normal developer might change significantly.

> "A coder wants to be innovative and have freedom to come up with new ways to execute different solutions. By implementing too strict product lines, the coder's work might become just adding existing plug-ins to the current solution. This can reduce a normal coder's enthusiasm towards his/her job"

This was a very interesting view. Decreasing the software development's satisfaction towards their jobs is not something you want to do. Other respondents did not mention this as strictly, but the views of the developers were in line with each other. This should definitely be considered when implementing a product line – by making it too strict, you reduce the freedom of developers.

Several respondents also brought up, that the negative side in implementing SPL's can be the limitation of agility. Many organizations promote agile methods, and some might see the common infrastructure as a barrier to respond quickly to changes and hopes that the customers present.

> Respondent 7: "Implementing product lines might cause the development teams to be less agile. Small changes in the user interface might end up being very big because of the common infrastructure between products. If the product line has many products in it with different maturities, challenges in development may occur"

> Respondent 5: "We can promote efficiency by using software product lines but sometimes it also limits us. The customer might want very specific custom solutions and we might not be able to produce them. It is a challenge."

It was also mentioned, that sometimes the customer is just not fit for the organization, if product line architecture is implemented – especially in the case that customer wants a lot of custom solutions.

> Respondent 5: "We might have a 20000-dollar solution that fits their needs as it is, and it can be delivered right away. But if the customer wants a high-end solution that is worth 100000 dollars, we might not be the ones that can provide it. Some projects are just not for us, if we see that the customer would not be happy with our product – we say it out loud."

Products usually have different maturities, and it might be challenging to see beforehand, whether using the common infrastructure is a good idea or not. It might slow down the process and bring problems to the development. Respondent 5 also mentioned, that implementing the software product line architecture took time for the firm to get used to but paid off in the long run. It takes a lot of effort and some long internal conversations to get past the obstacles in the beginning.

As a problem in product line utilization few respondents mentioned, that there is a possibility that the organization gets distracted and the development starts going in the wrong direction. To avoid this, someone needs to look at the bigger picture there and make sure to detect if there are problems. Respondent 2 and 4 mentioned, that if the development starts going to the wrong direction for too long, it is hard to correct it. The IT field develops rapidly, and the good old solutions might not do the trick anymore. Changes are needed. If the organization holds on to the old components for too long, problems may occur. The architecture changes so much, that they are not a fit anymore at some point. In those situations, sometimes a better solution exists in open source libraries, but it is hard to let go of the old components that have worked well in the past.

Concerning product lines, the respondents were asked if the utilization of product lines is acknowledged, and if the process is documented. In many respondents, this question put a smile on their faces. This was clearly something that they should do, but it is not done well enough.

> Respondent 1: "Product line usage and reuse is acknowledged, at least partly. When we start a new project, it is clear for everyone that we do not start from scratch. Nobody wants to do extra work if the solutions already exist. We do not have a written guide on our process with this, it varies between our development teams. It has become more of a habit, than a strict rule. Our experiences have taught us that it is the right thing to do."

> Respondent 3: "It is acknowledged, and we are encouraged to it, as we do not have too much resources. Documentation, however, is in a terrible shape. It is outdated and it would require a lot from us to fix it."

> Respondent 4: "To us, the code itself and the documentation are extremely important. Strict product lines are not used, but through good documentation it is possible to reuse or at least copy and modify earlier solutions to the new ones."

It seems, that majority of the organizations are not satisfied with the level of documentation. The problem is to find the resources to fix it – all of them seem to be in a hurry with the current development tasks. In order to be "product line-ready", this is something that the organizations should do to reach the maximum potential that the method can bring. Several respondents agreed, that this could bring long-term results.

An interesting part of the interviews was also to have two participants from the same company to answer the questions – in this case it meant interviewing the CEO first, and then a software developer or architect. The interviewees were not aware of the answers of others from the same organization. As observations from the interview results, when comparing their results, it seems that the management seems to think that they have more of a common development process and there are widely-known product lines. The developers agreed that this was the thought, but they did not see it functioning on as high level as the CEOs did. This is probably a cause of the differences between development teams. For an organization attempting to implement product lines, these results highlight the importance of communication between the management and development in the organization. Another remark was that the management seemed to think about product line from a less technical point of view, which is quite usual. They saw less technical difficulties on the approach, and the technical problems were brought up more by the developers.

5.3 Component reuse

In this chapter, the theme component reuse in the interviews is covered. In the beginning, it was made clear for the respondents, what the main themes are. This way it was clearer for the respondent, what is coming up – for example, it was important to mention for the respondent, that first, the SPL methods are talked about and the different theme component reuse will follow. This avoided mixing up the terms and understanding the difference.

The participants were asked, how component reuse would relate to their jobs. Various answers were given, but as mentioned earlier, most of them saw component reuse as an important part of development and business. For example, following answers were given:

> Respondent 1: "Basically everything on user interface level is made reusable, if possible. When modules are made independent and they do not rely on customer's business, they can be reused."

> Respondent 3: "Reuse is extremely important for us. When we are developing new products, the existing platforms and architectures are gone through first. They are taken into use if possible, to avoid extra work and to save resources."

> Respondent 5: "As we are a small organization, we have to constantly come up with solutions to enhance efficiency. This is where reuse and product lines come into the

> picture. We try to exploit reuse constantly. Duplicating the existing solutions is right at the core of our business."

> Respondent 4: "Reuse is a bit complicated in project business. You can reuse, as in do things the same way as before and gain advantage through it – you still have to remember, that the technological solutions must be up-to-date to the latest trends, or you are producing new products with old methods."

Each respondent replied, that reuse is done at least on some level. For project-oriented organizations, the term product line seemed "too heavy". Reuse is something that is done automatically in the organization, and is more of a habit, but software product lines demand more – systematic planning and long-term commitment. The overall process of committing to the process demands a lot from the organizational culture and strong commitment to acquiring the needed knowledge, skills and motivation to launch and maintain a SPL. In addition, as mentioned in the text, it demands financial investments and resources in the beginning to establish and implement the method to the whole organization. From this, we can draw a conclusion that it is a large decision to start implementing software product line architecture. Systematic reuse might be the first step to it, in order to see the results in smaller scale before implementing full-on product lines. About their experiences on the matter, respondent 5 mentioned the following:

> "Nowadays the whole organization is committed to the method. We have been using Product line methods for almost ten years now. Things proceeded slowly at first, we still did too much individual work for each customer. We kept pushing towards more generic solutions that could be reused, and it is also part of our strategy. Implementing it has been a long process."

This statement supports the earlier made conclusion about the fact that it takes a long time for the organization to get used to software product line architectures and to get proof of results.

The respondents were asked, if the reused components are seen only as technical components. This brought a lot of ideas and new perspectives to the conversation. All the respondents agreed, that reuse can be done and is implemented with business components too. A lot of different examples were mentioned: reused business components can be project plans, processes concerning new customers, marketing plans and business plans, for example. Many respondents mentioned that by using common document templates for making such processes, a lot of time can be saved by having consistent ways to proceed with them. Respondent 4 mentioned that actually the business side is actually the environment where they do reuse the most – the written procedures, instructions and reporting are examples where their organization does reuse. It reduces the risk of mistakes made in the process. Respondent 7 answered:

> "Of course, business components can be and are reused! A good point. We do it a lot, and I think other organizations do too, at least on some level. When a new project

> starts, we have the same checklists and document templates waiting for the project manager."

Also, respondent 5 pointed out, that the quality of their project plans and reports got a lot better after using the time to make common templates and instructions to follow. It seemed, that the respondent saw less risks in reusing business components than technical components, as the reuse of business components seems less definite. Respondent 2 mentioned:

> "Reusing both technical components and business components brings trust to our own products and skills, and both are reused in our company."

All the organizations offering only custom-made products for the customer mentioned, that implementing reuse is often challenging, as many of the solutions' Intellectual Property Rights (IPR) belong to the customer. This is usually the case when the software development is bought from another company – the solutions cannot be used for commercial use by the supplier. Respondents 1 and 7 had quite similar views on this matter. Respondent 7 mentioned, that the more the offering goes to producing services than background-systems, the more there is possibilities for reuse. Respondent 1 mentioned the following:

> "The systems are almost always the customer's own, as they own the IPRs. This limits our possibilities concerning reuse, but we still do it whenever it is possible. We exploit our own modules and components that are generic and save us resources. Especially open source libraries are used during development."

The IPRs might make the reuse a bit more complex process, but it is done whenever it is possible. The developers do not want to do the same solutions twice. Especially, when starting a new project, the base modules and components can be reused, even though the customer would own the IPRs of the final product.

Concerning reuse, it was also mentioned, that in order to keep up with the latest trends, reuse is done but in shorter time frames. Architectures can change quite fast, but when a company is large and does various projects at the same time, the same solutions can be used in different projects. Concerning this, respondent 4 mentioned:

> "The implementations of the same era are alike. This means, that the solutions you make this year, can follow the same patterns. The development can be done with same technologies and by using the same languages and libraries. However, the field develops rapidly, so you must be ready for changes. The final user-interface solutions always depend on customer's hopes and demands."

From this, and the results of earlier research, we can conclude that in constantly changing environment, executing large software product lines might not be the

best solution, as organizations must be able to change rapidly to new demands. Therefore, the solutions that would work the best would be something in between software product line approach and agile methods. This was also mentioned by few of the respondents, and one of them called it as "a hybrid-solution mixing agile methods and software product lines".

The project-oriented organizations might not have found the product lines as the best options for them, but they admitted that reuse is done by everyone in some form at least. For example, project-oriented organizations mentioned, that in short projects the same pattern repeats itself and it is easy to reuse components in those situations. Some plugins such as logging, and infrastructure management are easily reused from different projects. They are not the main solution, but important supportive tasks. If a component is needed that does not exist in the firm, it is searched for in open source libraries. Respondent 4 mentioned:

> "For us, strict product lines do not bring the best results – in project business, it is more about the tools and libraries that you use. The basis is, how and with what kind of tools you work with. The way the solutions be do are built, is reuse. It comes from the knowledge we have on the platforms that we use and the experiences with different open source libraries. So, it is more of a "mental capital" that is increased.".

Overall, the views of product-oriented companies and project-oriented companies seem to vary a lot about this subject. While product-oriented organizations tend to talk about SPL and reuse more positively, the project-oriented organizations see a lot more problems in utilizing them. Partly the explanation to this might be the fact, that in product-oriented businesses the organization itself has more possibilities to decide the process themselves, and they own the final product. As companies nowadays are more focused on the customers and their needs, both product and project-oriented companies try to answer to the feedback and requests as well as possible. Still, the product-oriented companies have a bigger influence in the details that are made to the final version.

According to the interviews, the role of reuse is still seen as strong in the future. The style of using it might change, but it will stay as a way for organizations to make their processes more efficient. Respondent 7 mentioned in the end of the interview:

> "From my point of view, the fuzz on software product lines and reuse calmed down as the agile methods became popular and it became a trend in the industry. But now, after many years, reuse and product lines are making its way back, as the open source libraries are used widely. Large organizations in the field can offer a lot of help to reduce the time spent on development, if the developers can find the solutions already from the libraries."

Other respondents also mentioned the component libraries which will continue to grow bigger. In the future, they are even more used and the libraries can be huge time-savers, in terms of resources. For some of the interviewed

organizations, this is already the case, and some are building towards this. Respondent 6 mentioned:

> ” Software architecture work and development nowadays is about how much we can exploit the existing solutions. For us, reuse means using the provider’s components or ready-to-use services. We are not talking only about component libraries, but also about large-scale service libraries.”

Other interesting views were also presented by the respondents. Respondent 6 mentioned, that working environment for developers changes, and it is not that much about software design anymore – companies use existing solutions and make their actions more efficient that way. Also, the amount of integration will grow significantly, and the companies must have standardized ways for it. The existing capacity is used, and solutions are combined for a better outcome.

By this, we can conclude that reuse will keep its role as an important part of development in many organizations. Overall, the vision was, that even though SPL utilization is not for everyone, reuse at least is. The change happened mostly in the way the organizations do reuse. As mentioned, we no longer talk only about our own components, but also about larger open source libraries and libraries provided by leading IT-organizations in the world. The used libraries naturally depend on the platform that is used. Respondent 5 mentioned, that their transition from having own virtual server environments to new technologies, that uses Microsoft Azure as a platform, brings a lot of new possibilities. They can exploit complete solutions from Microsoft’s libraries and it saves a lot of resources in development and maintenance. Respondent 4 also mentioned, that they are using many various libraries in different projects. Of course, it takes some time to get familiar with them, but after that is done, it opens a lot of new possibilities. Respondent 2 brought up, that having common way of working with the libraries and documentation is the key for a widespread exploiting of reuse. Otherwise, the usage is more time-consuming.

5.4 Linkage to competitiveness and capabilities

The results of the research support, that product lines and component reuse have an impact on organizations’ capabilities and competitiveness. The respondents all mentioned positive impacts in capabilities and competitiveness, but there were also some shortcomings mentioned.

A commonly mentioned benefit from product lines and reuse was the ability to move resources and workforce between projects a lot easier. Employees in different teams and projects are aware of the work that the others are doing and know the mutual habits and the way software is developed. This develops the capability of the firm to focus more resources on a project that demands more input. On challenging circumstances, this can have a crucial impact on the

competitiveness, especially if there is a Proof-of-concept product to be done in a competition to catch a new client. In the long run, this can have a major impact on the annual turnover of the company.

Respondents also mentioned, that as it is extremely hard to find new skilled developers from the job market, the organizations are forced to make the developers' work and environment as productive and efficient as possible. Reuse and product line approach bring certain rules to the developers and they know how to proceed in the process. Common way of working also gives the developers and architects a chance to focus on what is essential. Respondent 5 mentioned:

> "Software product lines definitely bring large advantages in the long run. We have one large product line, which divides in to smaller product lines within. Each product has its own, and the solutions in the picture are as generic as possible. It brings flexibility to us and it is easier to control."

Most of the respondents agreed with the positive impacts on both competitiveness and capabilities. Some highlighted the importance, some saw it as a supportive action to boost the performance. Respondent 1 mentioned the following:

> "It is not a prerequisite for success, but it definitely helps. For us, this would be something we could concentrate more in, to boost our performance."

The impact on the capabilities is mostly about the boost in the ability of an organization to manage their resources. To reach efficient results, company's organizational capabilities must focus on the ability of their business to meet customer demand. As mentioned earlier, the easy movement of workforce between projects is a key to flexibility and gives the organization a capability to react to changes faster. This was mentioned by every respondent.

Alongside the mentioned impacts, also the developers interviewed pointed out, that the usage of SPL and reuse can also develop the know-how of each developer in the firm. An interesting remark was also, that the widespread knowledge about the bigger picture could raise the level of developer's satisfaction to their job. Respondent 3 mentioned:

> "Through utilization of it we could develop our own skills. The developer does not fall into one slot but knows what is going on in the other projects too and how things are done there."

Many respondents pointed out, that it is important for the developing teams to also understand the bigger picture the organization is working on. This reduces the number of faults in the code and developing teams can better support the management with their expertise. Also, when everyone understands the architecture and limitations that there is in the software, it is easier to make decision how to proceed. Respondent 5:

> "We have gotten more results and better competitiveness through these methods. It is in our business one of the biggest advantages in the competition. We could also add some lean way of working to this, by tackling the bottlenecks and factors that make us slower."

Usually less faults in the code means better quality. Respondents 5 and 7 mentioned the quality of the results to be the biggest impact that reuse and SPL usage have on capabilities and competitiveness. According to respondent 7, the biggest benefit is the quality, especially on a time frame of 1-2 years. Also, better quality leads to better trust on the organizations own products. This was mentioned by the respondents in both technical and business components.

Business components were also mentioned to have a positive impact on the competitiveness and capabilities of the organization. Respondent 4 mentioned:

> "For us, the main thing to reuse is the knowhow and knowledge that we already have. Why do the same things more than once? By reusing the business components, we can focus on the essential. We save resources and money. That is what gives us competitive advantage."

As an important part of the interview structure there was a question about how to measure the impacts of the actions taken in SPL and reuse. All of them agreed that it is actually very difficult, as the results are factors that affect in the long-term time frame. Therefore, it is hard to put something completely under the impact of product line approach – it is not easy to identify all the factors that have affected the outcome. The respondents had many good measurement tools to perceive the results. The most common answer was to measure the time it takes to bring a new solution to market, if there are several products in the product range. By committing to software product line approach, organizations are able to lower the time-to-market of the product. This was also mentioned earlier in the pros of the method. Respondent 2 answered:

> "We could measure the impacts in following the work estimates in the long run and comparing different scenarios – for example, how much this would cost us in resources to build this from scratch or how much it would cost to buy this solution from somewhere else, compared to leaning on reuse."

Also paying attention to the times when the work estimates hold instead of running over the limit, is a good way of measuring the effects. When the organization avoids unwanted surprises by being more accurate in the estimations, it brings trust to the ongoing workload. An interesting outcome mentioned by two of the respondents was, that because of reuse smaller companies are able to appear bigger than they actually are. For example, surprising the potential client by having a first version of the product fast. Appearing bigger to the current and potential customers can have a huge impact on the ability to win bidding competitions and new customers to count on you. On the measures, respondent 6 brought up:

> "I think you can see the impacts directly from the level of customer satisfaction. Through product lines we were able to raise the level of our service level agreements and the scalability our solutions got a lot more efficient."

Especially when delivering B2B solutions and SaaS-products, the level of customer satisfaction gets a boost if the organization is able to keep up to its promises and possibly even overdeliver. Better customer satisfaction means better service and less mistakes. This is also in line with the response of respondent 7:

> "The impacts can be measured by following the number of mistakes made in the development process. We noticed a significant impact on this ourselves. Also following the quality of documentation works as a tool for measurement."

The interview also has a question about how the organizations themselves have proven the benefits of reuse to be true and do they have any data to support it. The usual answer was, that it is hard to verify as there are a lot of other factors that affect the outcome. Nevertheless, the respondents had quite many answers to this. Respondent 1 mentioned, that many of their projects have shown impressive results through reuse and the projects were finished in shorter time than expected. Also, short-term differences in efficiency during certain sprints were noticed. The project duration was mentioned by many respondents, but it was also pointed out, that often the customer also impacts the project completion. Overall, the measurement and data questions received quite similar answers to the questions about documentation. They were not happy with the level it is done at the moment, and few of them had already plans to measure the impacts more closely. Respondent 5 mentioned:

> "We have not measured the impacts well enough. That is actually a problem we are tackling – the actions of sales is measured, but not the technical work."

Also, respondent 3 mentioned, that investment estimates should be made and comparison should be done between the two options – either reuse our existing architectures or buy the solution from somewhere else. By following the actual costs that occurred, we can also evaluate the accuracy of our estimates and see, if the option that was chosen was the right things to do. Respondent 4 mentioned, that the link from systematic component reuse to organizational capacity is the situation where the developers and the management share the same vision for the product and its future.

All the respondents found positive impacts on the organizations' capabilities and competitiveness. Also, many good perceptions on the measurement possibilities were made, even though accurate measurement has its difficulties.

6 DISCUSSION

In this chapter, the main findings of the research are reviewed and discussed. As an observation of this research process, it seems that the organizations and interviewees were able to bring value to this research. The idea was to get organizations that have differences in the way of working and some differences in the business itself. By having various organizations participating, better reliability was ensured. The following table presents the organizations and the respondents in them, and their views on SPL approach and component reuse.

Organization	Respondents	Business	Views on SPL implementation	Views on reuse
Organization 1	Respondent 1	SaaS-products, software projects	Positive	Positive
Organization 2	Respondents 2&3	SaaS-products	Very positive	Very positive
Organization 3	Respondent 4	Software projects	Negative	Positive
Organization 4	Respondents 5&6	SaaS-products	Very positive	Very positive
Organization 5	Respondent 7	Software projects	Neutral	positive

Table 3 - Interviewed organizations and their views on SPL and reuse.

The process of the empirical part and the interviews went as planned. First, the main frames for the interview were planned. After that, the search for potential organizations to interview started. The chosen organizations had a lot of interest to participate, so there was no need to ask for a lot of companies if they wanted to join or not. When asked to take part in to the survey, the companies were told that 1-2 employees of the firm will be interviewed, and the main themes of the research were presented. After this, the final questions were finished, and the interviews started. All the interviews were recorded and tran-

scribed word-by-word. The results were collected in to one document and compared to another. After this, the main results were presented in this chapter.

As mentioned earlier, the product-oriented organizations offering SaaS-products see both SPL and reuse as more effective than project-oriented organizations do. As explanations to these views, the respondents pointed out the problems in IPR's, as the produced solutions belong to the customer. Some generic parts could be reused, but the organizations offering custom software solutions seemed to think that the solutions can not be applied to other clients, as the final solution is custom-made for the customer from the start. However, even though they did not see strict SPLs as solutions for them, they agreed on the benefits of reuse. The generic parts of their solutions could be reused, and it helps in the overall development process. Product-oriented organizations seemed very happy with the SPL approach and were utilizing it. They saw it as a way for them to appear bigger than they are by making the development process significantly more efficient. This means, the usage of product line architectures and reuse will stay as a tool to increase competitiveness especially within the product-oriented companies.

Some differences in the opinions between development and management on SPL and reuse were also noted – both in the utilization at the moment in the firm and in the thoughts towards them. The reality of the current situation was presented more thoroughly by the developers. This is because the developers themselves have a better vision of the working habits within the development teams. They seemed to bring up a little more negative sides of the method and saw the implementation more realistically, when compared to the managements more of a "big picture" view.

The systematic reuse of components was seen to reduce the consumption of development resources. This gives the smaller companies a chance to compete in pricing with the larger service providers. Flexibility in pricing gives them the opportunity to attract more customers and grow their businesses. When compared to the companies that start their every project from scratch, they have a huge competitive advantage.

An interesting observation was to see, that the respondents mentioned the utilization of software product lines to have an impact on their own working experience. Developers, for example, can get a wider perspective on the development and the architecture of current technical solutions of the organization and learn new technologies. On the other hand, the coders want to be innovative and come up with new solutions, rather than just connecting existing solutions to each other. The work of a developer is changing, and it is more and more about interfaces and integration. Still, enriching the work of developers by feeding their desire to solve challenging technical challenges can have positive impacts on their satisfaction towards their jobs.

A lot of insights were collected on the views of SPL architectures and component reuse in the future. The respondents had quite similar views to the future – the companies want to be more able to respond quickly to changing customer needs by being able to make changes to the architecture easily. This

lowers the need for strict product lines. However, the future of systematic reuse is seen as bright, and all the interviewed organizations were implementing it. The organizations brought up the rise of the external component libraries, that support the organizations' own repositories and bring a lot of new possibilities to the development. The usage of external libraries is expected to grow in the future, and this means, that there will be more and more components available to use, especially offered by the large companies in the IT industry.

The questions about the benefits and shortcomings of product lines and component reuse caused a lot of discussion. As mentioned earlier, the main benefits that the interviewed organizations mentioned, were the movement of workforce, shared architecture and vision about the development in the firm, product quality and development efficiency. Naturally many other benefits were also mentioned, such as efficiency in testing the solutions and development's trust on the product itself, but the four main benefits were brought up by all the interviewees.

The following figure illustrates the results of the interviews and the main benefits that the SPL approach has for the organization itself. There were also other benefits, but the ones mentioned in the figure are the that were mentioned the most by the respondents. Component reuse is something you can do and have partly the same impacts, but a large-scale software product line brings more results in the long run. There is no SPL approach that does not involve component reuse. The benefits of both SPL and reuse have direct impacts on organizations capabilities and competitiveness. This figure was produced by combining the earlier literature and the results of this research. The figure itself does not consider all the benefits caused by SPL usage and reuse - the benefits mentioned in the figure are the strongest benefits that the method brings.

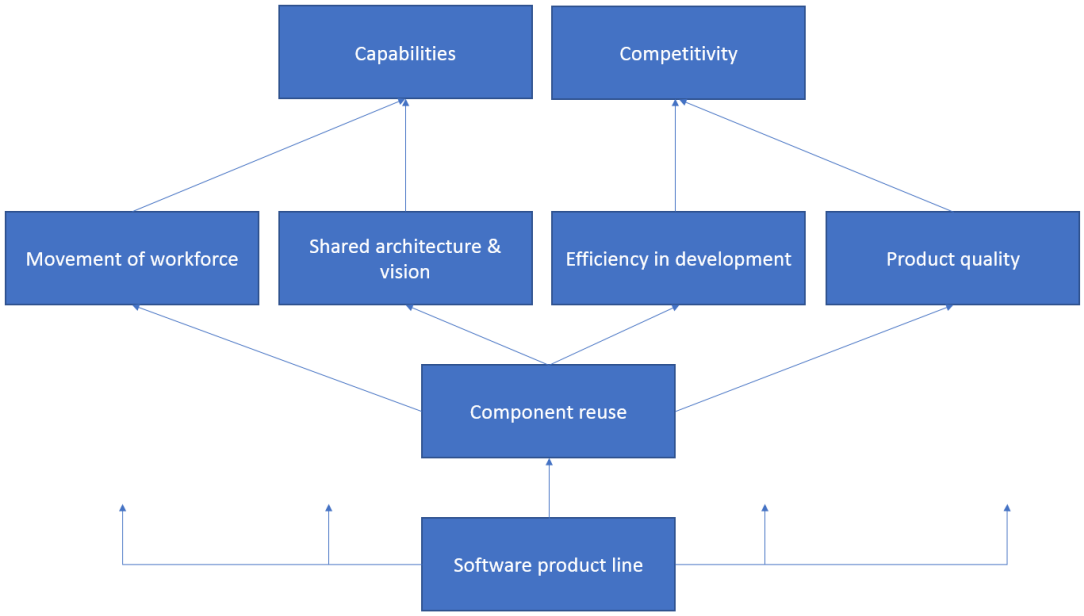


Figure 12- Benefits of SPL and component reuse.

The strongest potential shortcomings of product line architecture are mentioned in Figure 13. Problems caused by the method were slowness in responding to changing customer needs and the problems in deciding when to let go of a certain component. Also, problems in focusing on the long-term goals was mentioned.

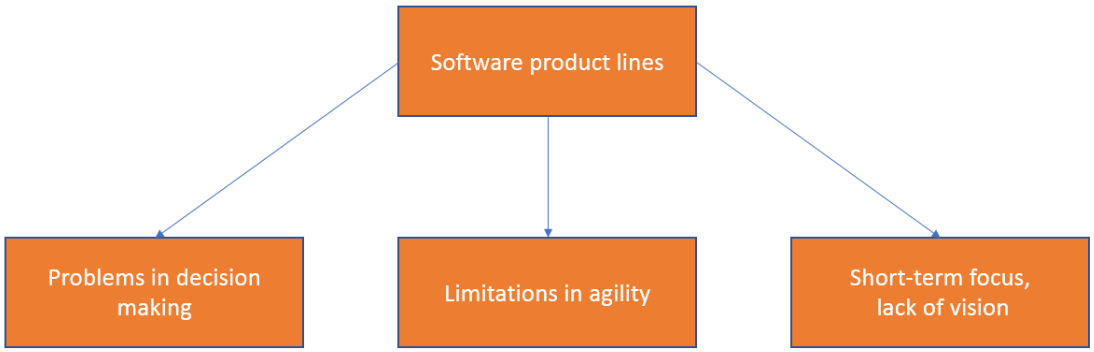


Figure 13 - Potential shortcomings of product line architecture

Not all these mentioned shortcomings follow the usage of SPL, but there is a risk of ending up with these problems, if the architecture is not built systematically and the big picture is not considered early enough. Being aware of these pros and cons, we must take into consideration the responsibility of an organization in taking the approach in to use. To be able to implement product line architecture and systematic reuse, the organization must have a certain level of maturity. If the organization is not ready for implementing the approach, it won't get as clear benefits as the ones that exceed the maturity level that is needed.

The respondents also mentioned that the interview itself was helpful and brought up some ideas for potential areas to work on in the future. They admitted, that it was good to experience the scientific point of view on the matter – they had mostly concentrated on the practical side.

From the results of the interview we can conclude, that SPL architectures are still used in many software companies and reuse is implemented in even more companies. Even though the approach has its flaws, more beneficial impacts were reported. According to the information gathered, the overall impact of SPL and reuse on capabilities and competitiveness is positive. This means that the approach is still widely used and brings results.

7 CONCLUSION

Software product lines and component reuse are both commonly used in software companies' development processes. Software product lines demand certain characteristics from an organization in order to succeed in implementing it, but component reuse is less complex to utilize – and is therefore used more widely. Software product lines are strict architectures that enable many benefits in the development, but it also brings some limitations to the organization, such as slowness in modifying the existing solutions (Hallsteinsen, Schouten, Boot & Faegri, 2006). The companies that utilize software product line architectures aim at gaining competitive advantage through achieving better efficiency in software development and minimizing the resources used for overlapping work (Northrop, 2002). It seems, that organizations need to fill certain kind of requirements in order to be able to implement product line approach, whereas reuse is done in most organizations – but there are many ways it is implemented and the commitment to it varies a lot.

Component reuse is seen as a crucial part of the development process for any software companies that offer multiple products or software projects to their clients. By implementing systematic component reuse the companies can appear a lot larger than they are by gaining speed in development and strengthening the quality of the product. Reused components can be seen both as technical components and business components. These views gathered from the interviews are also in line with the findings of Clements and Northrop (2003).

This study aimed at finding answers to following questions: What are the impacts of software product line approach and reuse in organizations' capabilities and competitiveness, how organizations implement these approaches and how component reuse affects the new product development. As basis, a wide set of earlier research papers were used to get a broad view of the software product line architecture, its benefits and shortcomings. The empirical part of the research brought both convergent and new views to the matter.

In the empirical part of the research the answers to the research questions were looked for. In total 7 interviewees from 5 different companies were inter-

viewed. With semi-structured theme interviews, the aim was to get a realistic view of the current development process of the companies by being able to concentrate on the topics that the respondent had a lot of views about. Companies with various types of business strategies and businesses overall brought a lot of insights on the matter from different perspectives.

As answers to the research questions, the research provided the following: The utilization of software product lines does have an impact on capabilities and competitiveness of a company. For example, interviewed organizations 2 and 4 were able to reach both short- and long-term benefits by using SPL architectures and implementing systematic reuse. They saw their way of working as a factor to competitive advantage, in comparison to their competitors. The reuse of both business and software components also speeds up the new product development process and gives smaller companies an ability to compete with the larger ones. In software companies overall, these methods are used widely, but mostly in product-oriented companies, which offer multiple software products to their customers. Implementing systematic reuse of components is more common than implementing large-scale product lines within the company. This is because the implementation of SPL demands larger up-front investment to begin with.

Implementing the SPL architecture takes time but the interviewees brought up, that it also spreads the knowledge within the company and therefore raises the level of capabilities of each developer, for example. It was also noted, that developers demand challenges and want to be innovative. This was also brought up by Sojer and Henkel (2010), who found that integrating challenges into developers' tasks that are in the best interest of the firm. The most commonly mentioned benefit, faster product-to-market, also raises the level of competitiveness by being able to react faster to the market need than the competitors.

The key to successful implementation of product line architecture is to ensure the commitment of the whole organization to the new approach. The process takes time, and therefore also resources from the company. These answers are in line with the views of Käkölä and Duenas (2006). Still, the commitment to this method pays off in the long-term. SPL architecture brings positive impacts such as product quality, more efficient product development, easier movement of workforce between projects and a shared vision for the company to pursue. All of these are connected to capabilities and competitiveness of a company. These views are in line with the research of Wesselius (2006). In new product development the main benefit is the decreased time-to-market per product. However, the utilization of the method can also bring some downsides, such as slowness in reacting to customer's requests for customization and the risk of using "outdated" components in the key solutions. A successful product line process consists of producing reusable components, brokering components and consuming the components (Forsell, 2002). Product line architecture must be implemented throughout the organization to maximize the benefits. The organizations seemed to have problems in measuring the impacts of the product line

architecture as it is difficult to differentiate the impacts to be only a cause by this method. In the interviews, we managed to find some ways to measure the impacts, such as observing the running times of the projects and efficiency of smaller sprints in software development. Both theoretical and empirical parts of the research supported the fact that product line activities are only causing real impacts when the management is committed to the implementation, along with the technical teams. As mentioned by Frakes and Kang (2005), for example cost-benefit analysis, assessing maturity, reusability assessment models and failure modes can work as measurement tools and metrics for component reuse impacts.

Organizations had various approaches to utilize the SPL method and reuse. All of them were reusing components, but only two organizations out of five claimed to use full-on product line architectures. Usually the problem was the amount of work and investment to be made to begin the usage of the method. The approach is not easy to sell for the management, as it does not bring results right away. Still, it is used in many software companies and it brings competitive advantage to its users. The SPL approach has a significant impact on new product development of software companies. Especially the small companies can boost their new product development significantly by reusing the solutions from existing products and implementing product line architecture. With this approach, overlapping work is avoided and smaller companies can appear bigger than they actually are by producing new products fast. This also benefits customer experience and their commitment to the service provider.

The interviewees mentioned, that implementing SPL architecture demands large up-front investment and time. The investment made is bigger when compared to single system, but when having over three products in the product line, the architecture starts to pay off as savings in development resources and code with better quality. Sharing the same platform might have its difficulties from time to time, but the long-term results support the usage of product lines architecture, as also mentioned by Van der Linden, Schmid and Rommes (2007). Earlier studies showed examples of companies succeeding in the implementation of SPL and achieving competitive advantage through it – even though the challenges in measuring the impacts of product line architecture still exist.

The results also showed differences in the views about the usability of product lines. Product-oriented companies seem to benefit from reuse and product line architecture a lot more than project-oriented ones – and therefore it is more widely used in companies offering multiple products. Their business and solutions usually have a lot in common, and therefore the repeatability is a lot higher. This makes implementing the software product line architectures a lot easier. The views of the interviewees were also in line with this – product-oriented companies were all aware of the possibilities that the approach brings, but the project-oriented ones seemed to see more shortcomings than benefits. Also, the management and the developers seemed to have a little bit different views on the way that the method is implemented – the members of manage-

ment tended to say their processes are entirely product line-based, but the developers saw more flaws in the reuse process and SPL architecture within the organization, when compared to the management.

When comparing agile methods to product lines and component reuse, the views on which brings more results vary (Ahmed & Capretz, 2010). This was also noted during the interviews. Over the years, agile methods have raised a lot of attention, but it also has its flaws. The interviews brought up, that resource planning is quite hard with agile principles when compared to product line architectures. Even though agile methods make organization more flexible to the customer's demands, they do not always bring the best result. Also, the final output made with agile methods can become fragmented, when different teams work with different cycles. SPL usage and reuse, however, promote quality and it is easier to plan the usage of resources.

In general, component reuse is becoming more and more common – and more importantly, more systematic. Many companies stated, that it is essential if you want to keep up with the competition. Component reuse is not anymore only reusing the components that the organization itself has produced, but also using external open source libraries to support the efficiency of the development process. The leading software companies are producing component libraries that can be used by the customers. If someone has already come up with a solution that can be used in the product, it saves a lot of resources to use it. Even though the developers might have to learn new languages and solutions to understand and modify the solutions, it still beneficial for the company. In the future, the importance of component reuse will continue growing larger. The field of software development develops rapidly, and the solutions are more and more about reused components and having interfaces that support the efficient integration of external solutions. It seems, that the more customer-centric thinking and the changing customer needs are affecting the usage of software product lines – in the sense, that the organizations do not utilize strict software product lines that often anymore but implement systematic reuse and react to customers' needs by customizing the products when needed.

Overall, both the theoretical and the empirical part gave a lot of insights to the matter and the research questions. The research can be seen as valid and reliable, but the results might not be generalizable. Validity and reliability could have been made higher by having more companies attending the interviews. Also, all the companies that were interviewed are Finland-based and mostly small companies, which can also indicate limitations of this study. This study does not bring insights on how to promote the usage of SPL within an organization, and it does not offer frameworks to help the companies start the implementation of the method.

The goal was to find information about different ways of implementing these methods and the ways the organizations rationalize and measure the utilization of the method. The selected organizations were deliberately chosen to be different and their views on SPL and reuse varied a lot. For further research on the matter, we could focus more on product-oriented organizations, as they

seem to be more positive towards the approach. The results of this research support the differences between product- and project-oriented organizations. By engaging more product-oriented organizations to the research we could get a wider view of the ways the organizations utilize the method. An interesting question would be, how to make the process of utilizing the method easier. If the starting process would be more efficient and took less resources, the threshold to start implementing the approach would be significantly lower. If the upfront investment was lower, it would be less of a risk to implement and try if the approach fits the organization. But as it came up in the study, it is also about whether the organization fits the approach and has the characteristics to implement it efficiently. The larger the organization is, the more resources it demands to utilize the approach. As mentioned earlier, the level of efficiency can grow rapidly especially in the new product development of small organizations – often making a proof-of-concept solution for the potential customer can prove that the small organization can offer all the services that the customer needs. This is a clear benefit on the competitiveness of the company.

The utilization of SPL and its impacts was a very interesting topic to do research on. The rapidly changing trends in the software industry demands ability to change from the companies. Interesting research topics in the future could be for example the usage of external component libraries and their growth, a hybrid-model between agile methods and reuse, as well as evaluating the software architectures between companies that use product line approach and those who lean on agile methods. The overall objectives of SPL and agile methods have a lot of similarities, and there has been some studies on the matter. Integration of SPL and agile methods can effectively address time-to-market constraints, new requirements and the need for product variety (Mohan, Ramesh & Sugumaran, 2010). Also, there has been studies on the development of product lines, for example dynamic software product lines (DSPL) and software ecosystems. Evaluating these concepts and their impacts would be good research topics in the future.

REFERENCES

Ahmed, F., & Capretz, L. F. (2010). An organizational maturity model of software product line engineering. Software Quality Journal, 18(2), 195-225.

Böckle, G., Clements, P., McGregor, J.D., Muthig, D. & Schmid, K. (2004). Calculating ROI for software product lines. IEEE software, 21(3), 23-31.

Bosch, J. (2009). From software product lines to software ecosystems. In Proceedings of the 13th international software product line conference (pp. 111-119). Carnegie Mellon University.

Bosch, J., & Bosch-Sijtsema, P. (2010). From integration to composition: On the impact of software product lines, global development and ecosystems. Journal of Systems and Software, 83(1), 67-76.

Bosch, J., & Bosch-Sijtsema, P. M. (2011). Introducing agile customer-centered development in a legacy software product line. Software: Practice and Experience, 41(8), 871-882.

Bosch, J., & Eklund, U. (2012). Eternal embedded software: Towards innovation experiment systems, In Leveraging Applications of Formal Methods, Verification and Validation. Technologies for Mastering Change, p. 19-31, Springer; Berlin, Heidelberg.

Calantone, R. & Di Benedetto, C. (1988), An integrative model of the new product development process: An empirical validation. Journal of Product Innovation Management: An international publication of the product development & management association, 5(3), 201-215.

Clark, K. B., T. Fujimoto. (1991). Product Development Performance. Harvard Business School Press, Cambridge, MA.

Clements, P. & Northrop L. (2003). Product line systems program. Software Product Lines: A New Paradigm for the New Century. The Journal of Defense Software Engineering.

Clements, P., Kazman, R. & Klein, M (2001): Evaluating a software architecture.

Cooper, R. (2001). Winning at New Products, Perseus Publications, Cambridge, MA, 20.

Cooper, R. (1990): A new tool for managing new product process: Steps, deficiencies and impact. Business Horizons, May-June 44-56, 1990.
Dynamic software product lines.

Engström, E., & Runeson, P. (2011). Software product line testing–a systematic mapping study. Information and Software Technology, 53(1), 2-13.

Figueiredo E, Cacho N., Sant'Anna, C., Monteiro, M., Kulesza, U, Garcia, A., Soares, S., Ferrari, F., Khan, F. , Castor Filho, F. & Dantas, F. (2008), Evolving Software Product Lines with Aspects: An Empirical Study on Design Stability.

Forsell, M. (2002). Improving component reuse in software development (No. 16). University of Jyväskylä, Jyväskylä studies in computing.

Forsell, M., Halttunen, V., & Ahonen, J. (2000, June). Use and identification of components in component-based software development methods. In International Conference on Software Reuse (pp. 284-301). Springer, Berlin, Heidelberg.

Frakes, W. B., & Kang, K. (2005). Software reuse research: Status and future. IEEE transactions on Software Engineering, 31(7), 529-536.

Frakes, W., & Terry, C. (1996). Software reuse: metrics and models. ACM Computing Surveys (CSUR), 28(2), 415-435.

Gomaa, H. (2005). Designing software product lines with UML. (pp. 160-216). IEEE.

Grant, R.M. (1996): Prospering in Dynamically-Competitive Environments: Organizational Capability as Knowledge Integration. Organization Science, Vol. 7, No. 4 (Jul. - Aug., 1996), pp. 375-387

Griffin, A. (1997). Drivers of NPD Success: The 1997 PDMA Report, Product Development & Management Association, Chicago,

Haefliger, S., Von Krogh, G., & Spaeth, S. (2008). Code reuse in open source software. Management science, 54(1), 180-193.

Hallsteinsen, S., Hinchey, M., Park, S., & Schmid, K. (2008). Dynamic software product lines. Computer, 41(4), 93-95.

Hallsteinsen, S., Schouten, S., Boot, G. & Fægri, T. (2006). Dealing with architectural variation in product populations. In: Käköla T., Duenas J.C. (eds) Software Product Lines. Springer, Berlin, Heidelberg. (pp. 245-273.)

Hinchey, M., Park, S., & Schmid, K. (2012). Building dynamic software product lines. Computer, (10), 22-26.

Hirsjärvi, S., & Hurme, H. (2001). Teemahaastattelu: Teemahaastattelun teoria ja käytäntö. Yliopistopaino.

Holmström, H. & Bosch, J. (2013): Towards Data-Driven Product Development: A Multiple Case Study on Post-Deployment Data Usage in Software-Intensive Embedded Systems.

Kakölä, T. (2003, January). Software business models and contexts for software innovation: key areas software business research. In 36th Annual Hawaii International Conference on System Sciences, 2003. Proceedings of the (pp. 8-pp). IEEE.

Käkölä, T. & Duenas, J.C. (2006): Software product lines. Research issues in engineering and management. Preface. Springer, 2006.

Kusunoki, K., I. Nonaka, A. Nagata. (1998). Organizational capabilities in product development of Japanese firms: A conceptual framework and empirical findings. Organ. Sci. 9(6) 699–718.

Lee, J.N. (2001) - The impact of knowledge sharing, organizational capability and partnership quality on IS outsourcing success. Information & Management, 2001 – Elsevier.

Lesser, E. & Ban, L. (2016): How leading companies practice software development and delivery to achieve a competitive edge, Strategy & Leadership, Vol. 44 Issue: 1, pp. 41-47.

Liao, S. H., Fei, W. C., & Chen, C. C. (2007). Knowledge sharing, absorptive capacity, and innovation capability: an empirical study of Taiwan's knowledge-intensive industries. Journal of information science, 33(3), 340-359.

Lim, W. C. (1998). Managing software reuse: a comprehensive guide to strategically reengineering the organization for reusable components. Prentice-Hall, Inc.

Mansell, J. (2006). Experiences and expectations regarding the introduction of systematic reuse in small- and medium-sized companies. In: Käkölä T., Duenas J.C. (eds) Software Product Lines. Springer, Berlin, Heidelberg. (pp. 91-124.)

Metsämuuronen, J. (2005). Tutkimuksen tekemisen perusteet ihmistieteissä.

Metzger, A., & Pohl, K. (2014, May). Software product line engineering and variability management: achievements and challenges. In Proceedings of the on Future of Software Engineering (pp. 70-84). ACM.

Mohagheghi, P., & Conradi, R. (2007). Quality, productivity and economic benefits of software reuse: a review of industrial studies. Empirical Software Engineering, 12(5), 471-516.

Mohan, K., Ramesh, B., & Sugumaran, V. (2010). Integrating software product line engineering and agile development. IEEE software, 27(3), 48-55.

Nambisan, Satish. (2003). Information Systems as a Reference Discipline for New Product Development. MIS Quarterly, Vol. 27, No. 1 (Mar., 2003), pp. 1-18.

Northrop, L. M. (2002). SEI's software product line tenets. IEEE software, 19(4), 32-40.

Paulk, M. C., Curtis, B., Chrissis, M. B., & Weber, C. V. (1993). The capability maturity model for software. Software engineering project management, 10, 1-26.

Paulk, M. (2002). Capability maturity model for software. Encyclopedia of Software Engineering.

Pavlou, P.A. & El Sawy, O.A: (2006): From IT Leveraging Competence to Competitive Advantage in Turbulent Environments: The Case of New Product Development. Information Systems Research, 17(3), 198-227.

Rabiser, R., Schmid, K., Becker, M., Botterweck, G., Galster, M., Groher, I., & Weyns, D. (2018, September). A study and comparison of industrial vs. academic software product line research published at SPLC. In Proceedings of the 22nd International Conference on Systems and Software Product Line-Volume 1 (pp. 14-24). ACM.

Sojer, M., & Henkel, J. (2010). Code reuse in open source software development: Quantitative evidence, drivers, and impediments. Journal of the Association for Information Systems, 11(12), 868-901.

Stonehouse, G., & Snowdon, B. (2007). Competitive advantage revisited: Michael Porter on strategy and competitiveness. Journal of Management Inquiry, 16(3), 256-273.

Ulrich, K., and Eppinger, S. D. (2000). Product Design and Development, McGraw-Hill, New York.

Utterback, M., Abernathy, W. (1975): Dynamic Model of Process and Product Innovation. Omega, 3(6), 639-656.

Veryzer, Robert (1998): Discontinuous Innovation and the new product development process. Elsevier, Product innovation management 1998;15:304-321.

Wesselius J.H. (2006). Strategic Scenario-Based Valuation of Product Line Roadmaps. In: Käkölä T., Duenas J.C. (eds) Software Product Lines. Springer, Berlin, Heidelberg. (pp. 53-89.)

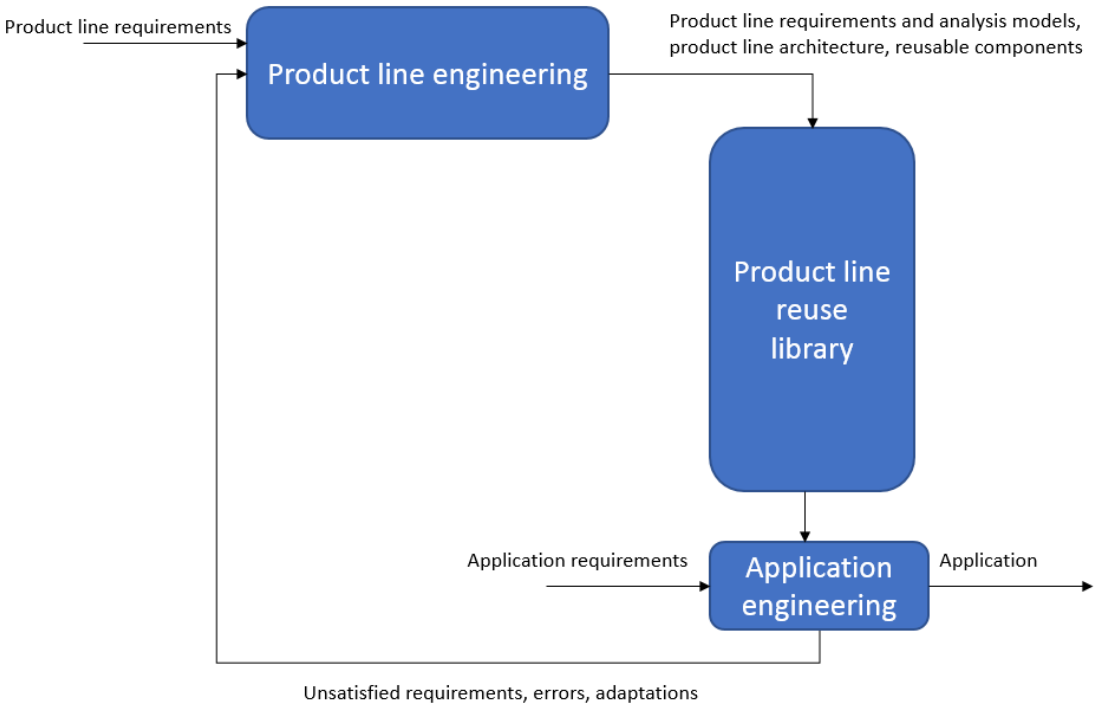
Yee, K. P., & Eze, U. C. (2012). The influence of quality, marketing, and knowledge capabilities in business competitiveness. International Journal of Innovation and Learning, 11(3), 288. doi:10.1504/ijil.2012.046067

APPENDIX 1

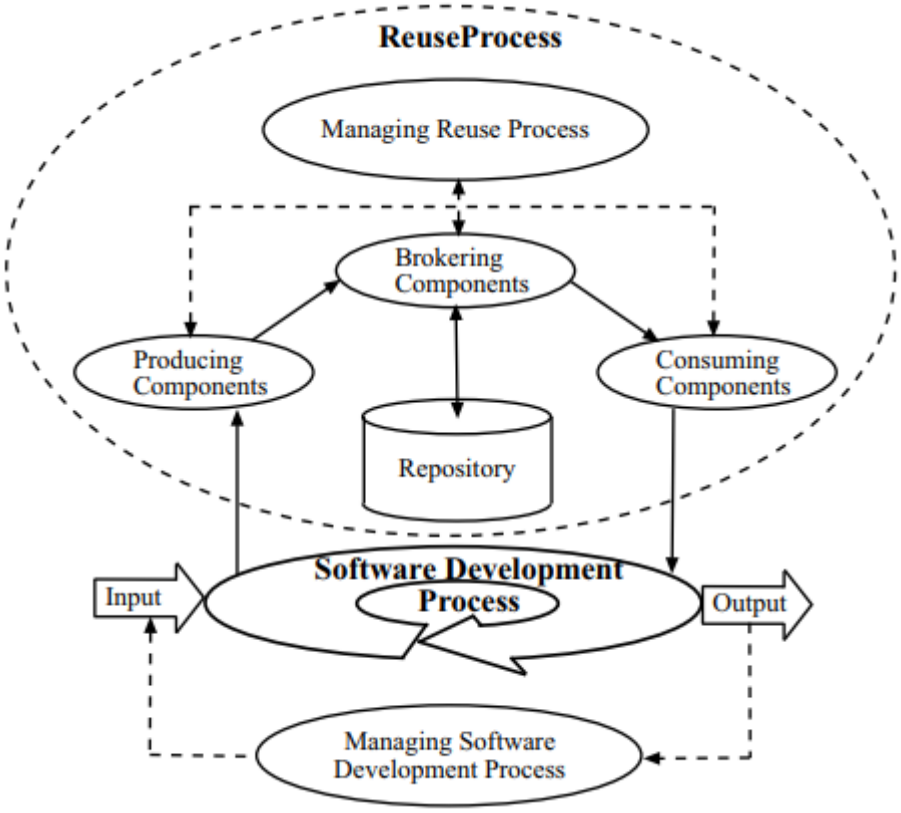
Empirical part , Interview plan

Structure of the interview:

First, the topic of the research was explained briefly and the interviewee is informed about what is meant by component reuse and software product lines. The following examples were presented:



Cycle of the software product line process. (Gomaa, 2005)



Software component reuse process. (Forsell, 2002)

1. **General information about the interviewee**

- Describe your position and tasks in the firm
- Have you received any education about Software product lines or component reuse?
- How does component reuse relate to your job?
- What kind of experience do you have in the field, and how long have you worked in your position?

2. **General information about the company**

- How many employees work in the company?
- What is the annual turnover?
- What kind of services does your company provide?
    - How many different systems or services does your company have?
        - 1-2
        - 3-4
        - 5+
    - Are the products SaaS-products or separate custom-made systems for clients?

- How many different products do you have?
    - What is the main product about?
    - Are the other products related to the same field?
- What is your organization's field of expertise?
- What kind of services does your company provide?
- How many different systems or services does your company have?
    - 1-3
    - 3-5
    - 6+
- Do all of them share the entirely or partly the same technical architecture?
    - Why?
    - Why not?
- Would you agree that the mutual architecture in the products would be suitable for entirely a new kind of software service?
    - partly
    - fully
    - not at all

3. Software product lines

- What is your current experience of Software product lines? Does your company have such architecture?
- What are the potential benefits and shortcomings of the method?
- How is the usage of product lines acknowledged and documented?
- Do you see SPL as an important part of development?
- Is this done systematically?
- What kind of problems can product lines cause?
- Can an organization gain competitive advantage through SPL usage? How?
- Does this affect the capabilities of the organization?

4. Component reuse

- Is this executed in your company?
- Are the reused components seen as only technical components?
- Can also business components be reused?
- How does component reuse affect new product development?
- Is there any data supporting the benefits of reuse?
- What are the links between systematic component reuse and organizational capability?
- What is the impact of reuse on competitiveness?

5. Linkage between the terms

- Reuse is seen as a part of software product line thinking. How do you see the relationship between software product lines and component reuse?
- How is successful use of SPL linked in to success of a company?
- How is component reuse related to organizational capability?
- What kind of measurement tool could be used to measure the impacts?
- How do you see the role of reuse in the future?